PRENTICE HALL INTERNATIONAL PERSONAL COMPUTER BOOK

MASTERING YOUR MACINTOSH

WILLIAM SKYVINGTON

MASTERING YOUR MACINTOSH

A User's Guide to Apple's Macintosh Computer

MASTERING YOUR MACINTOSH

A User's Guide to Apple's Macintosh Computer

William Skyvington

Prentice/Hall PHI International

Englewood Cliffs, N.J. London New Delhi Rio de Janeiro
Singapore Sydney Tokyo Toronto Wellington

The cover photograph has been kindly supplied by William Skyvington.

Photographs and picture of Japanese girl reproduced by permission of Apple Computer Inc.

Library of Congress Cataloging in Publication Data
Skyvington, William.
Mastering your Macintosh.
Includes index.
1. Macintosh (Computer) I. Title
QA76.8.M3S59 1984 001.64 84-16071
ISBN 0-13-559527-4 (pbk.)

British Library Cataloging in Publication Data
Skyvington, William
Mastering your Maintosh
1. Macintosh (Computer)
I. Title
001.64'04 QA76.8.M3
ISBN 0-13-559527-4

ISBN 0-13-559527-4

PRENTICE-HALL INTERNATIONAL, INC., London
PRENTICE-HALL OF AUSTRALIA PTY., LTD., Sydney
PRENTICE-HALL CANADA, INC., Toronto
PRENTICE-HALL OF INDIA PRIVATE LIMITED, New Delhi
PRENTICE-HALL OF JAPAN, INC., Tokyo
PRENTICE-HALL OF SOUTHEAST ASIA PTE., Ltd., Singapore
PRENTICE-HALL INC., Englewood Cliffs, New Jersey
PRENTICE-HALL DO BRASIL LTDA., Rio de Janeiro
WHITEHALL BOOKS LIMITED, Wellington, New Zealand

Typeset by Communitype Ltd., Leicester, England
Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

For my friends at Apple France,
members of the Lisa/Macintosh team,
who enabled me to write this book:

Bruno Rives
Miriam Souza
Neil Minkley
Christophe Droulers
Alain Andrieux
Christine Thevenot

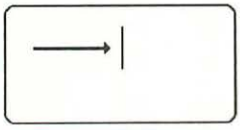
A mountain in labor cried so loud
that everyone, hearing the noise,
ran up expecting that she would
surely give birth to
a city bigger than Paris;
she brought forth a mouse.

— Jean de la Fontaine, *Fables* V.10

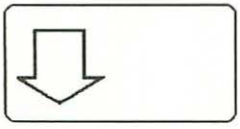
There are two types of Macintosh keyboard: this book follows the style of the American version, which uses words to denote the keys.

Readers who are using the 'International' style of Macintosh keyboard will already be aware that the symbols on their keyboard indicate the following:

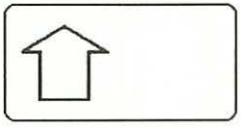
Tab



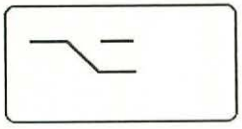
Caps Lock



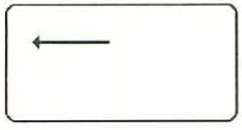
Shift



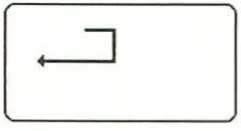
Option



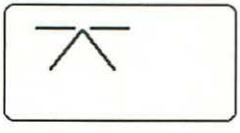
Backspace



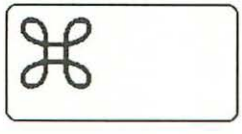
Return



Enter



Command



Contents

Introduction

On 24 January 1984, in a series of gala events throughout the world, Apple Computer unveiled its latest machine: *Macintosh*. Viewed from the outside, this personal computer is a beige box that incorporates a small black-and-white screen and a built-in disk drive. There is a separate keyboard, but most commands are transmitted to the computer by means of the *mouse*, which is a hand-held device about the size of a cigarette box. When the user glides the mouse over the table alongside the computer, a pointer on the screen of Macintosh reproduces an analogous movement, making it possible for the user to actually point at various objects (words or images) displayed on the screen. There is also an elegant printer, called the *Imagewriter*, that can produce a hard copy of any text or graphic elements displayed on the screen.

Priced in the vicinity of $2,500, Macintosh has been designed primarily for the small-business market, but the machine should become an ideal companion for knowledge workers in many different professional spheres: engineers, architects, lawyers, writers, research workers. . . . University students, too, will appreciate the high-resolution graphic capabilities of Macintosh, which make it an attractive educational tool.

From the very first instant that you come into contact with Macintosh, you realize that this machine is very different from conventional microcomputers, particularly because there are so many operations that can be carried out effortlessly by means of the mouse.

No doubt the most effective manner of getting the feel of Macintosh, at the beginning, is to play around with the entertaining demonstrations provided by Apple on their *Guided Tour* disk. At one point, for example, an entire piano keyboard appears on the screen, and you suddenly discover that you can play a melody in real time simply by pointing at the notes with the mouse and pressing the button on its back. Later on, you use the mouse to point at top hats that are lined up in front of a magician, and the contents of the various hats appear on the screen each time you press the mouse button.

When it arrived on the market, Macintosh was armed above all with two sophisticated software products developed directly by Apple Computer: a word processing system called *MacWrite*, and a remarkable graphics design tool called *MacPaint*. These two products are fundamental in the Macintosh context, not only because they can be used conjointly to create top-quality documents containing both

text and images, but because they provide an admirable and exhaustive demonstration of the performance and power of the new machine. Two other application programs have been developed directly by Apple Computer: a second graphics package named *MacDraw*, for structured drawings, and a modeling tool named *MacProject*. In this user's guide, a full chapter is devoted to each of these four products.

At least two years before the official announcement of Macintosh, Apple Computer started to contact over a hundred software and hardware firms, with a view to encouraging them to develop projects for the new machine. Relatively few of these products have yet appeared on the market, for the simple reason that the definitive specifications of Macintosh were finalized by Apple Computer only shortly before the public announcement of the machine, and so the developers have needed time to test the final versions of their products before releasing them on the market. Several products have nevertheless already appeared, including Microsoft's *Multiplan* and *Chart*. A chapter of this guide is devoted to each of these products.

There is an interesting general question that must be brought up in the case of Macintosh, namely: How will the machine be programmed? Up until now, as most professionals know, microcomputing has been pervaded by a slight misunderstanding. Customers purchasing microcomputers were generally told that these machines could be programmed using an elementary language, *Basic*, which is very easy to learn. Consequently, many of these newcomers to microcomputing were under the illusion that it would be easy for them to develop their own software. This, of course, was not quite true; software development has always been — and still is — an extremely difficult mixture of art, craftsmanship and industry . . . not to speak of imagination. Moreover, few developers would choose an "easy" language such as Basic for a major project.

In the case of Macintosh, this situation has been brought out into the open in a commendable fashion. Customers who are interested in becoming computer literate (and why not?) can choose between several interesting and easy-to-learn language systems: *Macintosh Basic*, *Microsoft Basic*, *Macintosh Pascal*, *Logo*, etc. An individual might even like to tackle the creation of simple but worthwhile programs using one or other of these elementary languages. On the other hand, Apple Computer has made it clear that it is quite impossible to use Macintosh itself, combined with one of the above-mentioned language systems, to develop software products that would look and behave, for example, like *MacWrite* or *MacPaint*. This question is dealt with in greater detail in the two appendices attached to this book, but it can be said for the moment that the creation of top-notch software products for Macintosh is an affair that is best left to experienced professionals, for the ideal solution consists of carrying out such development, not on Macintosh itself, but on a *Lisa*.

Mastering the Macintosh is child's play in the sense that nearly everything takes place under the control of *menus* that present a list of all the operations that you can carry out at any particular moment, making it practically impossible to mess up things by requesting unfeasible actions. Most users of the older generation of micro-

computers have had ample opportunities of experiencing the vague fear that might be termed the "crash syndrome". We were aware at most times that we only had to make one false move — which meant hitting a wrong key — and the machine would start beeping back at us in an unfriendly manner, with equally nasty hieroglyphics displayed on the screen to inform us (if we were capable of interpreting them) why the computer refused to carry on with our job. With Macintosh, this anguish no longer exists, for you soon manage to establish an *intuitive* relationship with the computer, and one has the impression that nothing can really go wrong.

The so-called *icons* that you encounter constantly on the Macintosh screen play an essential role in establishing this excellent man/machine interface. What is an icon? This fine old word, borrowed from Ancient Greek, simply means "image". In the case of Macintosh, an icon is a tiny illustration that represents a particular *thing* . . . such as a document, for example, or even a trash can. There is no better way of describing the intuitive man/Macintosh relationship than to look at the way in which you get rid of a document that has become obsolete. You simply use the mouse to drag the document icon across the screen so that it coincides with the

trash can icon, then you release the button on top of the mouse in order to indicate that the document is to be deposited in the trash can.

The documentation that accompanies Macintosh is remarkably well produced and printed by the machine itself. Some of the remarks seem strange for a computer manual. We are told, for example, that Macintosh — which consumes very little electricity (about the same as a 60-watt light bulb) — can be left switched on overnight, and that "it makes a fine night light". We learn, too, that there is a clock inside the machine, powered by a battery that keeps it ticking even when Macintosh is not plugged into a socket, and that this timekeeper was originally set to January 1, 1904. Elsewhere, we are informed that a Macintosh can be stored under the seat of most commercial airplanes, or in the overhead compartment. Other anecdotes concerning this revolutionary machine describe its habit of displaying an icon to either smile at you, or look sad, depending on whether or not everything is going along fine. And there is a brief mention of a puzzle — fifteen interchangeable numbered "tiles" in a square box — that can be displayed on the screen to entertain you whenever you get tired of using the more serious attributes of the computer.

It appears that the signatures of the forty seven team members at Apple Computer who designed Macintosh have been molded into the interior surface of the plastic that encases the machine. Funnily enough, there is no direct means of verifying this fact, since the manual threatens you with high-voltage death if ever you attempt to take the cover off the machine. (This is likely to shock the older generation of Apple II users, many of whom got used to working constantly with the lid off their machine, to facilitate the changing of electronic boards.) If the design of the machine is the outcome of a lot of creative imagination and hard work, it also represents a large sum of money. Apple invested over $100 million on the project, then they spent another $20 million to set up a Japanese-style robot factory, in the heart of Silicon Valley, to manufacture the computer, at the rate of one Macintosh every 27 seconds. The truth of the matter is that this robot factory also employs some 300 human workers, but this means that less than 1 per cent of the cost price of Macintosh is due to labor.

One final anecdote before we get on with the book: there exists a disk for Macintosh that enables the machine to talk back at you, describing itself in a broad American accent. There is no doubt whatsoever that little Macintosh has a whole bag of tricks in store for anyone who would like to see what avant-garde computing is all about. . . .

Image of a machine

Since 1983, the notion of machine *families* can be applied to the various products manufactured by Apple Computer. In order to show where Macintosh fits into the picture, let us take a glance at Apple's brief but spectacular history. It has often been pointed out that, back in 1976, the future company's first machine was built by Steven P. Jobs and Stephen G. Wozniak, not in a $20-million robot-controlled factory, but in a backyard garage. Apple Computer came into existence as a corporation in 1977, assisted by a smart businessman named Mike Markkula. Six years later, by the middle of 1983, they had sold a million machines.

The company's unique product, for several years, was the *Apple II* personal computer, based upon the 6502 processor. In 1981 a second product line was announced: the *Apple III*. Although this new and far more powerful machine was based upon the same 8-bit chip as the Apple II (which had evolved by now into what was called the *Apple II Plus*), the two computers used completely different operating systems, and there was no compatibility whatsoever between them.

Several important events occurred in 1983 which were to change considerably the Apple scene. On the one hand, a phenomenal but expensive new computer was created: *Lisa*, built around the 32-bit MC68000 processor designed and manufactured by Motorola Semiconductor in Texas. On the other hand, a sophisticated variant of the company's initial machine came into existence: the *Apple IIe*. Little by little, the compatibility barriers between the Apple II and the Apple III were broken down, above all by the introduction of a new operating system called *ProDOS*, which meant that one could at last speak of a 6502 family of Apple personal computers. And this is the context in which Macintosh was announced at the beginning of 1984. . . .

The processor inside Macintosh is an 8MHz version of the same MC68000 chip used for Lisa, which means that these two machines can henceforth be thought of as forming Apple's top-of-the-line *32 Supermicro* family.

The black-and-white Macintosh *screen* has a resolution of 512 × 342 pixels. Although it is only some 18cm wide and 14cm high (9-inch diagonal), this screen is exceptionally pleasant to work with. The text and images are clearcut, free from flicker, and highly readable. The user has no sensation of eye strain or fatigue, as with most monochrome monitors of earlier generation microcomputers.

It is interesting to note that the 175,104 pixels on the Macintosh screen are *bit-mapped*, which simply means that there is a specific bit in the memory of Macintosh for every dot on the screen. It is because of this property that you can actually drag images across the screen, in real time, using the mouse. Furthermore, besides black and white, the high-resolution screen allows many shades of "color", from light to dark gray.

Naturally, one might regret that an avant-garde computer such as Macintosh does not offer us the familiar phenomenon of color graphics . . . but it is a fact that this would have greatly increased the overall price of the machine.

On the front face of Macintosh, just below the screen, there is the narrow opening of a built-in disk drive. The disks themselves are the standard 3.5-inch

model developed by Sony. Single-sided and protected by a hard plastic case, a disk can store up to 400K of data.

Besides all the hardware features mentioned up until now, the main Macintosh unit contains a battery-powered *clock/calendar* that provides vital data both for application programs and for the system software. There is also a built-in *polyphonic sound generator* that includes a digital/analog converter that samples at a rate of 22KHz, making it possible to produce music or even human speech, which can be heard over a built-in loudspeaker. A port at the back of the machine allows such audio output to be sent, say, to an external amplifier.

In spite of all these attributes, the central Macintosh unit — whose upper face has been molded into the shape of a handle, so you can carry it around — is no bigger than a cube with edges of about 30cm, and it weighs only 9Kg. (Somebody has said, quite rightly, that Macintosh, when looked at from the side, reminds you a little of the profile of the famous cinematographic visitor from outer space named *E.T.*)

One amusing feature of the main Macintosh unit is so rare in the microcomputer universe that it deserves to be mentioned. There is not a single word on the facade of the machine, not even "Macintosh" . . . only the multicolored Apple icon. (Let's get used to using the word "icon".) If you search carefully, you'll discover a knob that controls the screen brightness, and that's about all.

Search even more carefully, just to the right of the horizontal disk-insertion slit, and you'll discover a tiny hole. Apple employees at Macintosh instruction classes get all panicky if ever a student asks them to explain what is the purpose of this hole. The truth of the matter is that, if ever a disk refuses to obey software attempts to eject it from the machine, you can stick a pin in this hole in order to activate what might be termed "hardware disk-ejection mode". Two *don'ts*: (1) Don't stick a pin in this hole, to eject a disk, unless you're really forced to. (2) Please don't tell anybody from Apple who told you what this hole is all about.

Peripheral devices

The functions of the Macintosh *mouse* will be dealt with exhaustively in the following chapter, and so the present discussion will be limited to a brief physical description of the beast itself.

It is difficult to imagine why anyone would have chosen to call this delightful instrument a "mouse". One thinks of the mouse as a nasty little animal that runs around erratically on its own, whereas the docile and well-behaved Mac mouse has to be moved around by the human hand, in orderly trajectories, almost like a chesspiece. It might have been called a *Queen*, for example. . . .

Be that as it may, the mouse attached to Macintosh is an ingenious input device. It consists of a rubber ball that seems to float freely under a sort of plastic tortoise shell. When you push the tortoise — sorry! the mouse — over a hard flat

surface such as a table top, the ball revolves as a result of the friction between it and the table. This movement of the rubber sphere, once captured by other components within the mouse, enables Macintosh to determine, in real time, both the direction in which the mouse is being moved, and the distance it is covering. In other words, the computer is capable of following the movements of the mouse.

The general idea is that, if you roll the mouse over the table in a sweeping curve, for example, then the pointer on the Macintosh screen will also trace out a curve of much the same form. Why "*much* the same", instead of "*exactly* the same"? There are both scaling and directional differences between mouse movements on the table top, and the corresponding pointer movements on the Macintosh screen . . . but this is something that you will soon get used to. A useful hint: always manipulate the mouse with its lengthwise axis perpendicular to the plane of the Macintosh screen. This is the only way of ensuring that up/down and left/right movements of the mouse on the table top will give rise — as might be expected — to analogous up/down and left/right displacements of the pointer on the Macintosh screen.

The Macintosh *keyboard*, attached to a flexible spiral cord that plugs into the front of the main machine, resembles a normal typewriter keyboard, except that there are four additional keys: <**Caps Lock**>, <**Option**>, <**Enter**> and <⌘> (the latter being the "Command" key). The various functions of these keys will be described as they are encountered in the following chapters.

To see how other peripheral equipment fits into the scene, we have to move around to the back of Macintosh, where there is a series of four black sockets, all of which bear descriptive icons molded into the plastic. (We also encounter the name of the machine, for the first time, stamped on its rear end!) Various accessory devices can be connected to Macintosh by means of these built-in hardware interfaces. The leftmost socket is for the mouse. The next one makes it possible to plug in an extra disk drive. The other two sockets are high-speed *serial ports* for peripheral equipment such as the *Imagewriter* printer and modems. Mass storage devices similar to Apple's *Profile* are being developed by several firms, and these hard disks will be connected to Macintosh through these serial ports.

Having seen what Macintosh looks like at a hardware level, let us now carry out a detailed examination — in the following four chapters — of the major functional and logical *concepts* upon which this new machine has been designed.

chapter 1
The Mouse Concept

Apple's promotional literature on Macintosh contains an intelligent rhetorical question on man/machine relationships: "*Since computers are so smart, wouldn't it make more sense to teach computers about people, instead of teaching people about computers?*"

The Macintosh mouse provides an example of how the machine has been "taught" to handle the typically human habit of *pointing* at things, which is probably an even more archaic and fundamental form of communication than language itself. Tourists in a Greek restaurant, for example, can order an entire meal by simply pointing at the dishes they want, without speaking a single word. So it's not surprising that machine designers should have invented methods for accepting this kind of input.

Various designers have actually taken the notion of pointing quite literally, by developing touch-sensitive screens or light pens. But the problem in such cases is that the user is likely to become rather tired at the end of an entire day spent holding up his/her arm to point at things on the screen. Other designers have thought that the ideal solution, instead of inventing techniques for pointing, would be to get the machines to understand spoken commands in a natural language such as English. But, even if that were to become a perfectly feasible method, it's strange to imagine the atmosphere in an office, say, with dozens of employees all conversing aloud with their machines!

The mouse provides an elegant solution to man/machine communication in the sense that it enables you to designate objects on the screen with a great deal of precision and with little physical effort. In fact, so many laudatory remarks can be made concerning the merits of this ingenious device that maybe, on the contrary, we should start out by indicating what appears to be the unique drawback of the mouse: namely, the fact that it is not easy to execute freehand sketching or writing with such a tool. We humans are accustomed to holding a pencil between our fingertips, and the manner in which you clasp the Macintosh mouse is hardly a natural position for wielding a writing or drawing instrument. To put it bluntly, using the mouse to execute a freehand drawing feels a little like trying to sign a check while wearing a boxing glove. But we shall see later on, when we get around to describing the *MacPaint* software product, that this deficiency of the mouse can be overcome in several ways.

In any case, the user is generally astonished by the ease and rapidity with which the mouse makes it possible to position the pointer at a precise spot on the screen. In other words, one has the unexpected impression, when using the mouse for the first time, that it's easier to dot one's "i"s than to trace the main part of the letter!

The mouse button

Since there is a single rectangular button on top of the mouse, it comes as somewhat of a surprise to discover how it can be used to carry out several different operations. There are no less than five different ways in which this button can be manipulated:

1 First of all, there is the trivial case in which you don't touch the button at all, even though you may be moving the mouse on the table. Usually, in such cases, the only result is that the pointer moves around on the screen, following the movements of the mouse on the table. It is nevertheless perfectly feasible that various events might take place on the screen, even though you never touch the mouse button. For example, the demonstration material for Macintosh includes a number of attractive maze games, and you only need to move the mouse, without ever touching the mouse button, in order to trace out your path through the maze.

2 Often you move the mouse until the pointer is located at a specific spot on the screen, and then you press the mouse button and release it almost instantly. This operation is referred to as *clicking*, and it is the conventional manner of informing Macintosh that you wish to select some particular object on the screen. For example, if you are interested in using the *MacPaint* software product, the first thing you must do is to move the mouse until the pointer is positioned on the *MacPaint* icon on the screen, and then you simply click the mouse button.

3 Sometimes you are expected to click the mouse button twice. This is called *double clicking*, and is generally used as a shortcut technique for doing something more rapidly than usual. For example, when you wish to rub out a drawing made by means of *MacPaint*, there are several ways of going about it, but the quickest technique is to simply double-click the eraser icon.

4 Often, instead of rapidly clicking the mouse button, you hold it pressed down for several seconds, while keeping the mouse immobile. This is a common technique for observing a menu, for example. In such a case, the menu appears on the screen as soon as the mouse button is pressed, and it disappears as soon as the button is released. To distinguish this mode of operation from clicking, we speak simply of *pressing* the mouse button.

5 The final variant of button manipulation, which is very important in the Macintosh context, is known as *dragging*, and it takes place in four consecutive

steps. First, you move the mouse so that the pointer is positioned at the desired place on the screen. Then you press the mouse button and keep it held down. Next, while still holding the button pressed down, you actually move the mouse, which generally causes various graphic events to take place on the screen. Finally, when the pointer has reached a certain position on the screen, you release the mouse button. There are two familiar examples of dragging. On the one hand, we have already mentioned that the bit-mapped screen makes it possible to actually drag images across the screen in real time. So, if you wish to throw away a particular document, you move the pointer to the icon that represents that document, you press the mouse button to select the document, then you hold the button down and move the mouse so that the icon is dragged towards the trash can, and you finally release the button to indicate that you want to deposit the document in the trash can. During the actual dragging operation, a flickering outline of the object being dragged moves across the screen with the pointer. The other familiar example of dragging concerns the selection of an option in a menu. The normal procedure is that you move the mouse and press the button to select a particular menu, which is immediately "pulled down" onto the screen. You then drag the pointer down over the menu until you reach the particular option you wish to select, and the actual selection is indicated by releasing the mouse button.

There is an interesting surprise in store for newcomers to the Macintosh mouse. You invariably discover with astonishment, after using the beast for no more than an hour or so, that your handling of the mouse and the mouse button has become almost instinctive. In other words, for the first hour, you might have to think consciously about the notions of clicking and dragging. But, as soon as you have acquired a feeling for these manipulations, you no longer even have to think about what you're doing, because your hand and eyes perform the necessary operations almost automatically.

Vocabulary

This is an appropriate moment to mention a subject that is quite significant in the Macintosh context: namely, the nature of the language that we use to talk about the machine.

In the previous section, precise terms such as "clicking", "pressing" and "dragging" were introduced in order to designate the various types of mouse manipulation. This is very typical of the Macintosh situation in the sense that it's a new kind of computer, and Apple has been obliged to invent an extensive vocabulary enabling us to talk about the machine in a clear and precise manner.

It is well worthwhile going to the trouble to learn this vocabulary off by heart and to use it consistently, because these terms reappear in all the Macintosh literature.

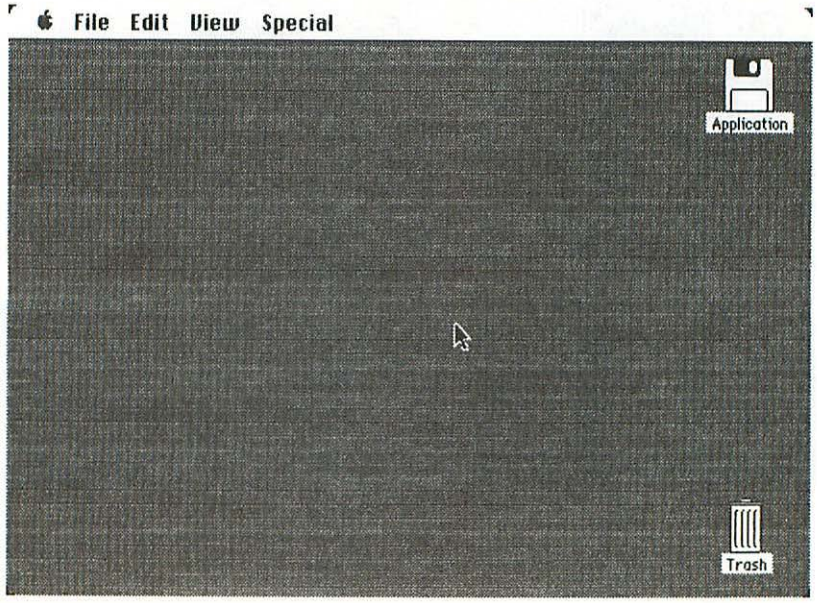
Here we encounter yet another major difference between Macintosh and the previous generation of microcomputers. Although Macintosh is being used to tackle many quite different types of applications, there is an undeniable resemblance between most of the programs you encounter, at least at a superficial level. They often look much alike in the sense that you invariably find the same kind of menus, presented in the same kind of graphic style.

This is not a chance happening. The main reason for this uniformity is the fact that almost all software developers are using the same basic routines that are stored in the *read-only memory* (ROM) of the machine. (See Appendix A for a fuller discussion of this subject.) Furthermore, Apple is encouraging software developers to respect various graphic conventions so that we can speak of a *unique and homogeneous interface* between Macintosh and the user. The advantages of this situation are obvious. Once you've learnt how to handle the machine in one application domain, you've already acquired sufficient skills to use Macintosh for any other application whatsoever.

Menus

Let us now take a closer look at the precise manner in which the mouse is used to deal with menus. In doing so, we shall be using the vocabulary that has already been presented, along with several new words.

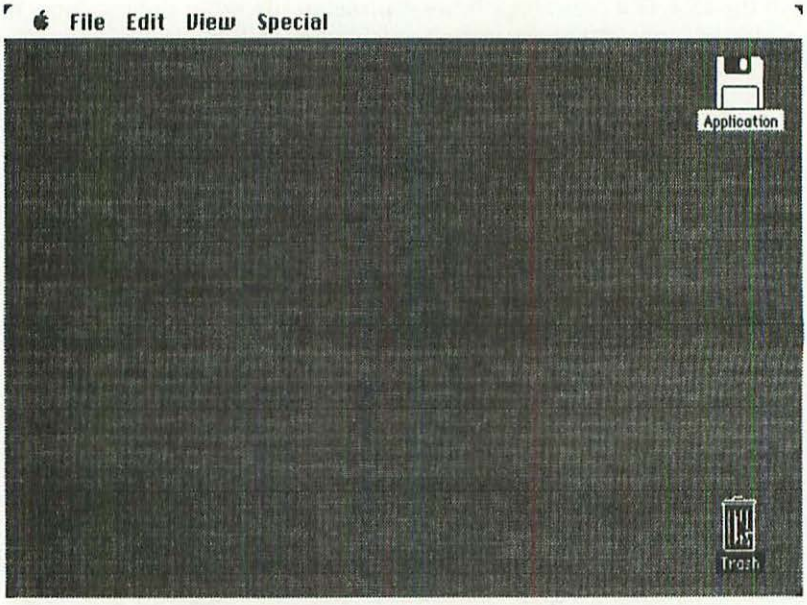
With the disk of a typical application program placed in the machine, the user might carry out one or two elementary clicking operations in order to obtain the following kind of image on the screen:



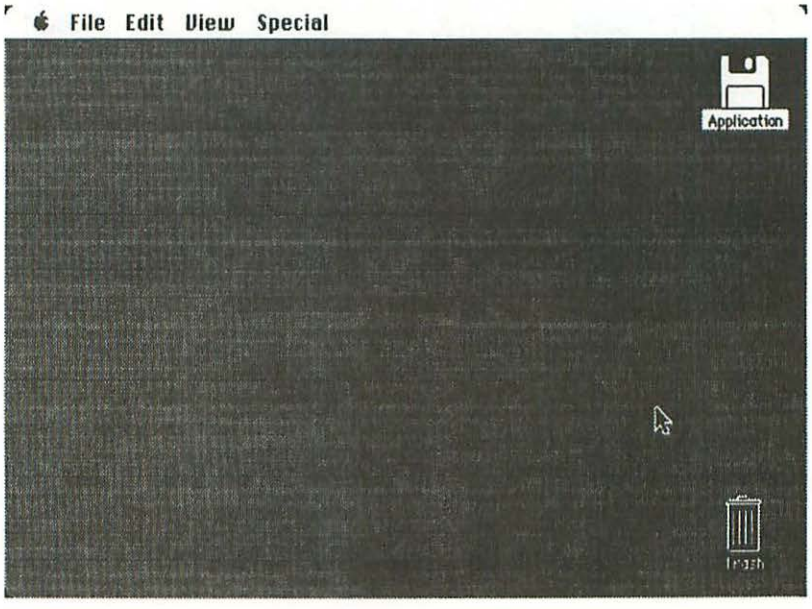
The narrow white section at the top of the screen, with the Apple image followed by four words, is the *menu bar*, whereas the main part of the screen, displayed in gray, is the so-called Macintosh *desktop*. There are two icons on the desktop: one that represents the disk itself, which happens to be called "Application" in the present example, and a second icon representing the trash can. As for the pointer, it is located for the moment in the middle of the screen.

We meet up, at this point, with the fundamental Macintosh concept of *selection*. To understand fully what this process is all about, you have to realize that most activities carried out by Macintosh take place in two steps. First, you select the object or the information you want to work with, and then you choose a particular *command* from one of the menus, to indicate to the computer what you want to do with the selected item.

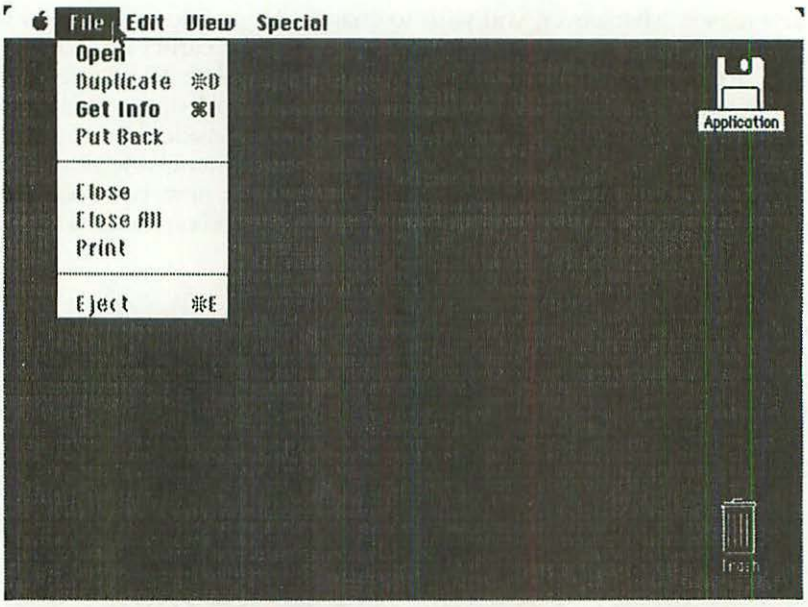
For the moment, both objects on the desktop are represented as white icons, which means that neither object has yet been selected. To select one of the two objects, use the mouse to position the pointer on the desired icon, then click the mouse button. Let's suppose that we want to take a closer look at the trash can. We start out by selecting the trash icon. This is done by simply positioning the pointer at that spot on the screen and clicking the mouse button. Notice that the trash icon is immediately highlighted, in the sense that it changes color from white to black, as shown in the following illustration:



If, for any reason whatsoever, you wish to change the position of an icon on the desktop, use the dragging technique that was mentioned earlier on. For example, you might decide to move the trash can further up to the left. All you need to do is to select the trash icon, keep the mouse button pressed down, and move the mouse so that the flickering rectangle with the pointer inside (see the following illustration) is positioned at the new spot. At that moment, release the mouse button, and the trash icon will jump magically to its new position. But let's suppose, for the moment, that the position of the trash icon on the desktop has not been changed.

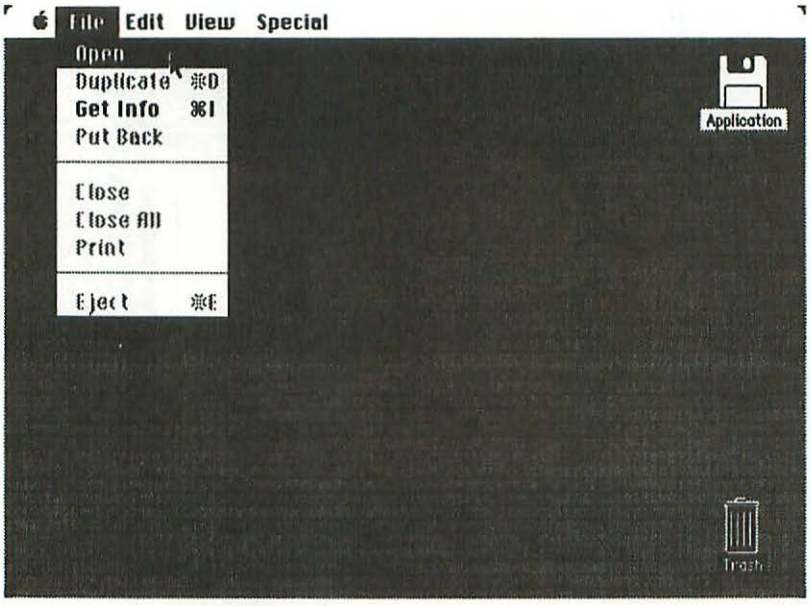


The next important step, if we wish to find out anything more about the trash can, is to pull down a menu. In most cases, it is the **File menu** that is used first of all. To pull it down, move the pointer to the word "File" in the menu bar and press the mouse button. The word "File" is immediately highlighted by being displayed in white letters on a black background. As long as you keep the mouse immobile, with the button pressed down, the screen appears as follows:



Notice that the File menu is composed of eight *commands*. Two of them - **Open** and **Get Info** - appear in boldface letters, indicating that these commands are valid options for the selected object: namely, the trash can. In other words, we have the right to either open the trash can, to see its contents, or to simply get information on this object. But the other six commands, displayed in dimmed letters, are not valid options (at least, not for the moment) as far as the trash can is concerned.

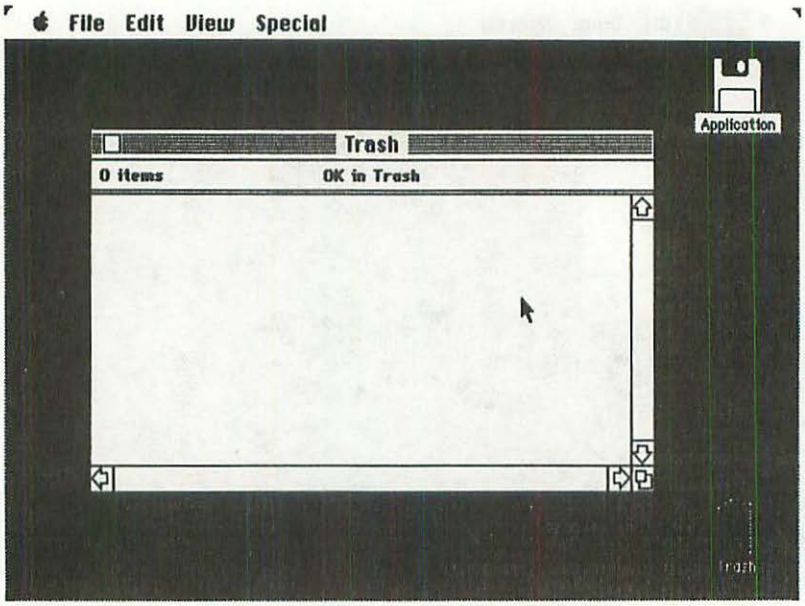
To choose a particular command, you keep the mouse button pressed down and you drag the pointer down over the lines of the menu. Each time the pointer is positioned at a valid option, the line is highlighted by being displayed in white letters on a black background. And, to indicate your choice of a command, you simply stop dragging and release the mouse button. In the illustration that follows, we can suppose that the Open command is about to be chosen:



Notice that three of the lines in the File menu have strange-looking symbols to the right. Consider the Get Info line, for example. The square symbol with loops at each corner is the <⌘> (Command) key, which is located just to the left of the spacebar on the Macintosh keyboard. The appearance of this symbol after the expression "Get Info" is simply a reminder that there is an alternative method of choosing this command. Instead of pulling down the File menu and dragging the pointer down to the Get Info line, you can hit the <⌘> key and then the letter "I". In other words, it's a shortcut that allows you to perform an operation from the keyboard, without using the mouse.

At the end of the dimmed lines for the **Duplicate** and **Eject** commands, you have what appears to be a different symbol, that looks like a Maltese cross. In fact, this is *not* another symbol; it's simply the dimmed version of a reference to this same <⌘> command key.

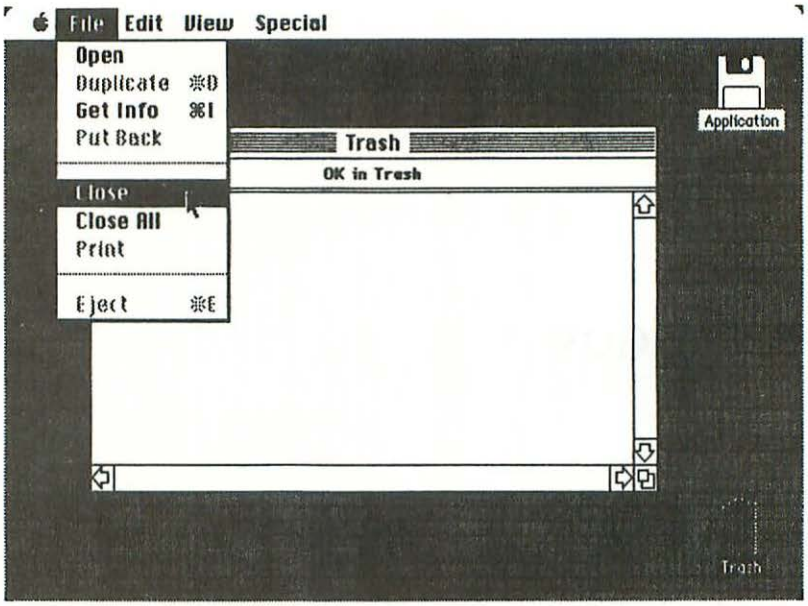
As soon as the Open command is chosen for the trash can, the following image appears on the screen:



In the middle of the desktop, the large white rectangle provides us with an example of a *window*, which is a fundamental Macintosh concept that will be dealt with in the following chapter. Windows normally display all kinds of data. However, since the trash can is apparently empty for the moment, this particular window has nothing to show us. The window in fact tells us this quite explicitly: the number of items in trash is indicated as zero, and the quantity of disk space devoted to trash is indicated as OK, meaning zero kilobytes. (Don't confuse this "zero kilobytes" indication with the familiar **OK** endorsement . . . which is also used quite frequently in man/Macintosh communications!)

Notice that the trash icon down in the righthand corner of the screen has once again changed its appearance. In its present state, it is described as being *hollow*, which serves to indicate that the icon has in fact been opened into a window.

For the moment, let us close the window that we just opened. This can be done by pulling down the File menu once again, and dragging the pointer down to the **Close** command, as in the following illustration:



Notice that the pulled-down File menu actually covers part of the window on the desktop. If ever you suddenly decided at this point, for one reason or another, *not* to close the window, then you would simply keep the mouse button pressed down, move the pointer back up to the word "File" in the menu bar, and release the button. The menu would then disappear, and the entire window would once again be visible on the desktop.

But we have decided to close this window. So, leave the pointer positioned at the Close command — as indicated in the preceding illustration — and release the button. Both the menu and the window disappear instantly from the desktop, and the trash icon — which remains selected - reassumes its initial highlighted appearance.

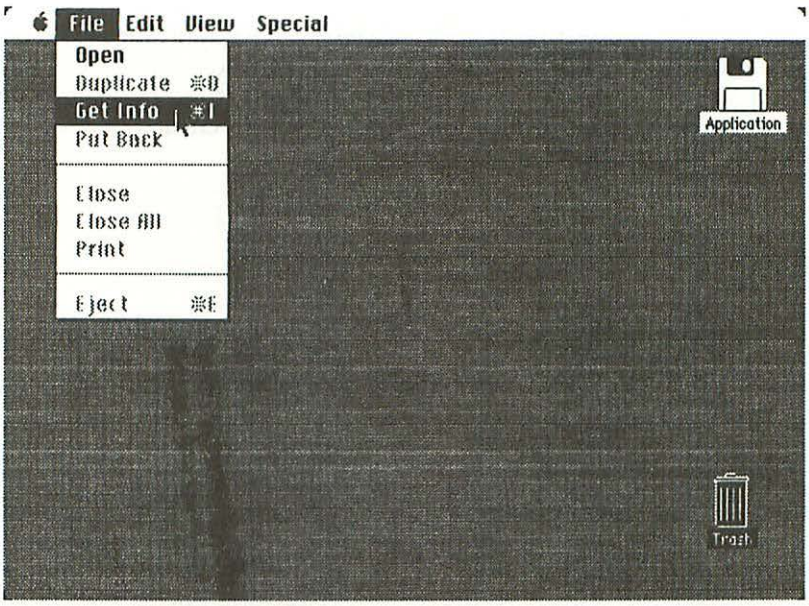
chapter 2

The Window Concept

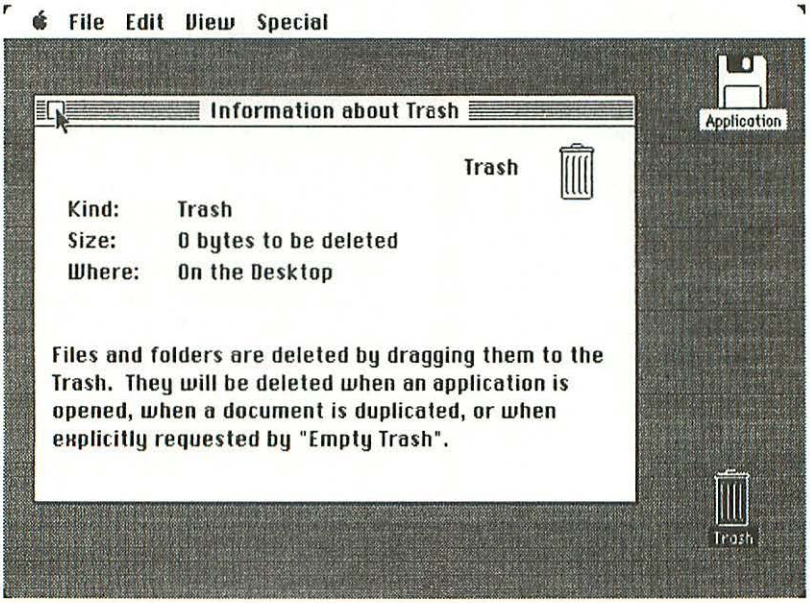
In our everyday existence we encounter all kinds of windows, which don't necessarily have much in common with one another, apart from the fact that they're transparent and generally rectangular: shop windows, train windows, envelope windows. . . . We must therefore determine the particular sense of the term "window" in the Macintosh context.

Maybe the best starting-point is to ask the question: What sort of things can you actually see in a Macintosh window? In this regard, the example of a window that we discovered at the end of the previous chapter was not very helpful, since there was nothing whatsoever to see!

In fact, one of the commonest things that you can find in Macintosh windows is ordinary text. To see an example of this, let us select the trash icon, as before, and choose another command: **Get Info**. As its name suggests, the role of this command is to provide us with information concerning the selected icon. So the File menu is pulled down in the usual manner and the command is chosen as shown in the following illustration:



The window that appears on the screen (see following illustration) bears the title "Information about Trash", and it includes a brief textual description, together with a graphic reproduction of the trash icon. We learn, in particular, the conditions under which objects are deleted from the disk by being dragged to the trash can.



Earlier on, we saw that it is easy to close a window by pulling down the File menu and choosing the Close command. But there is an even quicker way of doing it. You simply move the pointer up to the little square in the top lefthand corner of the window, called the *close box*, and you click it with the mouse button. In our example, as before, the window disappears but the highlighted trash icon still remains selected.

Summary

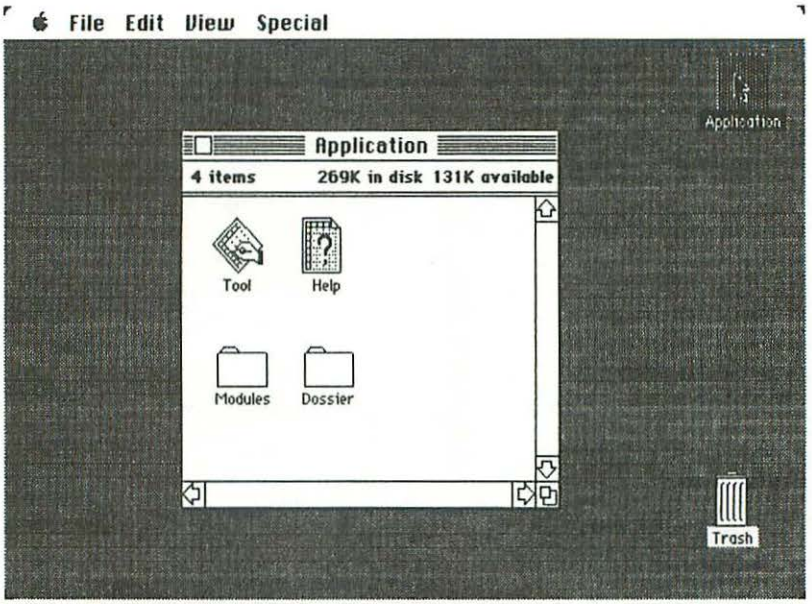
Besides text, what other things can we look at through Macintosh windows? This question provides us with an excellent pretext for summarizing the various Macintosh "things", displayed on the screen, that we have already encountered. They are exactly six in number:

1 The *desktop* is a large gray space, occupying almost the entire screen, that can carry graphic elements that may be either images (for example, the trash can) or text (the word "Trash"). Windows, too, can be opened on the desktop.
2 An *icon* is a small graphic symbol, with an attached caption, that stands for some kind of Macintosh object. So far, we have seen icons representing only two objects: the disk named "Application" and the trash can.
3 The *menu bar* is a narrow horizontal zone at the top of the screen that contains the image of an apple together with the names of several menus.
4 A *menu* is a block of optional commands that can be "pulled down" from the menu bar onto the desktop.
5 The *pointer* is a tiny graphic symbol (often an arrow) that can be moved — by means of the mouse — all over the desktop, the displayed icons, the menu bar, or pulled-down menus.
6 A *window* is a large rectangle (usually containing information) that appears on the desktop when we select an icon and then choose either the Open or Get Info commands of the File menu. The window disappears when we close it, either by using the Close command of this same File menu, or by clicking the window's close box.

We are concerned with the question of the kind of "things" — apart from text —that can be found in Macintosh windows. One answer is that many windows are simply filled with a variety of icons. So, maybe the best step now is to take a closer look at this fundamental Macintosh notion. . . .

Icons

Let us start with an example. Using the same disk as in the previous examples, suppose that we select and open the disk-shaped icon, in the upper righthand corner, labeled "Application". Here is the window that appears on the desktop:



As expected, the icon labeled "Application" is now hollow, to indicate that it has been opened into a window. As for the window itself, it encloses four new icons. In the top row, the first icon designates an application program named "Tool", and the second one indicates the existence of "Help" material (no doubt concerning the just-mentioned tool). In the bottom row, the other two icons represent *folders* for storing documents. The folder on the left, bearing the caption "Modules", is reserved for various components of Macintosh system software, whereas the one on the right, an empty folder labeled "Dossiers", is available to store documents created by the "Tool" program. (When you purchase a Macintosh application program, the chances are that, instead of our labels "Modules" and "Dossier", the new disk will carry the names "System folder" and "Empty folder". The truth is that you can change these names of icons, in an instant, in any way that pleases you.)

It has already been pointed out that every Macintosh icon stands for some kind of object. Now, what exactly is the object represented by the initial icon labeled "Application"? As you might expect, since the icon looks like a disk, it represents the disk itself that is currently inside the Macintosh disk drive. So, in opening this icon into a window, what we have actually done is to ask Macintosh to tell us what is stored on this disk. In other words, the window provides us with the *directory* indicating the contents of this disk, and we learn therefore that the disk labeled "Application" contains, at present, exactly four objects (or items), labeled "Tool", "Help", "Modules" and "Dossier".

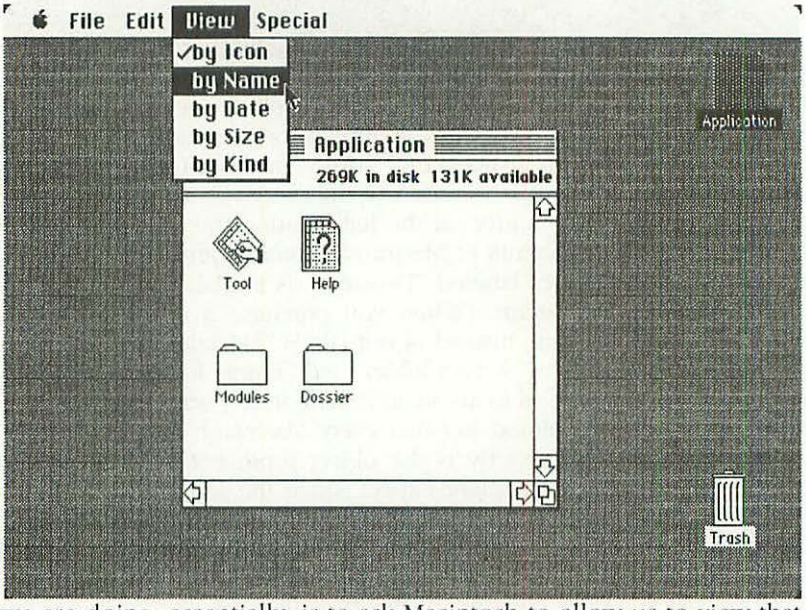
There is something that might nevertheless appear strange to many newcomers to Macintosh: namely, the incongruous nature of all the so-called "objects" that can be represented by icons. After all, the icon labeled "Application" represents a Macintosh disk, the "Trash" icon represents some sort of document

removal device, the "Tool" icon inside the window represents an application program, the "Dossier" icon represents an empty folder, etc. How can we even *define* what an icon is, if it can represent so many different types of objects?

The answer to this question will be obvious to anybody who is familiar with the rudiments of microcomputing. All these so-called "objects", represented by icons, correspond simply to various regions of the disk that is currently inside the Macintosh drive. Information stored on a disk — just like information inside the main memory of a computer — can represent all manner of things: programs, data sets to be used by these programs, results created by these programs, etc. So it is not surprising that the icons corresponding to these various blocks of disk-based information give the impression that they represent a certain number of different types of "objects".

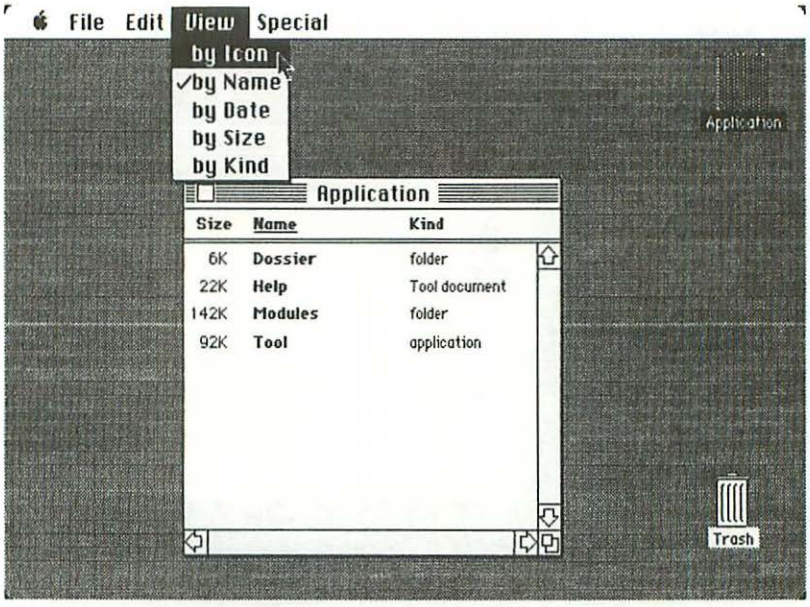
Any particular block of information on a disk, no matter what sort of information it is, can be thought of as a *file*. In other words, we can define an icon, most simply, as a graphic symbol that provides access to a specific disk file.

This notion of interpreting icons as symbols for disk files can best be appreciated if we make use of the **View menu** and the **by Name** command, as indicated in the following illustration:



What we are doing, essentially, is to ask Macintosh to allow us to view that same window with *names* instead of *icons* for the various disk files. The result (see following illustration) confirms the idea that each of the four objects to which the icons correspond is in fact a *file*, for this window even informs us of the precise size of each file.

In the same View menu, the **by Icon** command enables us to return to the original style of window presentation.

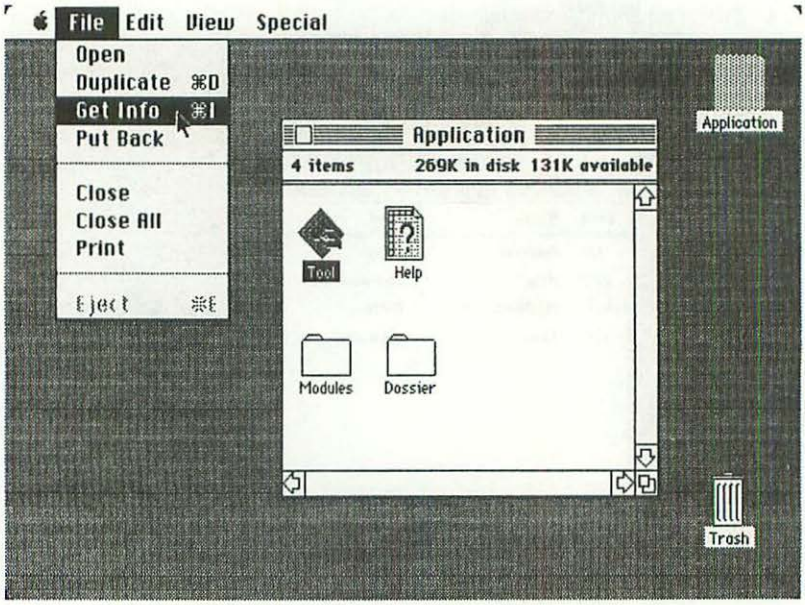


One final question: What, then, does the disk-shaped "Application" icon actually represent? The obvious answer is that it corresponds to a small *file directory* stored on the disk itself.

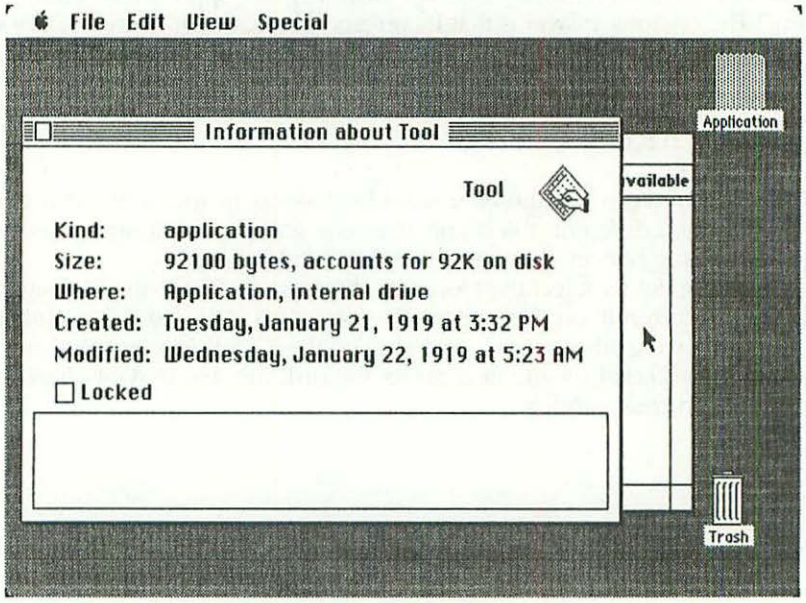
Multiple windows

The Macintosh desktop resembles a real-life desktop in the sense that you can place several quite different objects on it, at one and the same time, even if they end up overlapping one another and forming an untidy heap.

For example, let us select the icon with the caption "Tool" that belongs to the window that is currently on the screen. Then we shall pull down the familiar File menu (see following illustration), and choose the **Get Info** command. In other words, we wish to obtain some facts about the disk file that is symbolized by the first icon in the current window.



As expected, we obtain the following window entitled "Information about Tool", which is similar to the "Information about Trash" window that we saw earlier on:



Don't bother about the information itself in this window. (One has the impression that the person who supplied the dates was either very much in advance of his/her time, or else a liar! The truth of the matter is that the programmer who developed the prototype disk that we are using for our examples apparently did not go to the trouble of correctly setting the Macintosh clock before starting to work on the machine.)

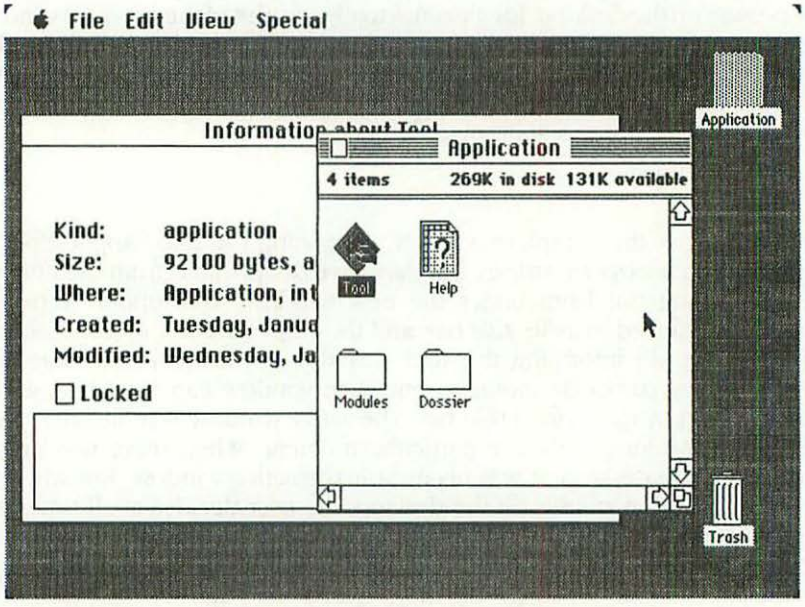
The thing that interests us most in this illustration is the fact that the new window which we have just opened seems to have been placed *on top of* the window that was already present on the desktop, for we can actually see a bit of the previous window — containing part of the word "available" — sticking out from under the righthand edge of the new window. But, if you look very closely at this piece of the old window, and compare it with the corresponding part of the complete window that we saw earlier on, you can discover three significant differences:

1 The top part of the complete window, containing the title "Application", was adorned with horizontal stripes, but they have disappeared from the little piece that is sticking out from under the new window. This upper section of a window is referred to as its *title bar*, and the stripes are not merely decorative. They are a code informing the user that this is the *active window* on the screen. At any particular moment, only one window can be active, and it is distinguished by the striped title bar. The active window is in fact the one that the user is working with at a particular moment. When there was only one window on the desktop, it was necessarily the active window. But when there are two or more windows on the desktop, the user decides at all times which window is to be active.
2 In the lower-right corner of the complete window, there was a graphic symbol composed of a pair of tiny overlapping squares, and this has disappeared from the piece that is sticking out from under the new window. This symbol is the *size box* which, in the case of an active window, can be dragged by the mouse to change the height and width of the window.
3 To the right of the complete window, and underneath it, there were columns with arrows at each end. In the piece sticking out from under the new window, the columns have remained but the arrows are no longer present. These columns equipped with arrows are called *scroll bars*, since they are used (in a way that will be explained in detail later on) for scrolling through documents that are so big that it is not possible to see their entire contents through the window at one instant.

Notice that the new window called "Information about Tool" has a title bar with stripes, indicating that it is the active window on the desktop. On the other hand, it is a special case of a window that has neither a size box nor scroll bars.

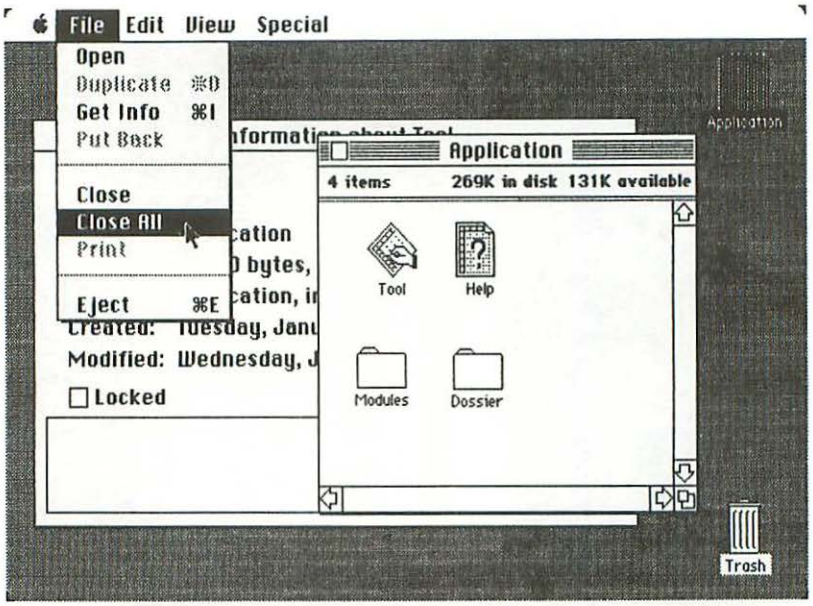
How do we get back to the initial situation, with "Application" as the active window? There are two techniques. If you are no longer interested in the window

entitled "Information about Tool", then you simply close it by clicking its close box, and the complete "Application" window reappears alone on the desktop, as before. On the other hand, if you wish to keep the big window on the desktop, in an inactive state, then you position the pointer at any location whatsoever on the piece of the "Application" window sticking out from under the big window, and you click the mouse button. The result is shown in the following illustration:



The "Application" window now overlaps the other one on the desktop, and the stripes indicate that it has once more become the active window. The size box and the two scroll bars are also back in place. Notice too that there are no longer any stripes in the title bar of the window entitled "Information about Tool".

Let us now close both these windows, in order to move on to another neighboring subject. It is interesting to note that the close box has disappeared from the upper-left corner of the window entitled "Information about Tool". In other words, an inactive window cannot be closed in this manner. The simplest technique to close both windows is to pull down the familiar File menu and choose the **Close All** command, as indicated in the following illustration:

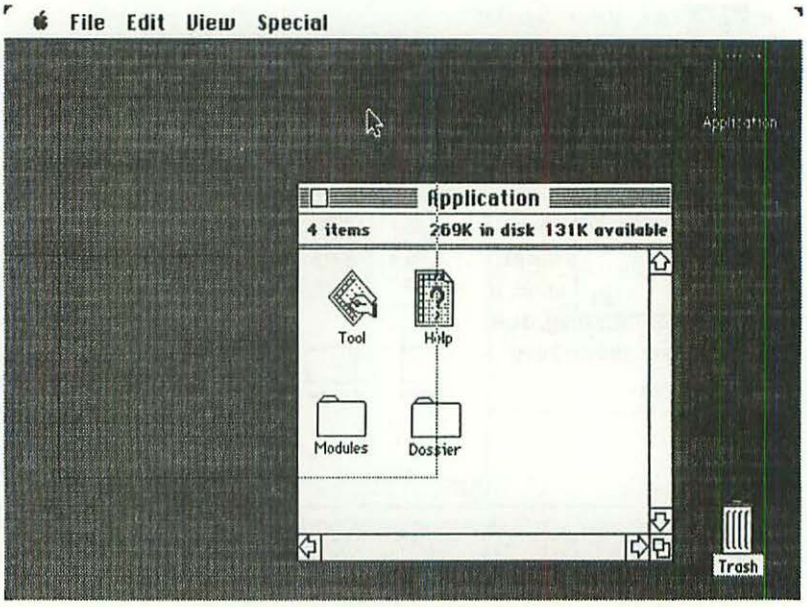


Once you release the mouse button, with the pointer in this position, both windows will disappear instantly from the desktop.

Window manipulations

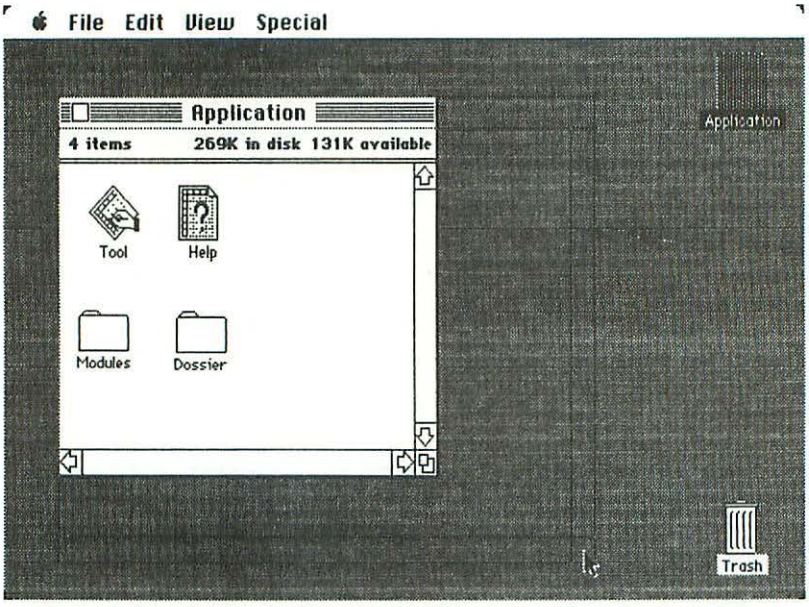
Let us put the "Application" window back on the desktop. Up until now we have done this by selecting the disk-shaped icon in the upper-right corner and choosing the Open command in the File menu. But there is a quicker technique for opening this icon: you simply double-click it.

Once the "Application" window is present on the desktop, there is an elementary manipulation that may be carried out on it: *window dragging*. You simply position the pointer anywhere in the title bar of the window (except for the close box), then you hold down the mouse button and drag the window wherever you like on the desktop. A flickering outline of the window accompanies the pointer during the dragging process (see following illustration). As soon as you release the mouse button, the window will reappear in its new position.

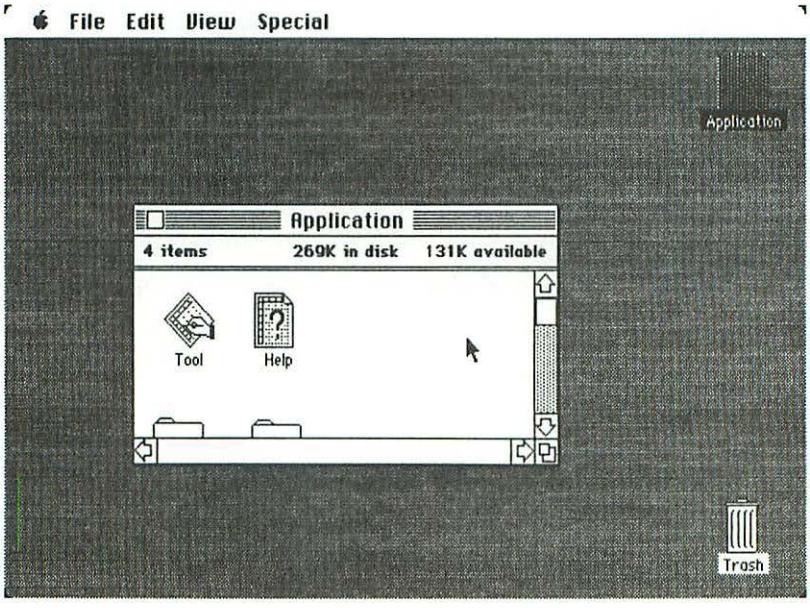


Real windows in the everyday world have two complementary functions: they *conceal* no less than they *reveal*. This, after all, is the essence of a window. People do not generally live in glass houses, we do not travel in plexiglass automobiles, nor do we mail off letters in clear plastic envelopes. The transparent window pane is inevitably set into some kind of opaque "wall" that prevents us, at any single instant in time, from viewing the entire scene on the other side.

Macintosh windows behave in much the same way. . . but they possess several magic powers that could hardly exist in the real world. First of all, it is an elementary task to *change the size* of a Macintosh window, simply by positioning the pointer in the size box, in the lower-right corner of the window, and dragging it over the desktop. A flickering outline of the enlarged or reduced window accompanies the pointer during the dragging process (see following illustration). As soon as you release the mouse button, the window will reappear on the desktop with its new proportions.



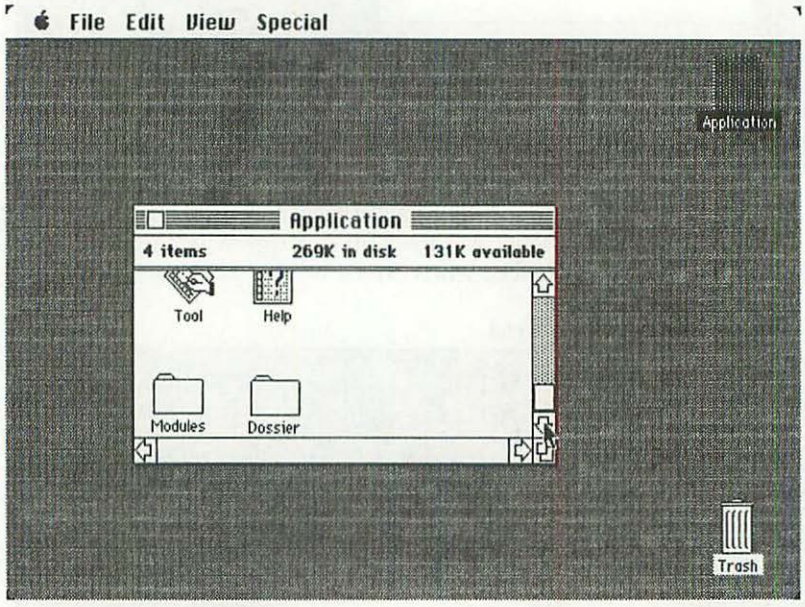
Let us suppose now that you use the size box to reduce the proportions of the window to such an extent that, as shown in the following illustration, it is no longer possible to view all the four icons on the other side of the window:



The *width* of the window is still sufficient to display the contents, but its reduced *height* prevents us from viewing the two folders down the bottom. This problem leads us into the subject of *scrolling*, which is important in the Macintosh context.

Examining the reduced window, we notice that something new has suddenly appeared on the righthand side. The *vertical scroll bar* used to be practically blank, but now it is filled with the following three elements:

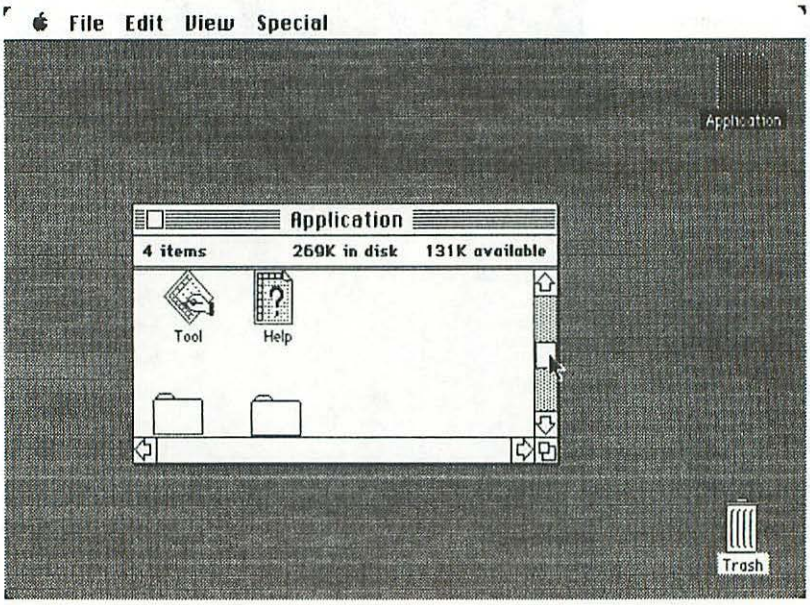
1 *Arrows*: Whenever the mouse button is used to click one or other of these symbols, the contents of the window change. When the upwards-pointing arrow is clicked, the displayed information scrolls line by line towards the top of the document; and when the downwards-pointing arrow is clicked (see following illustration), the displayed information scrolls in the opposite direction. Since there are only two lines in our example, these arrows make it possible to view either the top or the bottom half of the scene.



2 *Gray area*: Clicking in this zone of the scroll bar advances the displayed information, to the extent of one full window, in the direction indicated by the adjacent arrow.

3 *Scroll box*: This white square in the scroll bar works like an elevator in a building. You can drag it, with the mouse, to any point in the gray area, and the corresponding zone of the document will be displayed in the window. For example, if you drag the scroll box to the middle of the scroll bar, then it's the

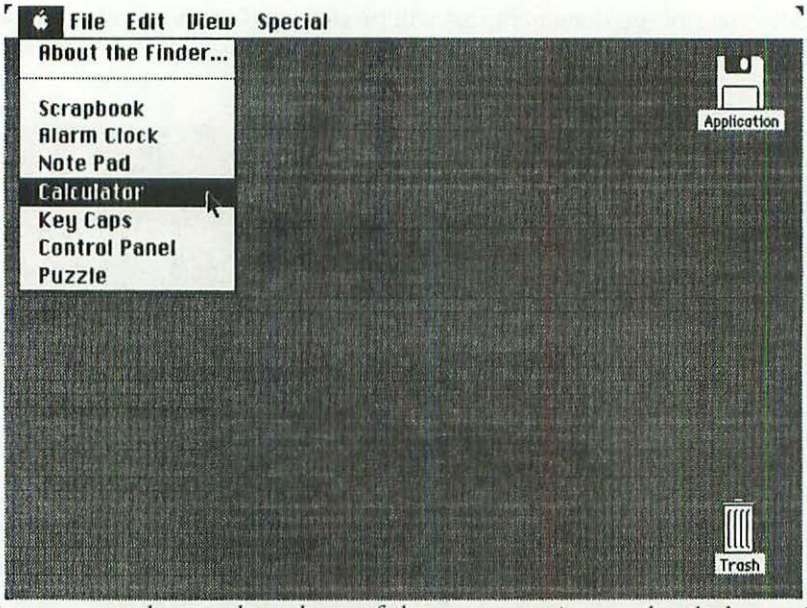
middle zone of the document that will be displayed in the window, as shown in the following illustration:



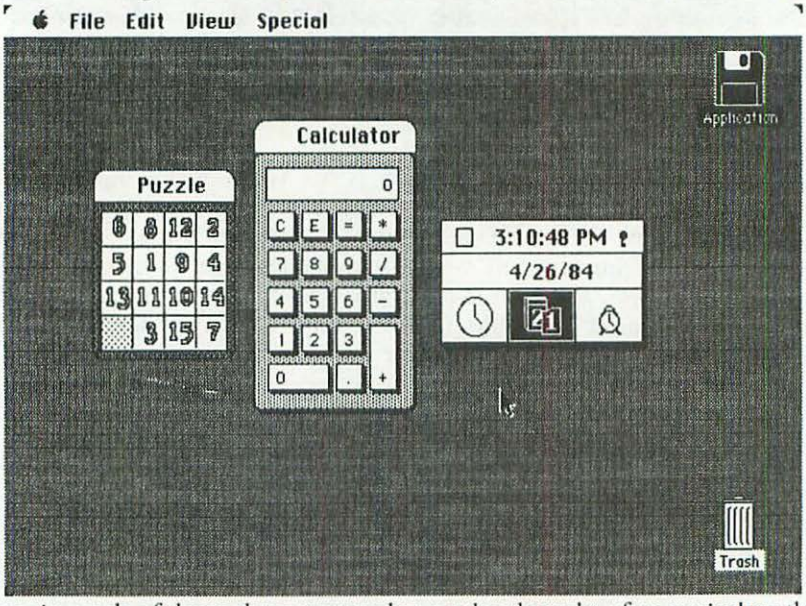
Admittedly, all this talk about scrolling is not very convincing when we are dealing — as is the case at present — with a document composed of merely four icons. But we have already suggested that the "things" that might be displayed in a Macintosh window can be as voluminous as the contents of any disk file whatsoever, and so it is imperative that this powerful scrolling device should exist, in order to enable users to "wander through" lengthy Macintosh documents in the most efficient manner.

Desk accessories

In the upper-left corner of the screen, in the menu bar, there is an image of an apple. This is actually yet another menu that can be pulled down (see following illustration) in exactly the same manner as the File menu. To choose one of these so-called *desk accessories*, you simply drag the pointer down to the name of the desired accessory, and you release the mouse button.



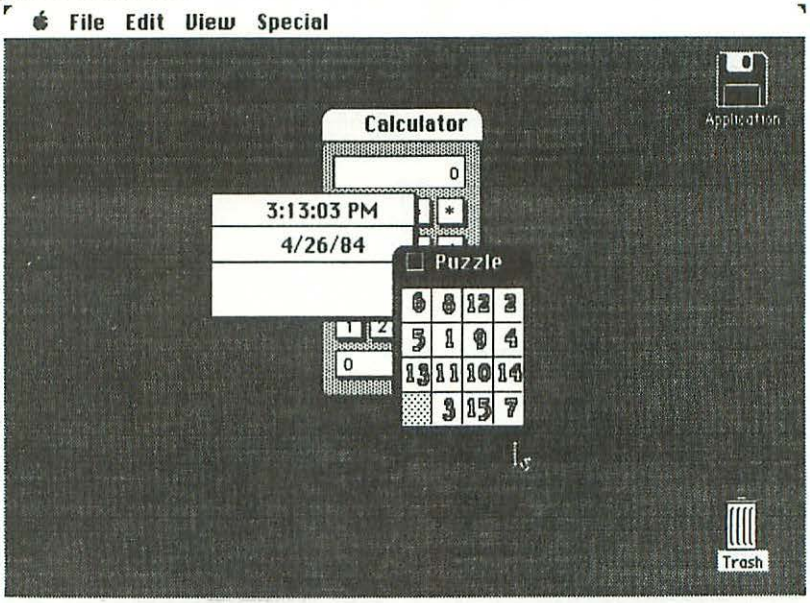
For the moment, let us place three of these accessories on the desktop, side by side (see following illustration): the **Alarm Clock**, the **Calculator** and the **Puzzle**.



Here again, each of these three rectangles can be thought of as a window that has been opened on the desktop. Besides, we know that the alarm clock is the active window, because it is the only one with a close box in the upper-left corner.

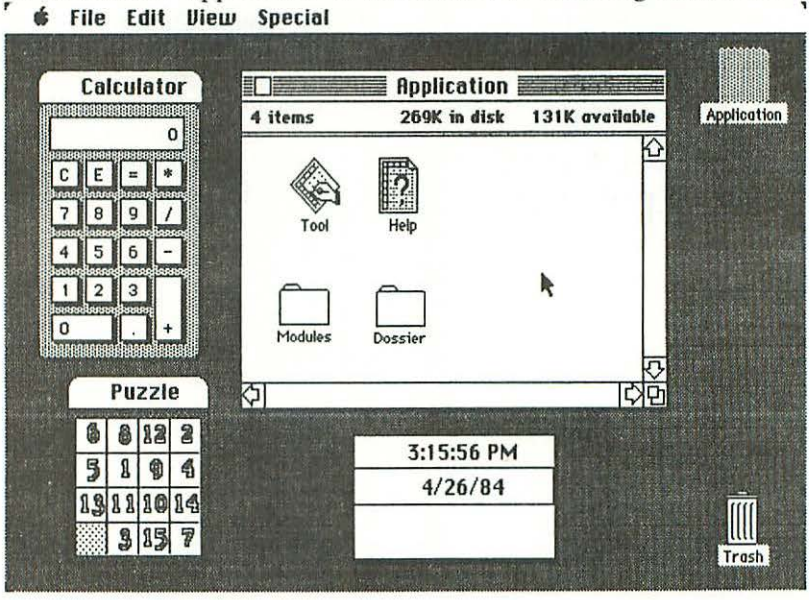
Although these windows have neither scroll bars nor size boxes, they do have

title bars, which means that they can be dragged around the desktop just like any ordinary window. In the following illustration, the three windows have been piled on top of one another:

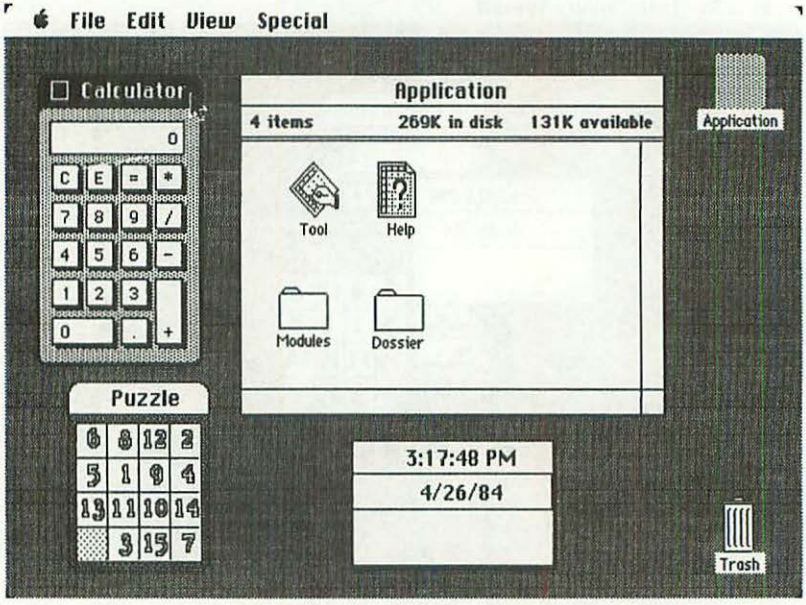


The puzzle window now has a highlighted title bar and a close box, which confirms that it is the active window.

The three accessories can be dragged to the edge of the desktop to make room for the familiar "Application" window, as in the following illustration:

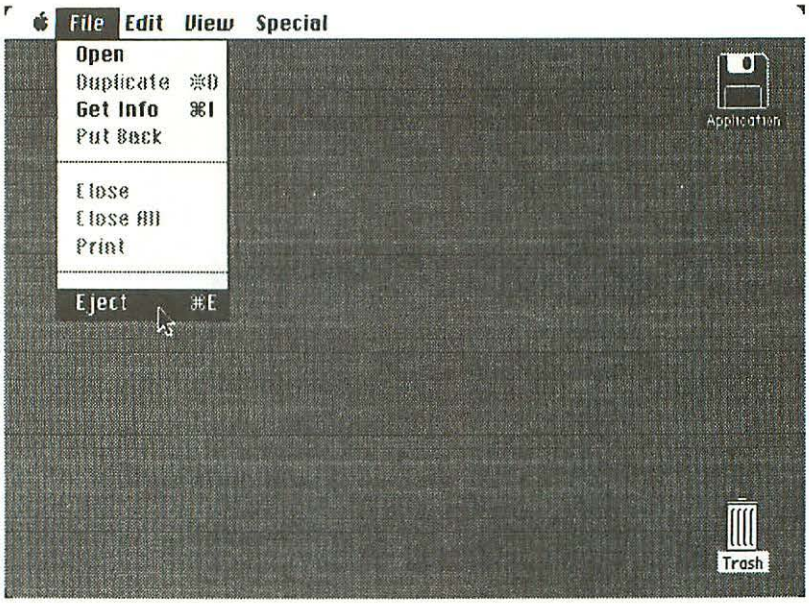


We see that the "Application" window is active, because it has both a close box and stripes in the title bar. But nothing prevents us from clicking the calculator, for example, which produces the following result:



Notice that the close box and the stripes have disappeared from the title bar of the "Application" window, whereas the title bar of the calculator is now highlighted, and it contains a close box.

Let us finally close all these four windows by means of the Close All command in the File menu, that we used earlier on. Having done this, we can make sure that the "Application" disk-shaped icon is selected, then drag the pointer down to the last command in the File menu, **Eject**, as shown in the following illustration:



When we release the mouse button, the "Application" disk will be physically ejected from the Macintosh disk drive.

chapter 3

The Cut & Paste Concept

The expression "cut & paste" is used in the press and publishing world to designate an everyday activity, carried out with scissors and glue, that consists of taking fragments of printed text and reassembling them, possibly with illustrations, to form pages. Writers, too, often work in a similar fashion when they are preparing articles or manuscripts.

The cut & paste technique might be described in very general terms as a three-step process:

1 Using scissors, you cut out the fragment — text or image — that interests you, and set it aside. Often, the sheet of paper that has just been cut into has to be glued back together again.
2 You locate the place to which this fragment must be moved. Sometimes this might mean that you have to use scissors once again to make a "hole", as it were, in an existing page.
3 Using glue, you fix the fragment in its new position.

Fortunately, the Macintosh implementation of the cut & paste technique is not nearly so messy, because the scissors and the glue have been replaced by two simple commands: **Cut** and **Paste**.

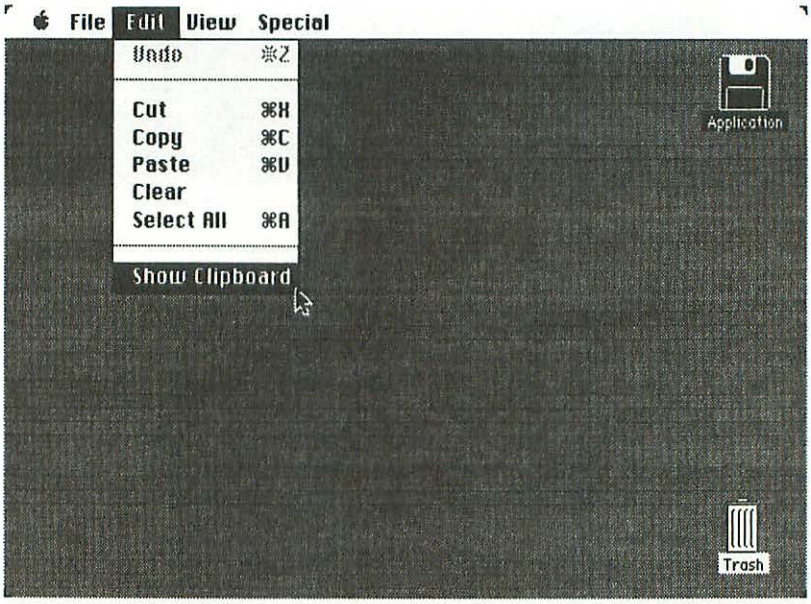
The Clipboard

Let us deal first of all with the *cut* aspect of the cut & paste technique. There are three problems that must be solved at this level:

1 We have to let Macintosh know exactly what it is that we want to cut. This action will be referred to as *selection*: the same term that has been used many times already with much the same meaning. We shall see that several different methods exist for the selection of the material to be cut.

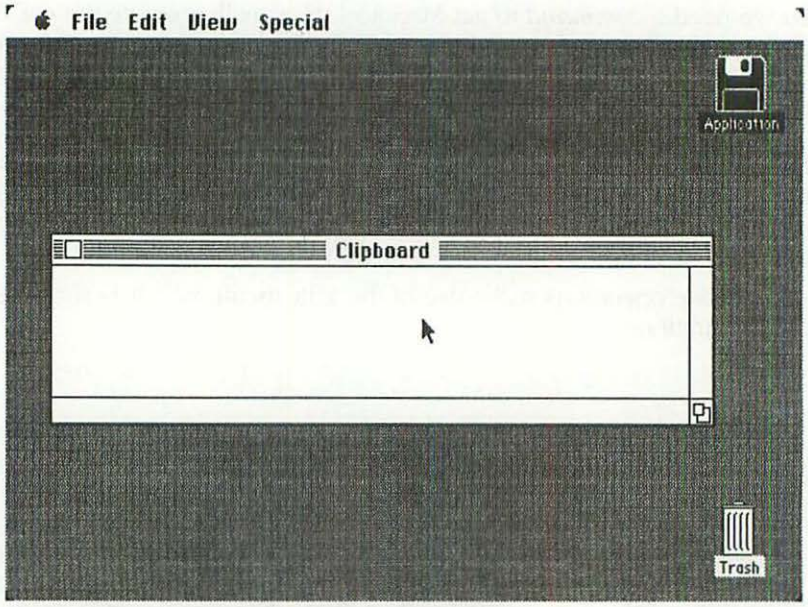
2 Next we need a *command* to get Macintosh to actually execute the cut. There are two possibilities at this level: **Cut** or **Copy**. In the first case, the cut material really disappears from the place where the cutting occurred. In the second case, the material is copied rather than cut.
3 Finally, we need a "container" to hold the cut material up until such time as we decide what to do with it. This is the role of the Macintosh **Clipboard**, which is one of the fundamental conceptual components of the computer. Anything that is cut or copied finds its way immediately and automatically to this Clipboard, and it stays there up until another cut or copy command is issued.

Most cut & paste operations make use of the **Edit menu**, which is shown in the following illustration:



There are seven commands in this menu, five of which can be issued by holding down the <⌘> command key and typing a letter.

When we choose the final command in the menu, **Show Clipboard**, the "Clipboard" window appears on the desktop, as shown in the following illustration:



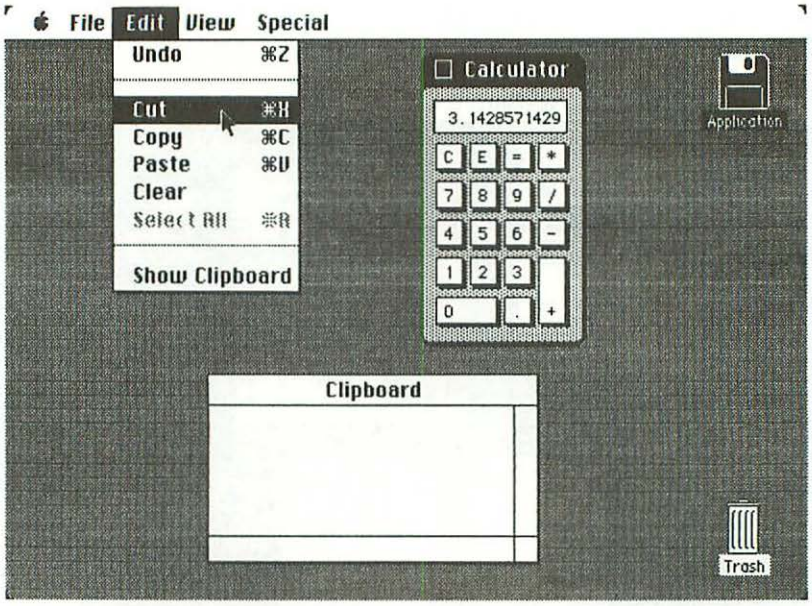
The Clipboard — which is empty in the present example — might be described as a rather "ordinary" Macintosh window. It has a conventional title bar and close box, together with a size box in the lower-right corner, but no scroll bars.

We shall see later on — and this is very important in the overall Macintosh context — that the Clipboard is capable of retaining not only text but images.

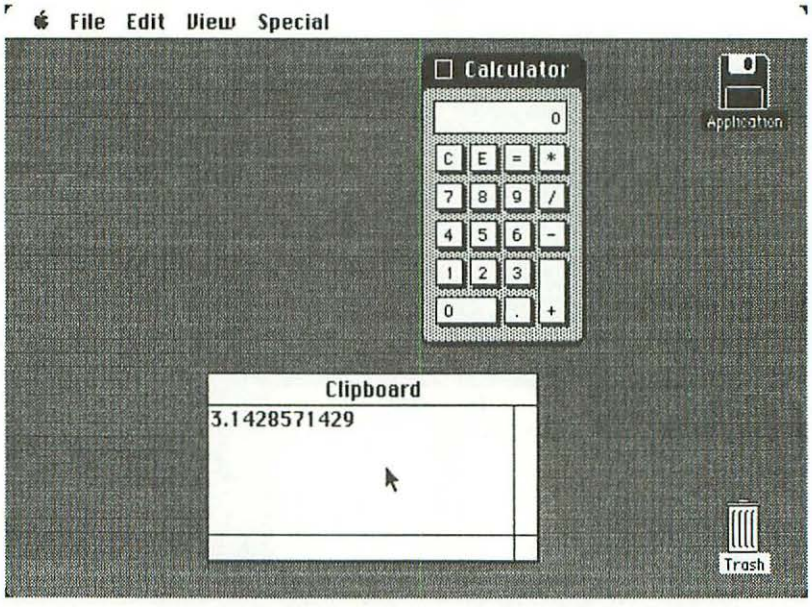
In order to see how the Clipboard works, what we need now is some kind of a document that we might "cut into". By way of an example, let us take the simplest source of information that can possibly be imagined in the case of Macintosh: the calculator.

First, we'll use the size box to reduce the dimensions of the Clipboard window, to obtain more room on the desktop. Then, let us imagine that we click the number 22 on the calculator keys, followed by the division symbol and the number 7. This gives us a result, displayed on the tiny calculator "screen", that we shall henceforth refer to as *pi* (see following illustration).

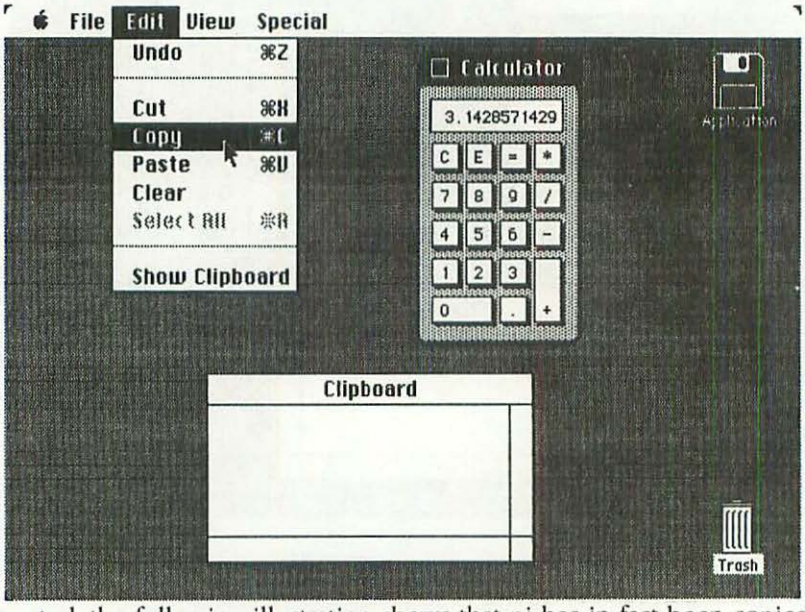
The title bar of the calculator is highlighted, which confirms that this window is active. Let us therefore pull down the Edit menu and choose the Cut command.



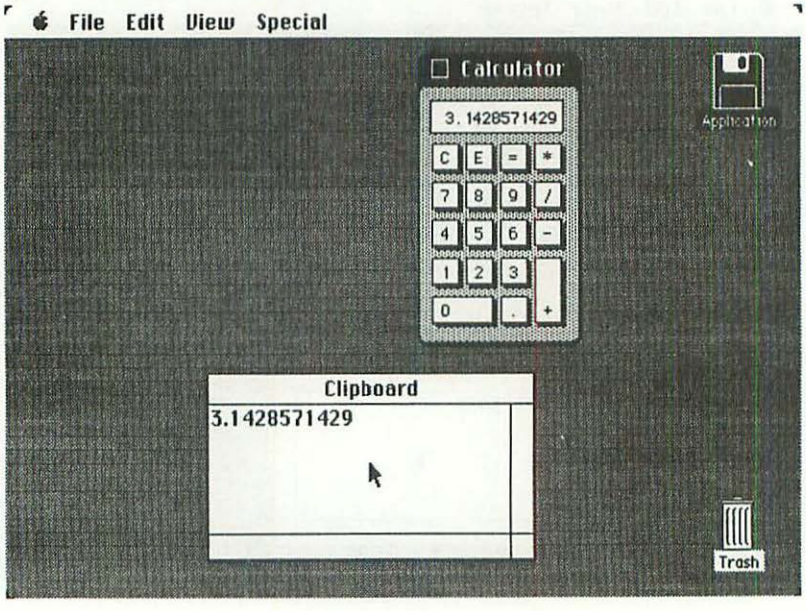
The following illustration proves that pi has been cut out of the calculator window (leaving a zero in its place) and transferred to the Clipboard:



Let us now repeat this little experiment (see following illustration), starting with a blank Clipboard and the value of pi on the calculator, but using Copy instead of Cut.



As expected, the following illustration shows that pi has in fact been copied onto the Clipboard, but it still remains in the calculator window:

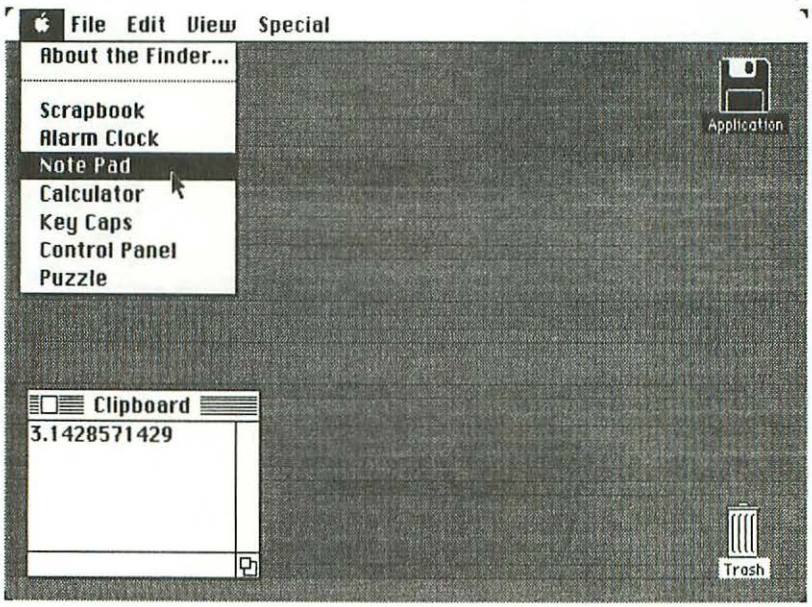


We can now turn our attention to the *paste* aspect of the cut & paste technique. Here again, there are three problems that must be solved at this level:

1 Where exactly is the pasting going to take place? This involves the familiar concept of the selection of a specific document that is to receive the cut-out information. We also need to specify a precise *insertion point* for the paste procedure.
2 What is it that is going to be pasted in at this point? The answer to this question is clear: the entire current contents of the Clipboard.
3 Finally, we need a command to get Macintosh to actually execute the paste operation. Its name: **Paste**. This command causes the contents of the Clipboard to be transferred to the insertion point of the target document, but it does *not* erase the Clipboard. In other words, you can paste in the same contents of the Clipboard, if you so desire, at several different points in the target document, or even in several different target documents.

In order to see how this second half of the cut & paste technique works, what we need now is some kind of a target document that we might "paste into". By way of an example, let us take the simplest information-recording device that exists in the case of Macintosh: the desk accessory called the **Note Pad**.

Pull down the menu designated by the Apple image, and choose the Note Pad as indicated:



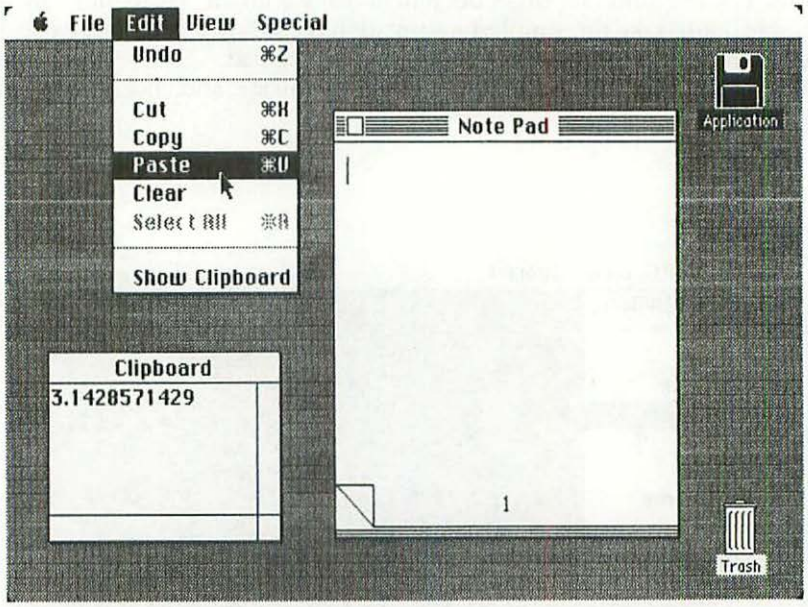
We have used the size-box and window-dragging techniques to squeeze a much-reduced Clipboard window into the lower-left corner of the screen, to make room on the desktop for the Note Pad.

The Note Pad itself (see following illustration) is a block of eight numbered sheets of "electronic paper" on which you can jot down anything you like in the way of text. (Unlike the Clipboard, the Note Pad does *not* accept images.)

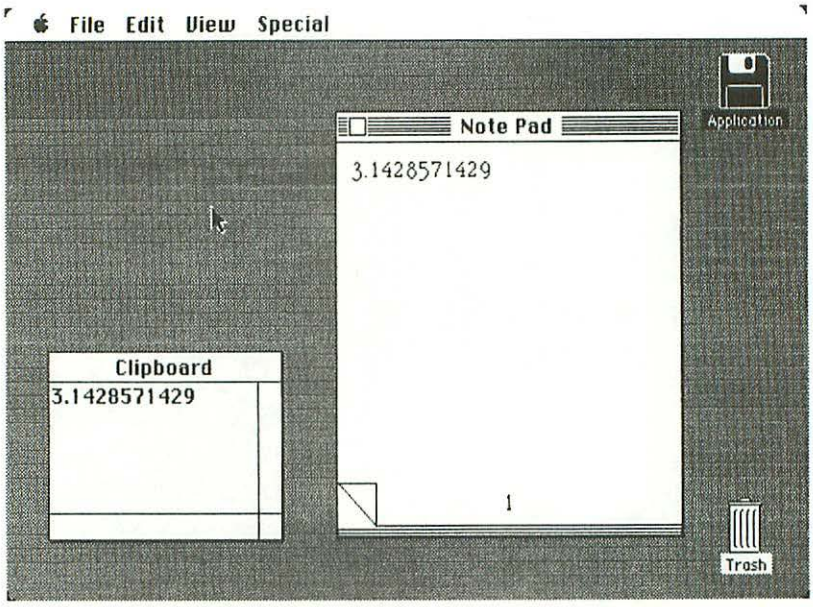
Like the Clipboard, the Note Pad — which is blank in the present example — might be described as a rather "ordinary" Macintosh window. It has a conventional title bar and close box, but neither a size box nor scroll bars. We shall look at it in greater detail later on.

In the upper-left corner of page 1 of the Note Pad, there is a vertical bar . . . which blinks, in reality. This marks the insertion point, and so it is here that our pi will be pasted.

To actually carry out the paste operation, we simply pull down the Edit menu and choose the Paste command.



The value of pi, recorded on the Clipboard, is immediately pasted onto page 1 of the Note Pad, as shown in the following illustration:

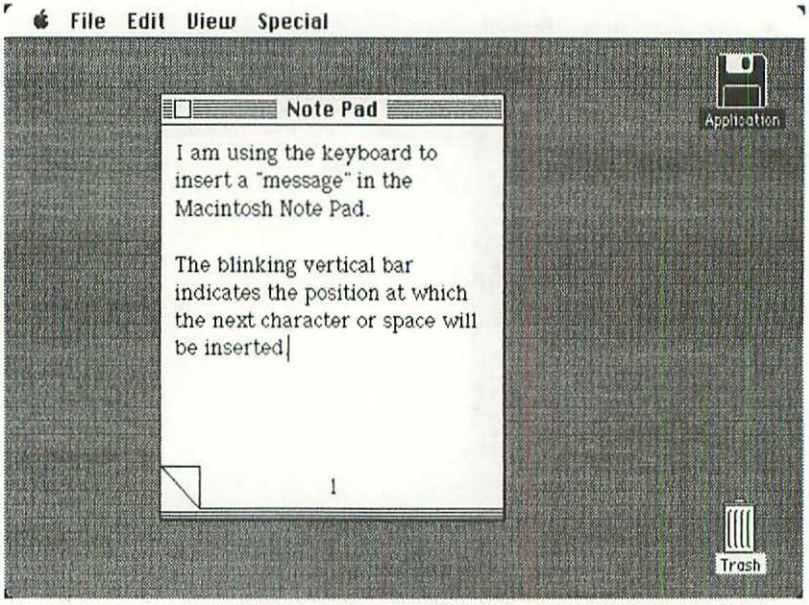


Notice that the value of pi is still present on the Clipboard. It will stay there — as we have pointed out already — up until another Cut or Copy command is executed.

The Note Pad

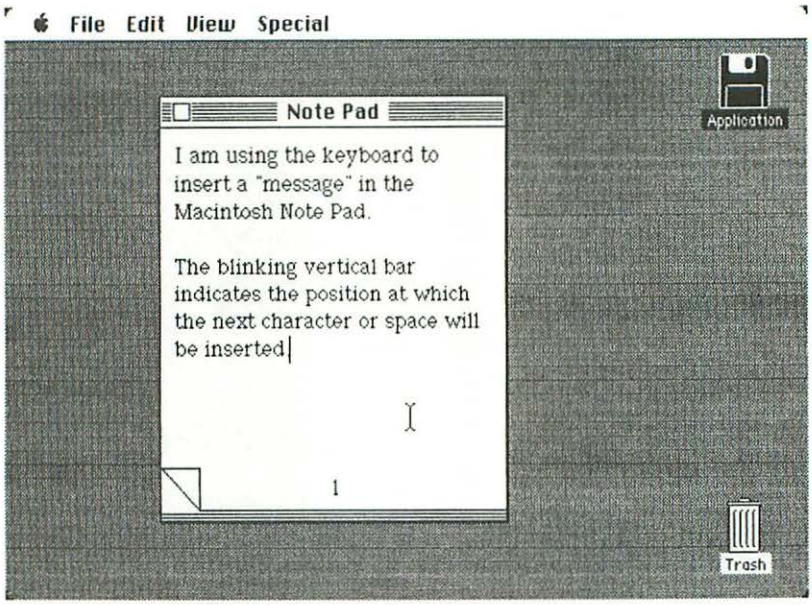
Let us take a closer look at this useful desk accessory called the Note Pad.

We have already observed the case in which a blank Note Pad is opened up on the desktop. A blinking vertical bar (not to be confused with the pointer) marks the insertion point, which is simply the position at which you can start to add new information to the Note Pad. You can then abandon the faithful mouse, if you like, and start using the keyboard to type something in the Note Pad, as shown in the following illustration:

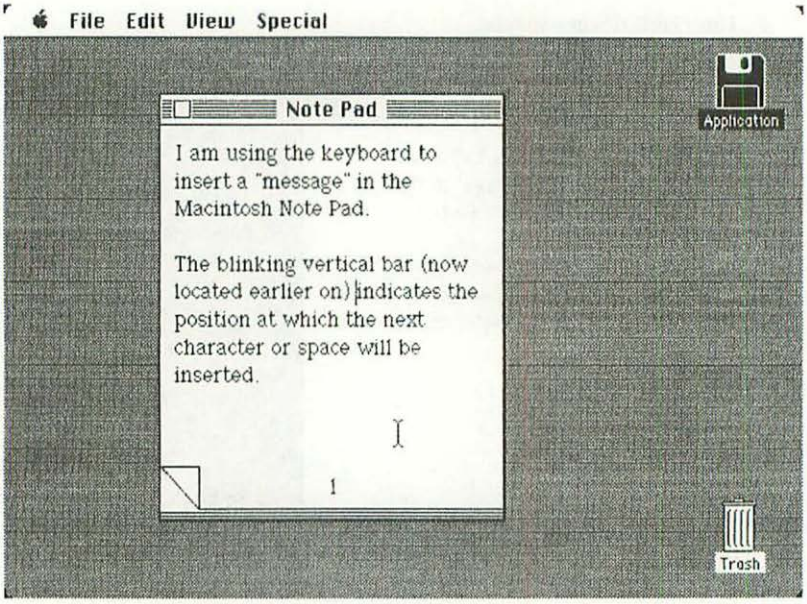


The blinking vertical bar moves along as you type. In the present example, the insertion point is now located just to the right of the last character that was typed: the full stop following the word "inserted". When typing, you don't have to worry about hitting <**Return**> to skip to a new line, for the machine does this for you automatically. To obtain the blank line between the two paragraphs, though, it *was* necessary to hit <**Return**> twice. It should be mentioned that Macintosh gets upset and starts to beep if you attempt to enter more text than can be fitted into a page of the Note Pad.

Notice that there is no familiar pointer anywhere on the desktop in the preceding illustration. It's a fact that, when you're entering information into the Note Pad, the pointer momentarily disappears, and all you see is the blinking vertical bar that represents the insertion point. To make the pointer reappear on the desktop, you only have to move the mouse a little. But the pointer that reappears on the Note Pad is no longer the familiar arrow. It has become a so-called *I-beam*, as shown in the following illustration:

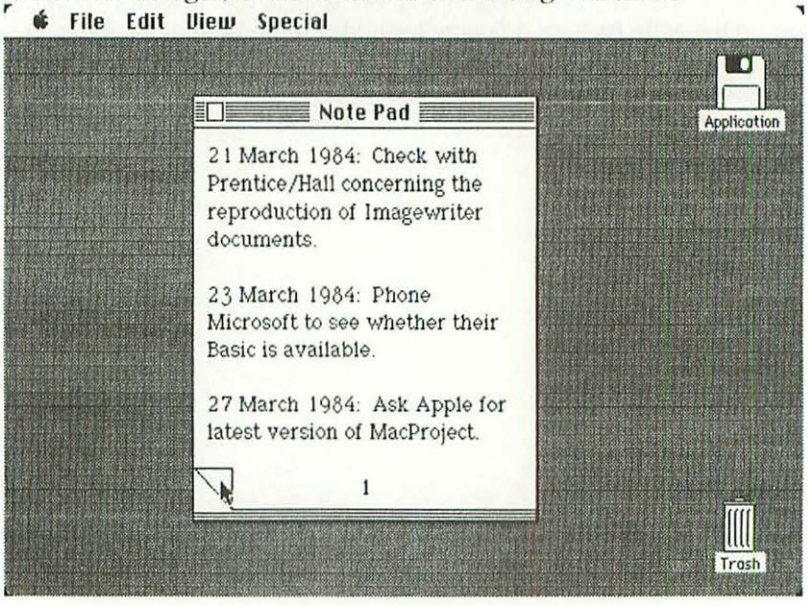


This is the type of pointer you'll encounter in the Macintosh context whenever it's a matter of entering or editing text. To see how the I-beam pointer works, use the mouse to move the pointer just to the left of the word "indicates", then click the mouse button. The result of this operation is that the insertion point changes position on the Note Pad, for it is now located at the precise spot at which you just clicked the I-beam. Type in the phrase "(now located earlier on)", for example, and the Note Pad will appear as in the following illustration, with the insertion point still located just to the left of the word "indicates":



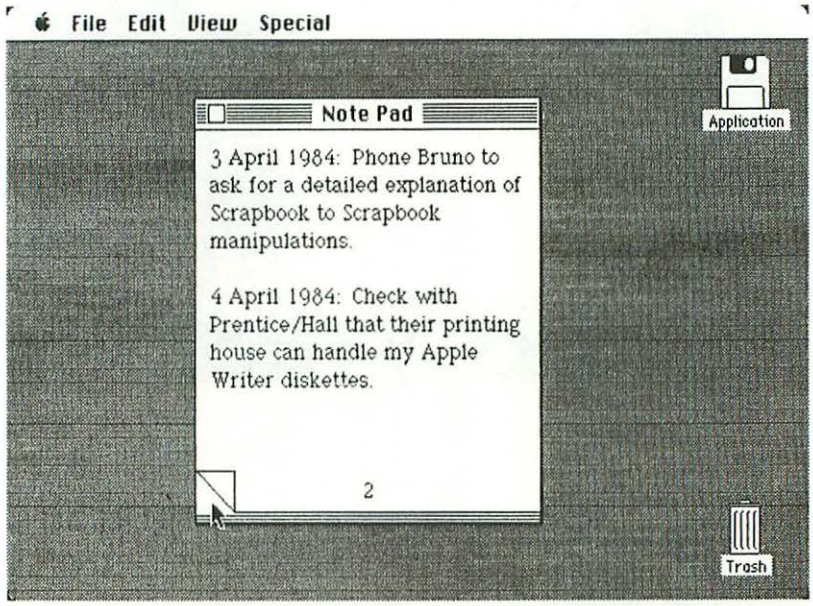
If you click the I-beam when it is located as shown, anywhere beyond the end of the text, the blinking vertical bar will return to its former position, just to the right of the full stop after the word "inserted".

Let us imagine that the Note Pad, when you open it up on the desktop, already contains various messages, as indicated in the following illustration:

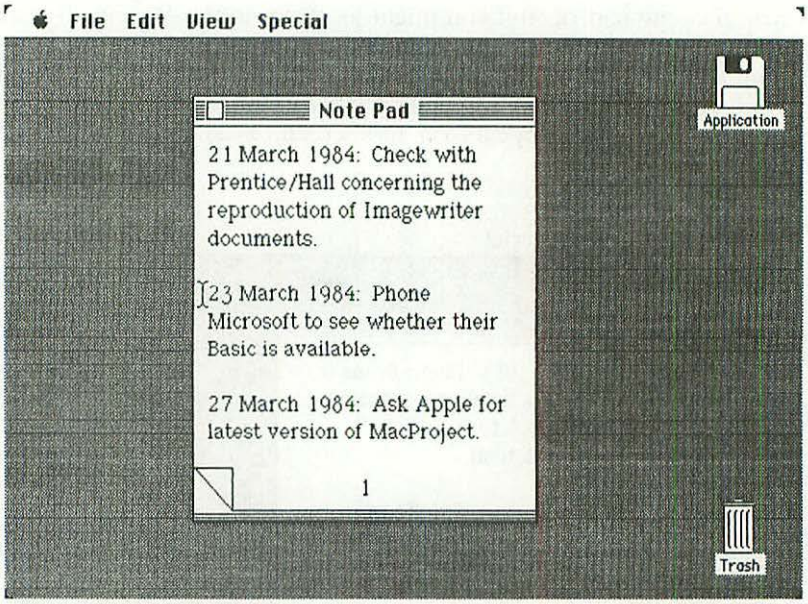


This is typical of the sort of stuff you might jot down in the Macintosh Note Pad: brief reminders of things to do, possibly with dates, like in an office diary.

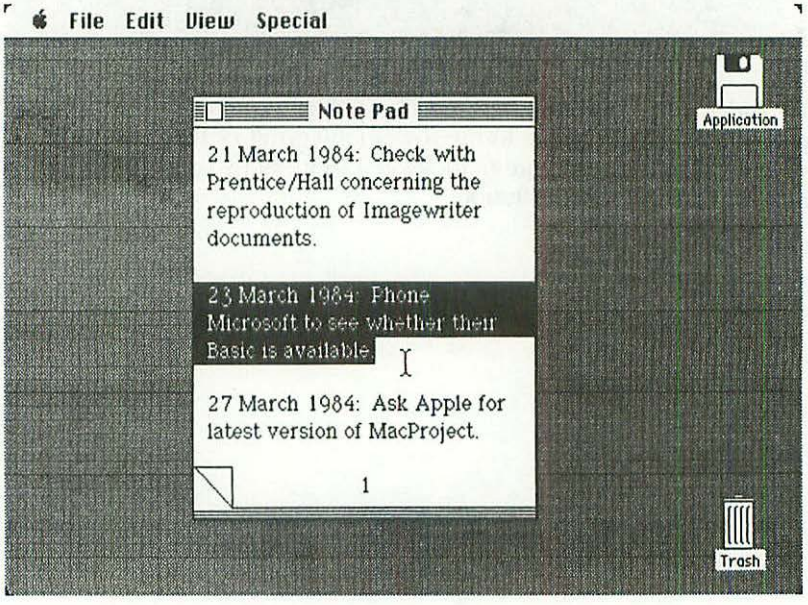
To turn the page, you simply position the pointer as indicated, in the turned-up lower-left corner of the page, and you click the mouse button. The second page of the Note Pad instantly appears on the screen, as shown in the following illustration:



Now, in order to get back to the previous page, you position the pointer as indicated, in the other triangle in the lower-left corner of the page, and you click the mouse button. The first page of the Note Pad instantly reappears on the screen, as shown in the following illustration:

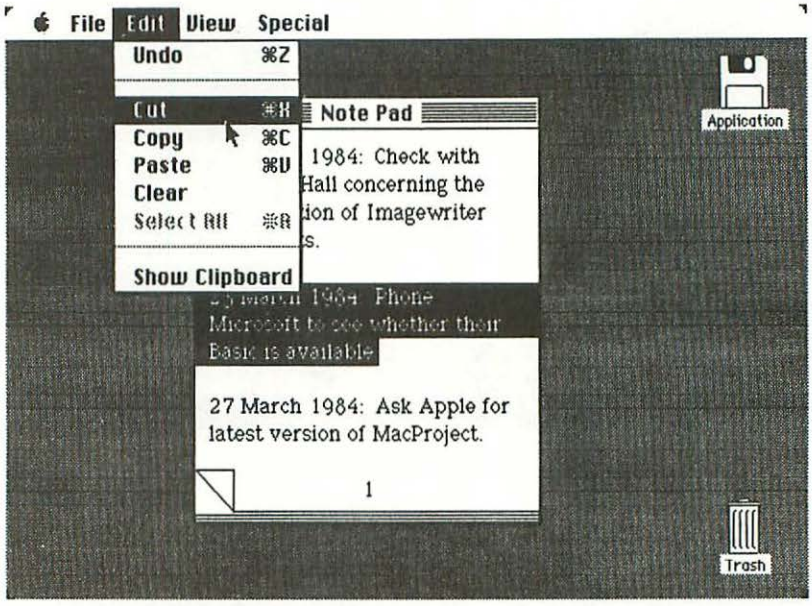


Suppose that we wish to remove the second message, dated 23 March 1984. We use the mouse to position the I-beam just to the left of the message, as indicated in the preceding illustration. Then we hold the mouse button pressed down, and drag the I-beam down across the three lines of text, in a "south-easterly" direction. The result is shown in the following illustration:

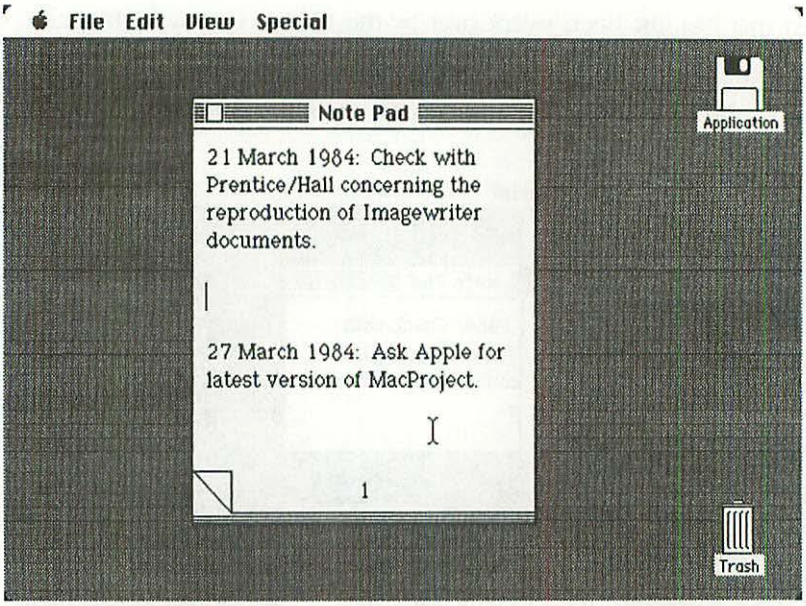


The text that has just been swept over by the I-beam is now highlighted, which indicates that it has in fact been selected.

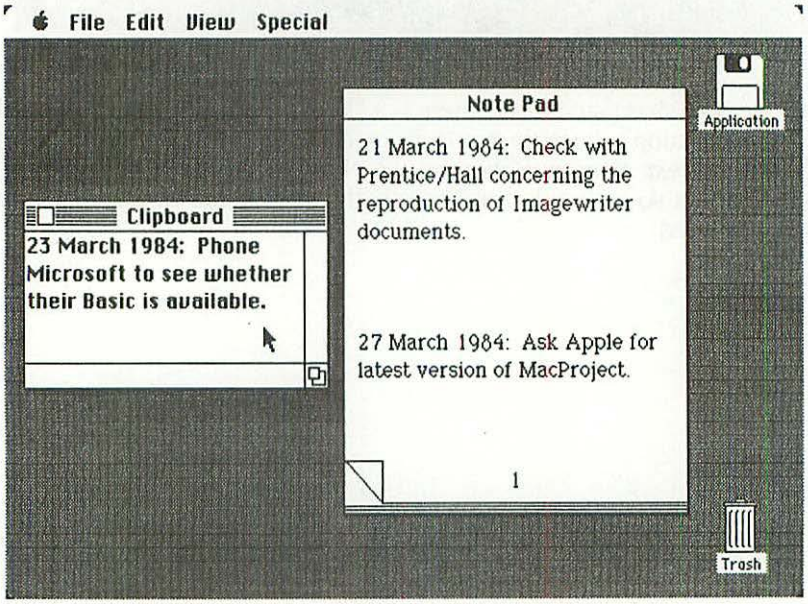
To remove this text from the Note Pad, you merely pull down the Edit menu and choose the familiar Cut command, as shown in the following illustration:



Once the removal has taken place, there is a "hole" in page 1 of the Note Pad (see following illustration), because we did not cut out any of the blank lines that surrounded the text. You could tidy up the Note Pad by deleting the extraneous blanks, if you felt so inclined, using exactly the same removal technique that we have just examined.



If we call upon the Show Clipboard command of the Edit menu to open the Clipboard on the desktop (see following illustration), we naturally discover the text that has just been cut.



If, for some reason whatsoever, we suddenly decided that this information should not have been removed from the Note Pad, then it is still possible to paste it back in again. But, as soon as another Cut or Copy command is executed, this information will disappear permanently from the Clipboard.

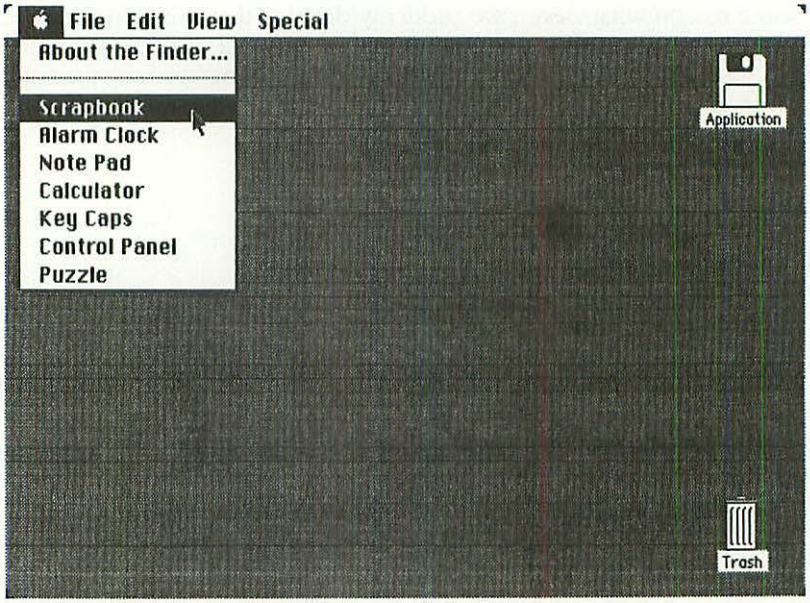
The Scrapbook

After the Clipboard and the Note Pad, there is a third member of the "cut & paste family": the **Scrapbook**. It resembles the Note Pad to a certain extent, in that it is used for recording information. But there are several important differences between the Note Pad and the Scrapbook:

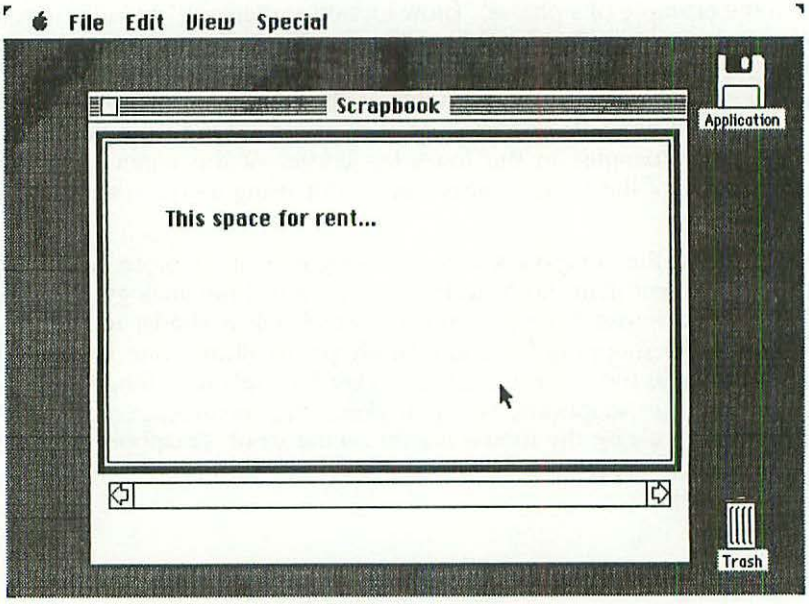
1 The Note Pad holds only eight pages of text, whereas a much larger quantity of information can be stored in the Scrapbook.
2 The Scrapbook can contain *images* as well as text.
3 Information can be recorded in the Note Pad simply by typing on the Macintosh keyboard, but this is not possible in the case of the Scrapbook. The only way of inserting anything into the Scrapbook is to place it first on the Clipboard (using a Cut or Copy command), and then Paste it into the Scrapbook.
4 A corollary of the preceding remark is that information recorded in the Scrapbook cannot be edited in the same way as Note Pad messages. We have seen the example of a phrase, "(now located earlier on)", being inserted into a Note Pad message. This would not be possible in the case of the Scrapbook, since there is no I-beam pointer for this document. Once a page is recorded in the Scrapbook, the only operation that can be carried out on it is Cut or Copy.
5 The pages of the Note Pad are turned by means of the unusual technique of clicking the triangles in the lower-left corner of the window, whereas the consultation of the Scrapbook is carried out using a conventional horizontal scroll bar.

In other words, the Scrapbook can be thought of as a more permanent and "serious" document than the Note Pad. To use a real-life analogy, the difference between the Macintosh Note Pad and the Scrapbook is similar to the difference between a family shopping list and a family photo album; one is an ephemeral document, whereas the other is intended to last for a relatively long time.

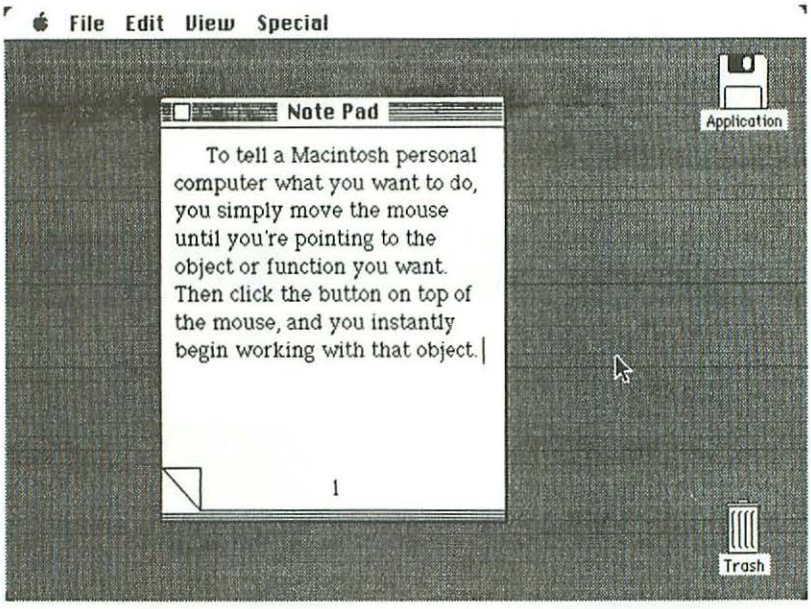
To access the Scrapbook, you pull down the menu underneath the apple symbol and you release the mouse button on the word "Scrapbook", as indicated in the following illustration:



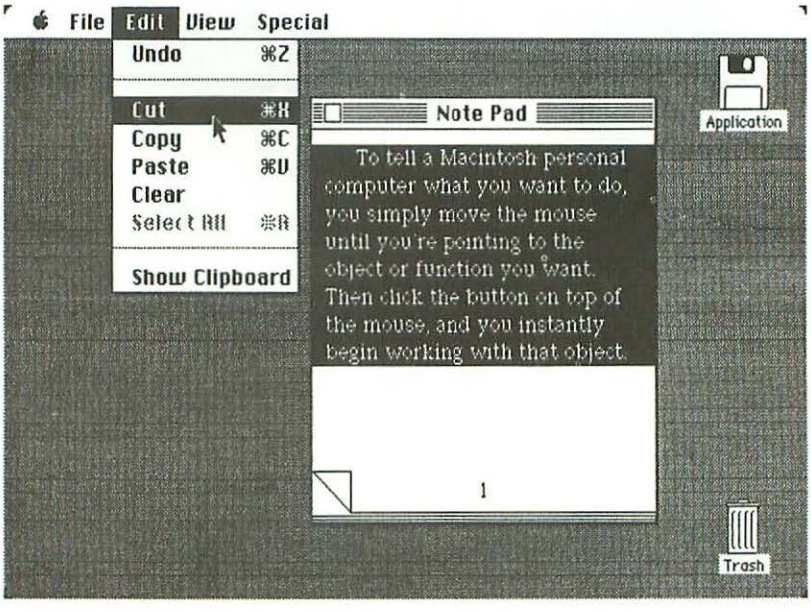
The following illustration shows us an empty Scrapbook, with a typical touch of Macintosh humor:



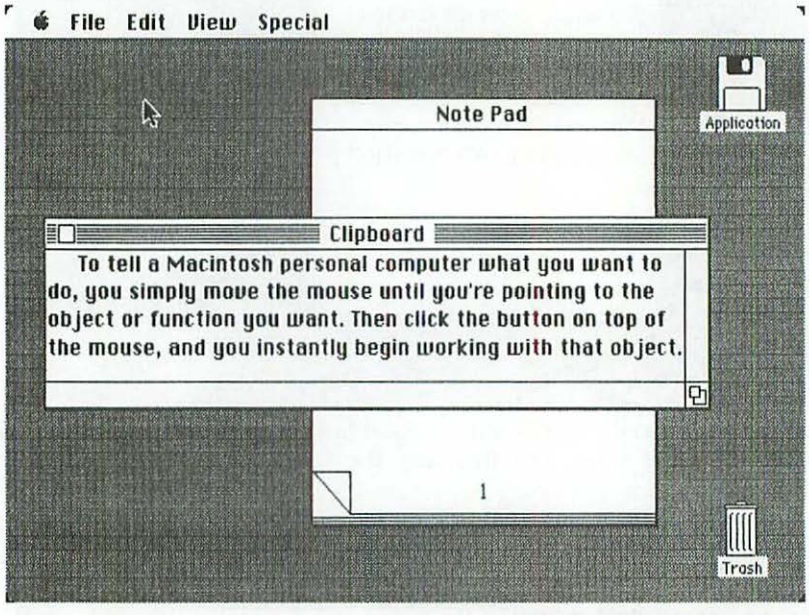
Since we need some edifying data to record in the Scrapbook, let us start out with the following short text, borrowed from Apple's promotional literature, and recorded on page 1 of the Note Pad:



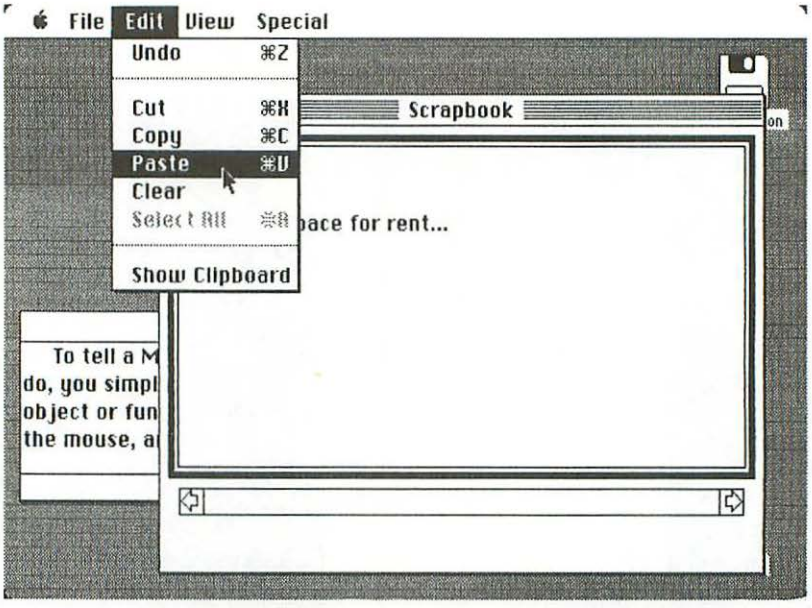
To move this text onto the Clipboard, we must first select it by dragging the I-beam down over the eight lines, and then use the Cut command, as shown in the following illustration:



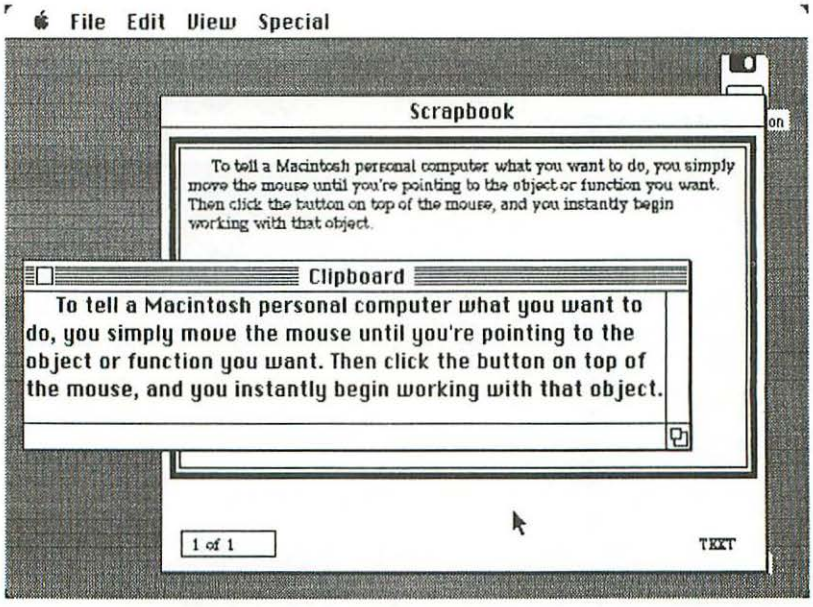
Here is the result (see following illustration). The text has been cut out of page 1 of the Note Pad, which is now blank, and moved to the Clipboard. (From now on, it is assumed that the reader is sufficiently familiar with the way in which various windows are opened, closed and moved around the desktop, so as not to need step-by-step explanations. For example, in the case of the following illustration, we did not repeat the explanations — provided already on page 37 — concerning the procedure for displaying the Clipboard window on the desktop. (In case you've forgotten, it's simply a matter of pulling down the Edit menu and choosing the Show Clipboard command.)



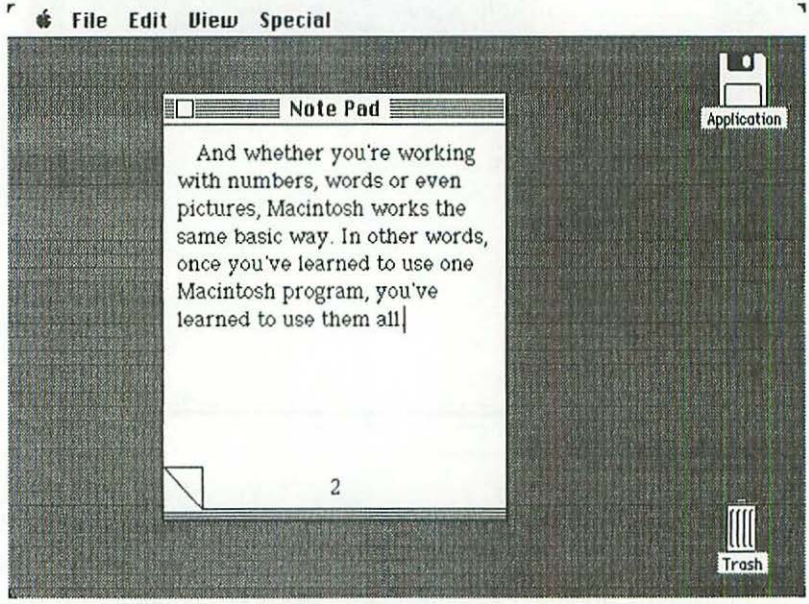
In the following illustration, we have opened the Scrapbook on the desktop, and its big window hides most of the Clipboard. In other words, it is the Scrapbook — with stripes in its title bar — that is selected, not the Clipboard. In order to move our text from the Clipboard to the Scrapbook, we simply pull down the Edit menu and choose the Paste command.



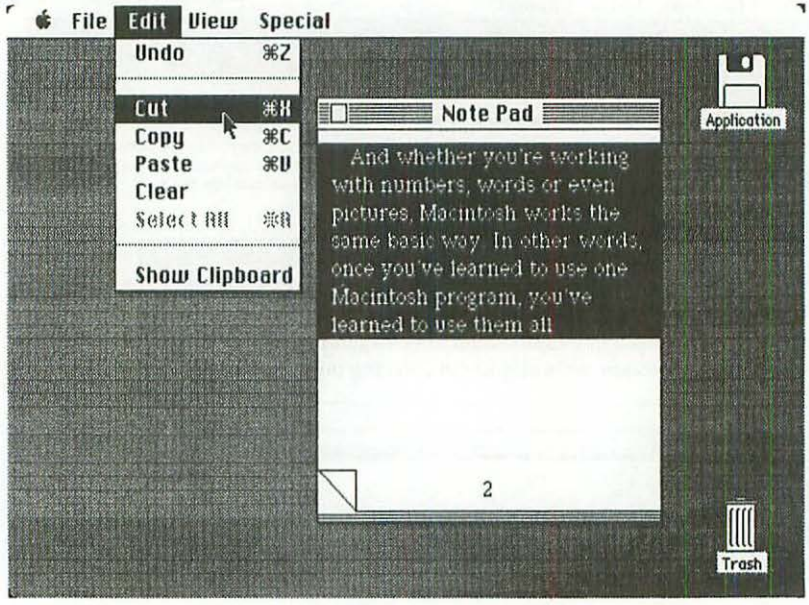
To show off the result (see following illustration), we have clicked the Clipboard to activate this window and therefore move it on top of the Scrapbook. As you can see, page 1 of the Scrapbook now contains a copy of the Clipboard text.



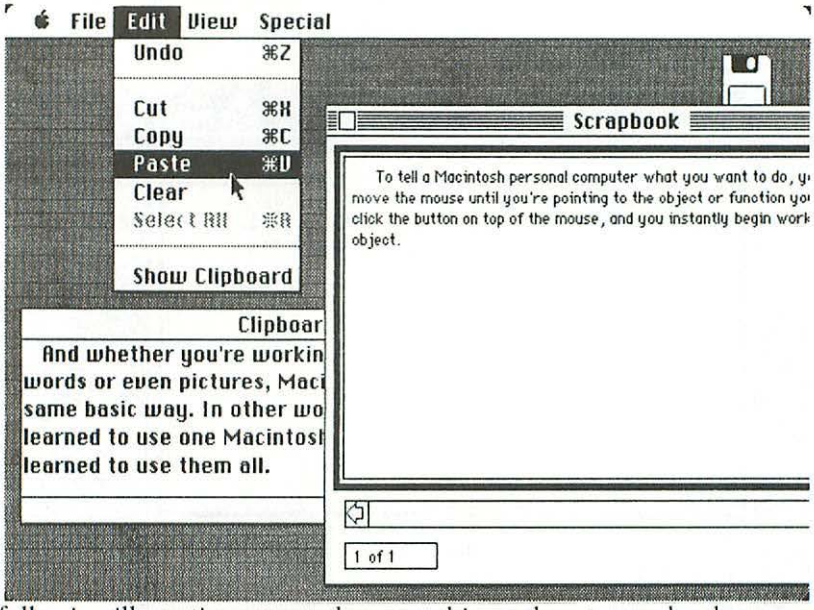
Let us record a second page of text in the Scrapbook. The following illustration shows us page 2 of the Note Pad:



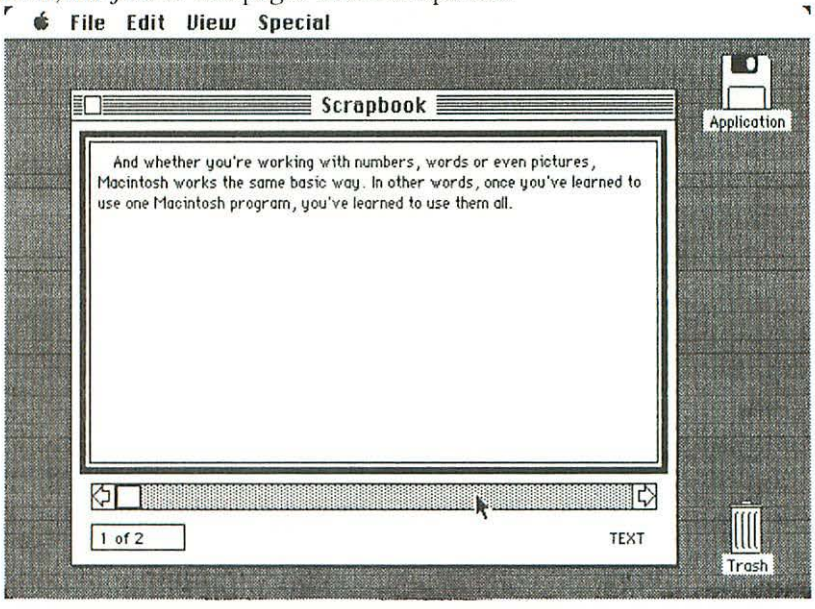
As before, we sweep the I-beam down over the text to highlight it on a black background (see following illustration), then we pull down the Edit menu and choose the Cut command.



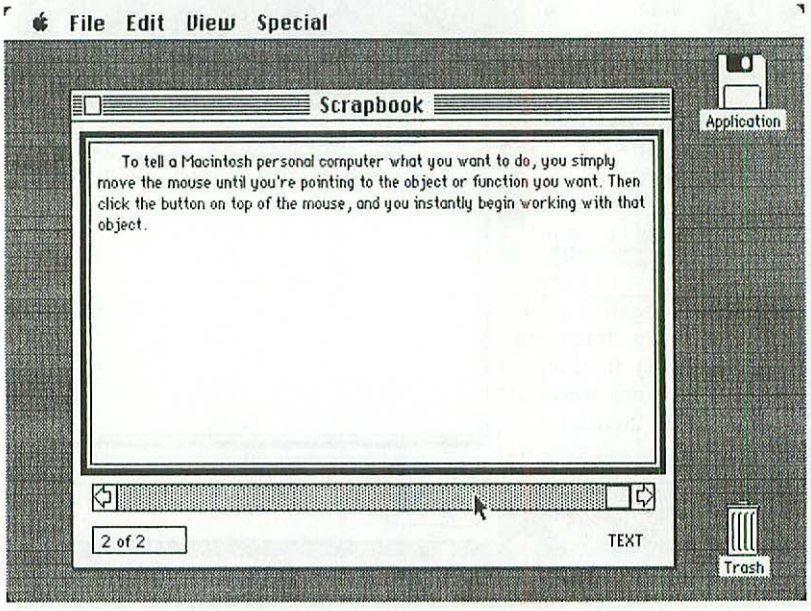
Once again, we select the Scrapbook (see following illustration). The big window of page 1 of the Scrapbook has been dragged over the righthand edge of the desktop, but it still hides a good part of the Clipboard, which contains the text that we are about to paste into the Scrapbook. As usual, we pull down the Edit menu and choose the Paste command.



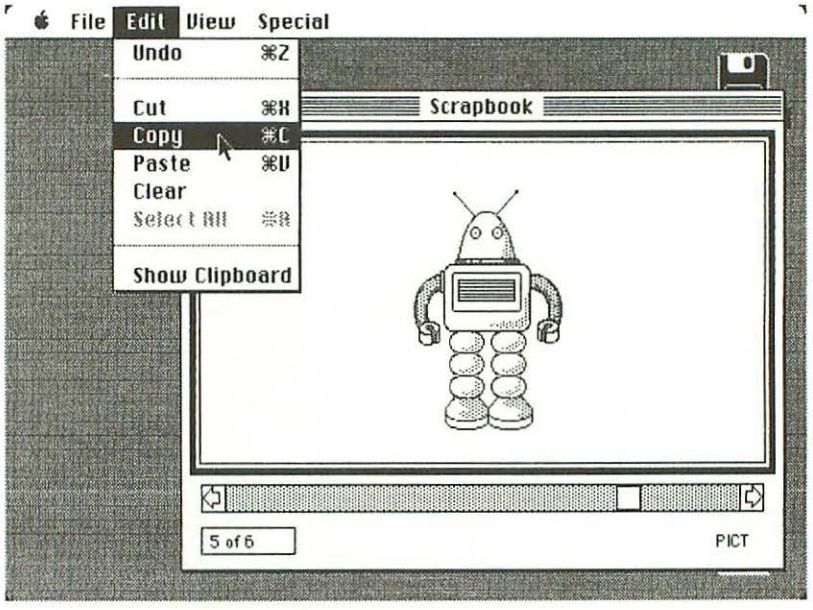
The following illustration proves that something rather strange has happened. This *second* page that we have just inserted into the Scrapbook is in fact labeled "1 of 2": that is, the *first* of two pages in the Scrapbook.



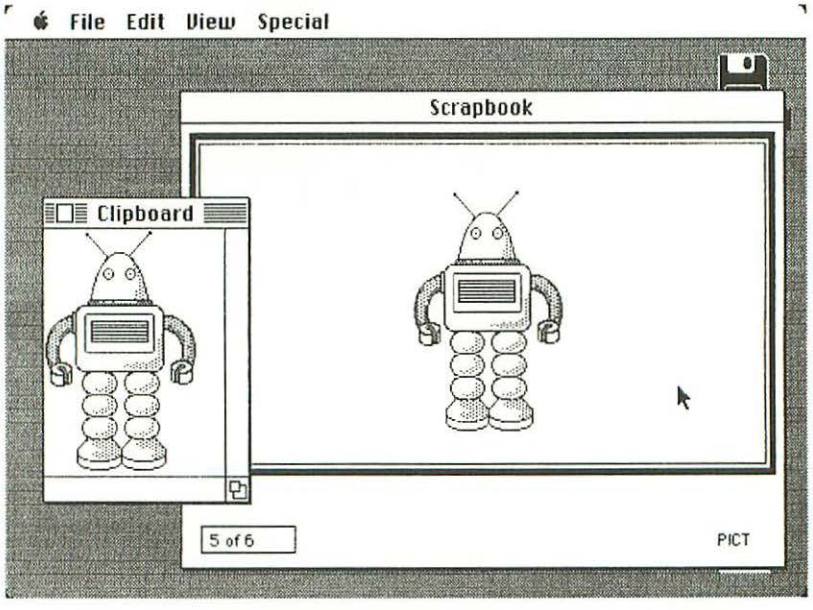
To jump to the second page of the Scrapbook, you click the gray area of the horizontal scroll bar. The following illustration reveals that the *first* page we inserted into the Scrapbook is labeled "2 of 2". In other words, each new page is inserted into the Scrapbook just before the page that was previously visible on the desktop, and all existing pages beyond that point are renumbered and pushed back — as it were — towards the end of the Scrapbook.



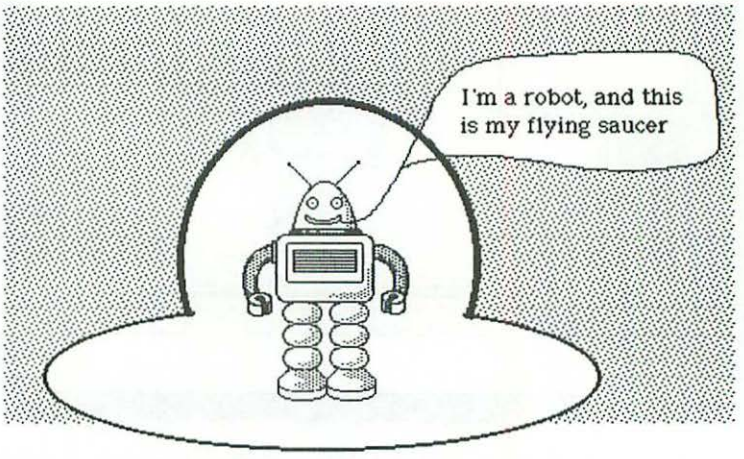
Obviously, if we go to the trouble of storing information in the Scrapbook, it's generally because we intend to copy this information into further documents, at a later stage. Let's take the example of an image stored on page 5 of the Scrapbook, as shown in the following illustration:



If the Edit menu is pulled down, and the Copy command chosen, the robot gets copied onto the Clipboard. The Show Clipboard command proves that this copy has been carried out successfully, as seen in the following illustration:



Now that the image of the robot is on the Clipboard, we can use it in any way we desire. That's to say, we can manipulate it by means of various application programs. For example, the following illustration provides a trivial example of the manner in which it's possible to play around with the robot's image using the facilities of the *MacPaint* tool:



This is the end of our presentation of the cut & paste concept . . . but the concept itself will reappear in the following pages, for it represents one of the fundamental procedures in the Macintosh context.

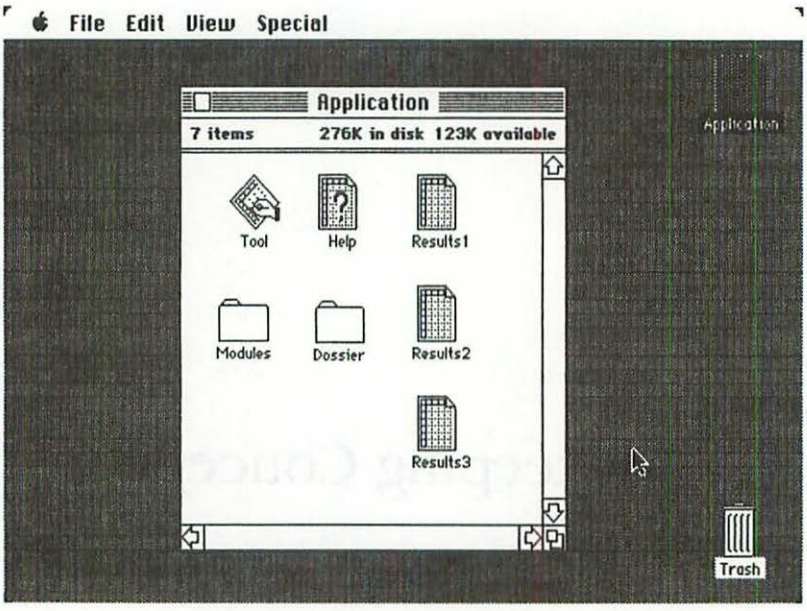
chapter 4
The Housekeeping Concept

In computing, the term "housekeeping" designates a vital activity related to the organization and management of information files, particularly on disks. Macintosh uses a software component called the *Finder* to carry out such tasks. Now, many of the Finder's capabilities have already been mentioned. It is the Finder that enables the Macintosh user to open icons and to manipulate windows, for example, and to carry out cut & paste operations. So, what we intend to do in the present chapter is to take a look at other Finder tasks that have not yet been illustrated.

Folders

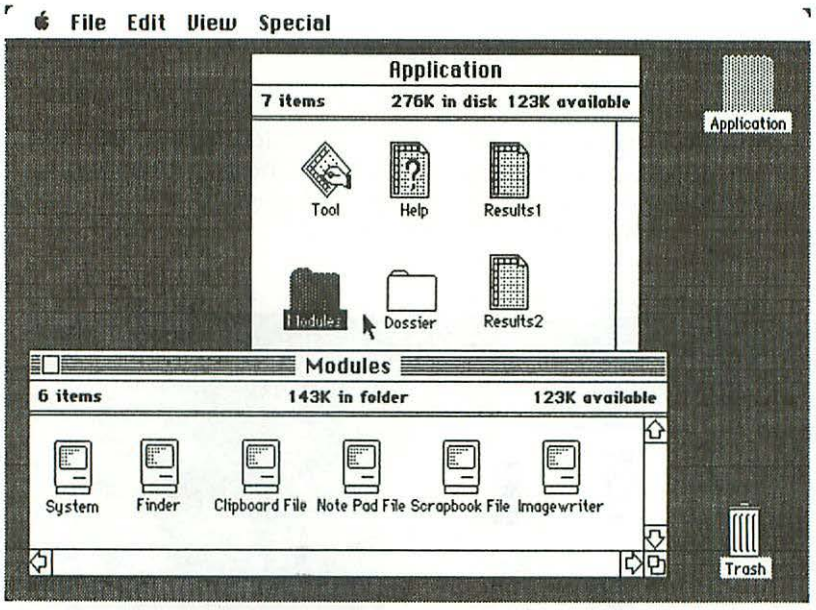
The notion of Macintosh *folders* has already been mentioned earlier on, but no examples were given of the way in which they are used for housekeeping.

The following illustration shows the directory window of the disk called "Application" that we have been using for all our examples:



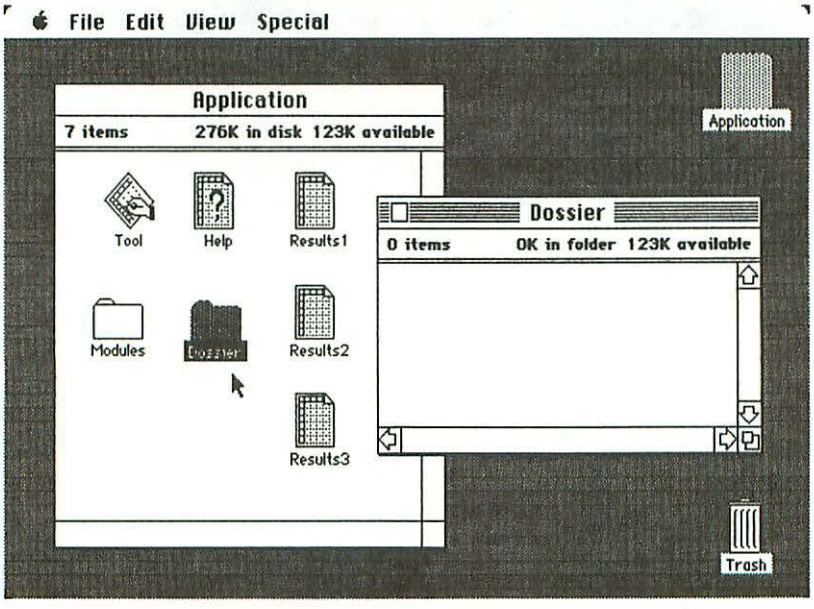
Notice that three new icons have appeared in the window. What do they represent? Without worrying about what "Tool" actually does, it suffices to say that, every time you run this application program, you have the opportunity of creating a new document represented by an icon of this nature. So, we can assume that "Tool" has been run three times, resulting in the creation of the documents labeled "Results1", "Results2" and "Results3". But don't bother, for the moment, about what these documents actually represent.

Let us turn our attention now to the pair of folders called "Modules" and "Dossier". To see what's inside the first one, you simply double-click it with the mouse button, just as you do to open any icon whatsoever. The result is shown in the following illustration:



"Modules" is quite a big and important folder, for it contains six fundamental software components that occupy more than a third of the disk. The presence of these elements on the disk confirms that "Application" can be considered as a *startup disk*. This means that you can take such a disk and insert it into the Macintosh drive as soon as the machine is switched on, whereas disks that do not contain the files named "System" and "Finder" cannot be used in such a manner.

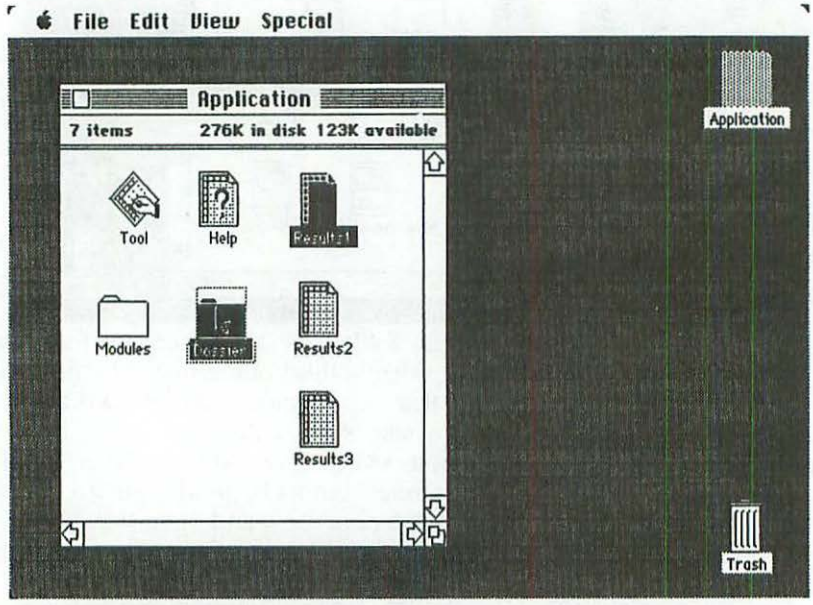
Close "Modules" now, by clicking the close box, and open the other folder, called "Dossier". The result is shown in the following illustration:



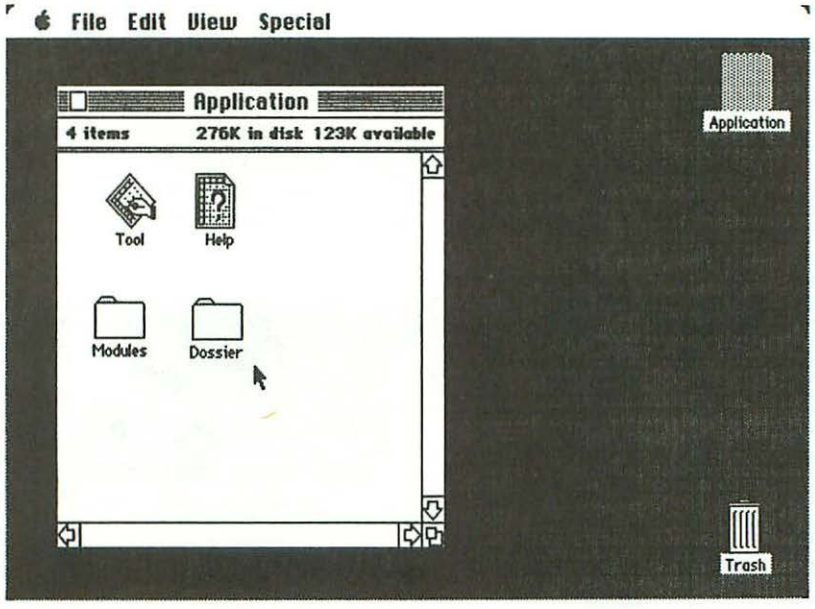
It appears that "Dossier" is a completely empty folder. So, why not make use of it to store away the three results obtained by "Tool"?

Close "Dossier" for the moment, by clicking its close box, so that the directory window is once again completely visible.

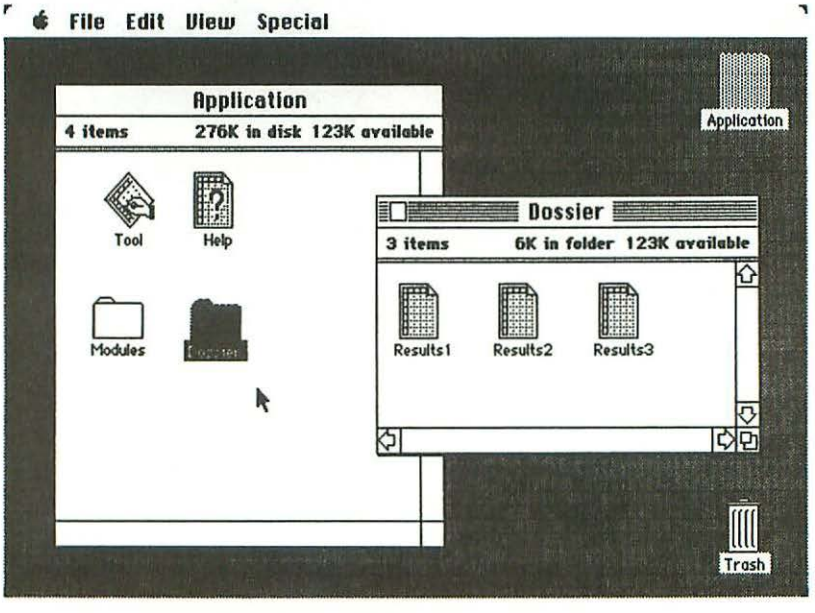
Now, to actually move each document into the folder, you simply position the pointer on the document icon, then drag it across to the folder icon (see following illustration) so that the flickering outline of the dragged icon actually covers the "Dossier" icon. Then you release the mouse button.



Each time that you release the mouse button, the document icon that you just dragged actually disappears from the window, because it has been placed — as it were — *inside* the folder. So, after the three dragging operations, the document icons appear to have disappeared from the window, as shown in the following illustration:

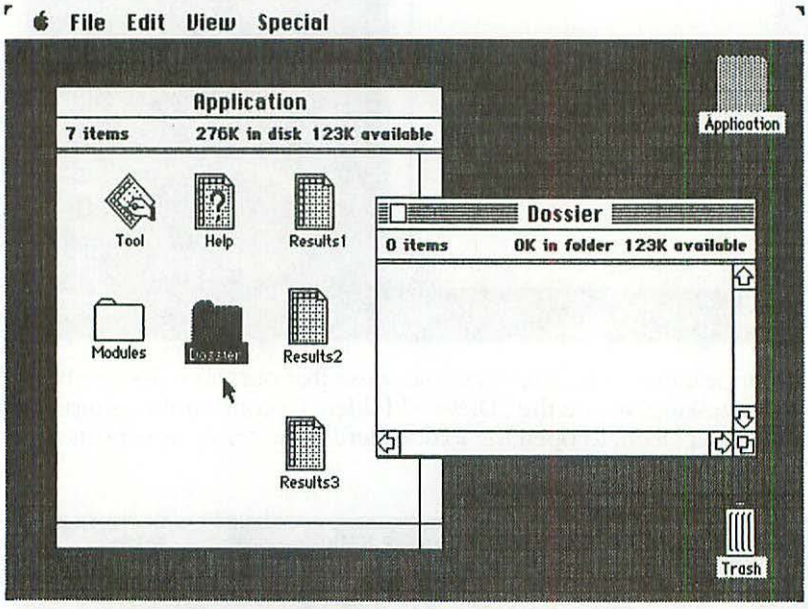


But this disappearance is an illusion in the sense that our three documents are still there on the desktop, inside the "Dossier" folder. To confirm this, simply double-click the "Dossier" icon, to open the folder. Here is the result, as expected:

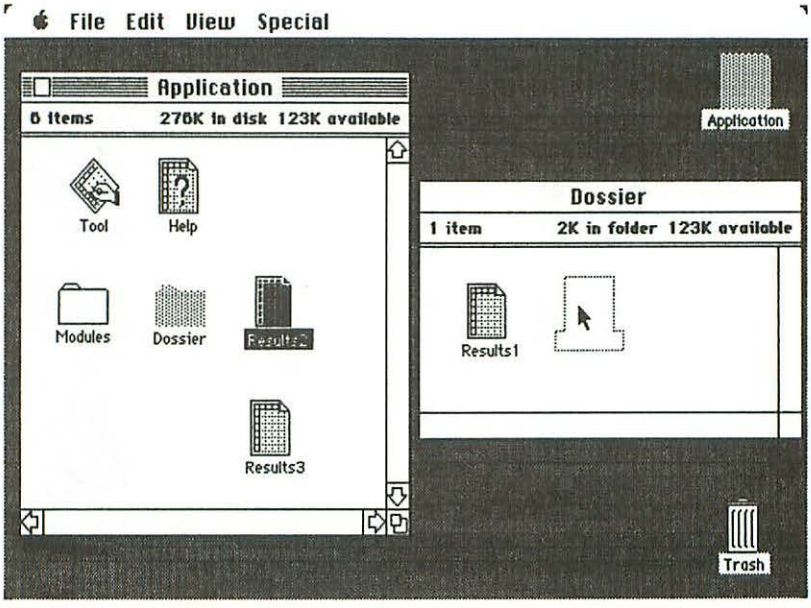


Now, this operation of placing three new documents in a previously-empty folder can be carried out in another fashion, which you might prefer. So, let's return to the initial situation where "Dossier" is empty and the three document icons are located in the "Application" window.

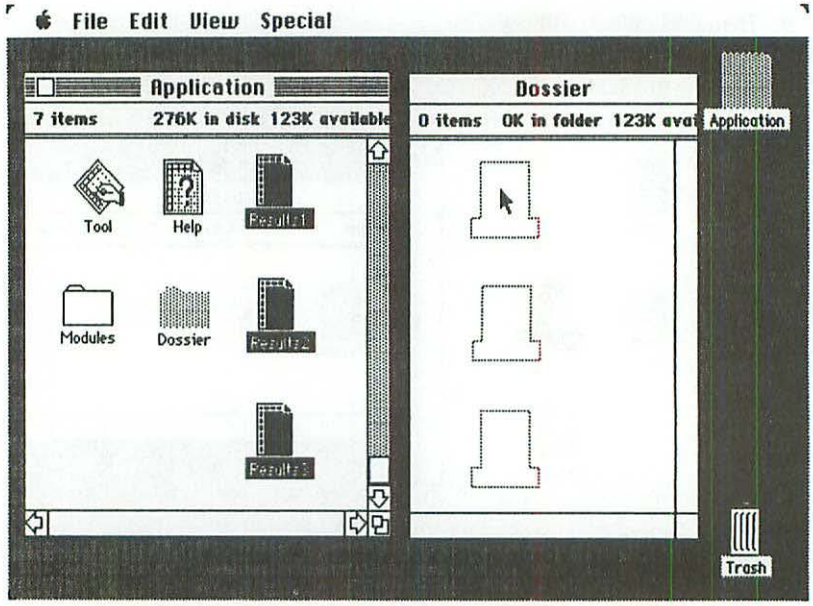
First of all, we double-click the "Dossier" icon so as to open up the empty folder on the desktop. Now, this time, we are actually going to leave the "Dossier" window open on the desktop, alongside the "Application" window, during the three dragging operations, as shown in the following illustration:



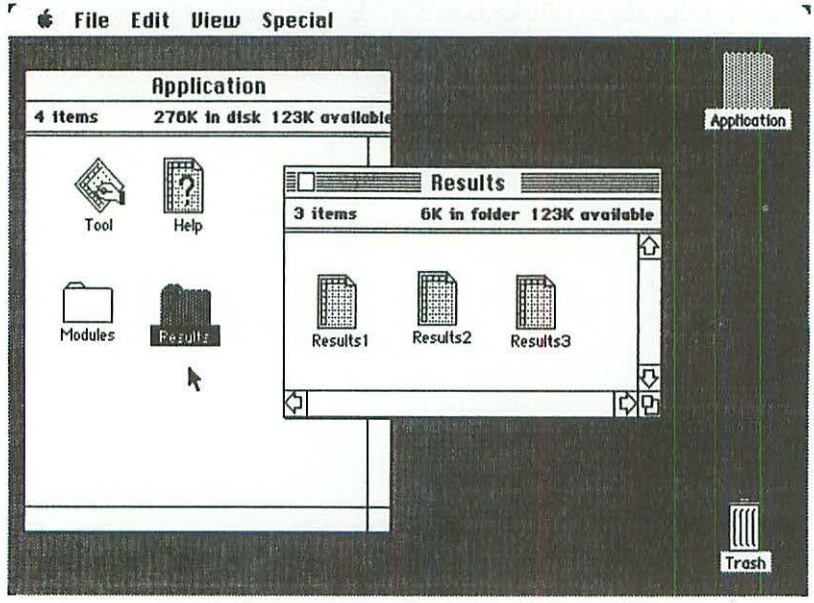
Instead of dragging each document icon onto the "Dossier" icon, as in the previous procedure, this time you actually drag each document icon out of the "Application" window (see following illustration) and across into the "Dossier" window. In other words, it's as if you physically transferred the three icons from one window into the other.



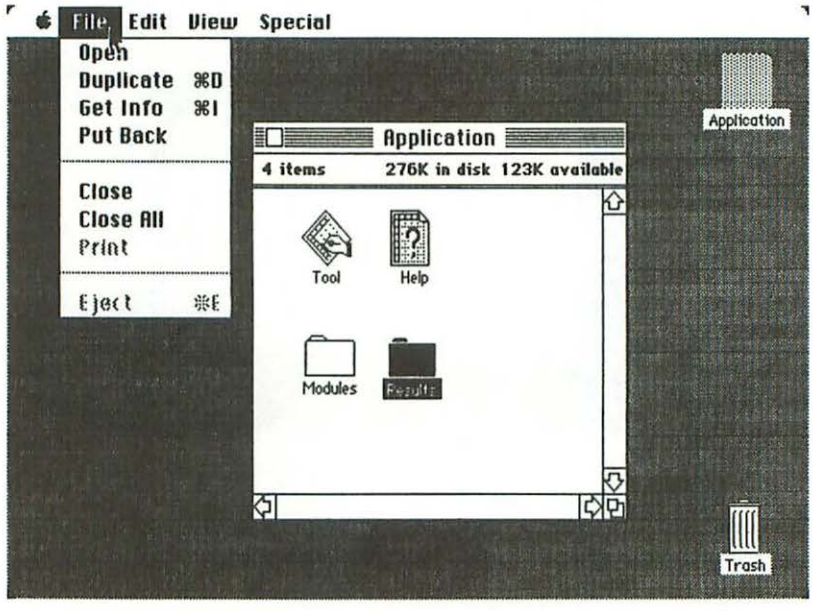
There's even an amusing variation on the dragging procedure, if you're really in a hurry to insert the three documents into the empty folder. This involves the technique of *multiple selection.* Up until now, the general rule was that we only ever selected one icon at a time on the desktop. But Macintosh provides a means of overriding this rule. To select all three document icons simultaneously, you start out by clicking the first icon in the usual manner. Then you hold down the <Shift> key on the Macintosh keyboard while clicking the two other icons. The result (see following illustration) is that all three icons are highlighted, confirming their selection, and a single drag operation suffices to move all three icons simultaneously from the "Application" window to the "Dossier" window.



Another elementary housekeeping function that you might like to carry out at this point is to *change the name* of the folder that now contains the three new documents. To do so, you simply click anywhere in the title bar of the "Dossier" window. (Not in the close box, of course!) At that point, you can immediately start typing in a new name using the Macintosh keyboard. The old name, "Dossier", disappears instantly, and it is replaced by the new name you type. For example, in the following illustration, we have chosen to rename this folder "Results":



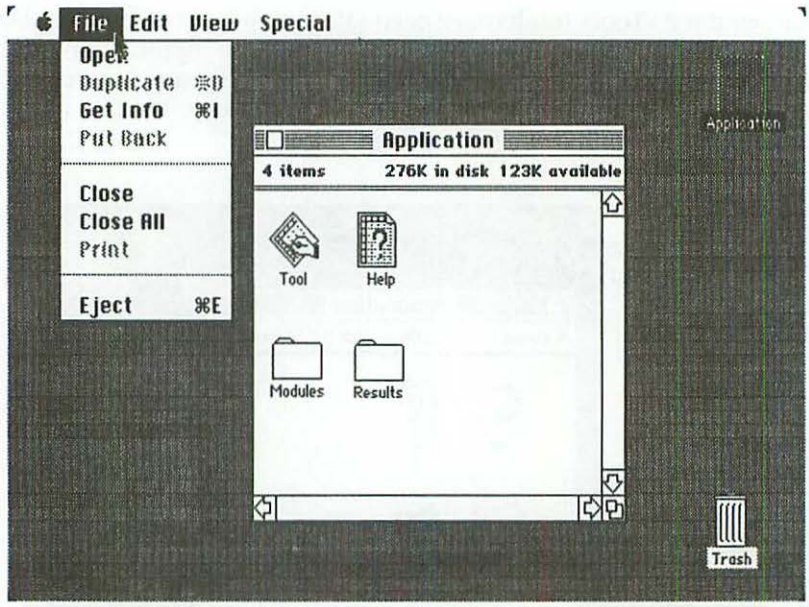
Now that our three "Tool" results have been safely stored away in a folder labeled "Results", we might decide that it is time for us to eject the "Application" disk from the machine, just as we did earlier on (see page 35). Let us suppose that we click the close box of the "Results" window, and then pull down the File menu. The outcome is shown in the following illustration:



As you can see, Macintosh does not allow us to choose the Eject command, which is dimmed. Why is this?

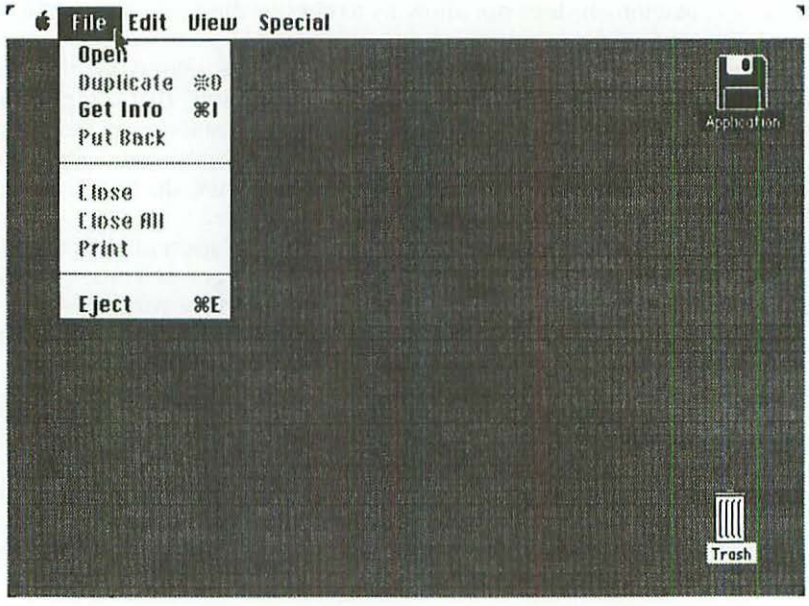
Common sense alone enables us to understand the reason for this refusal. Macintosh considers — quite rightly — that the only "thing" that can possibly be ejected from the machine is . . . a disk. Now, since the "Results" icon has remained selected, Macintosh stipulates that the Eject command is not valid, because the notion of ejecting a folder is nonsense. So, what we must do is to *deselect* the "Results" icon and select the "Application" icon in its place.

There are several ways of doing this. The simplest solution of all is to click the "Application" icon itself. The "Results" icon is no longer selected. As for the "Application" icon, it remains hollow, because the directory window is open on the desktop, but it turns from white to black, as shown in the following illustration:



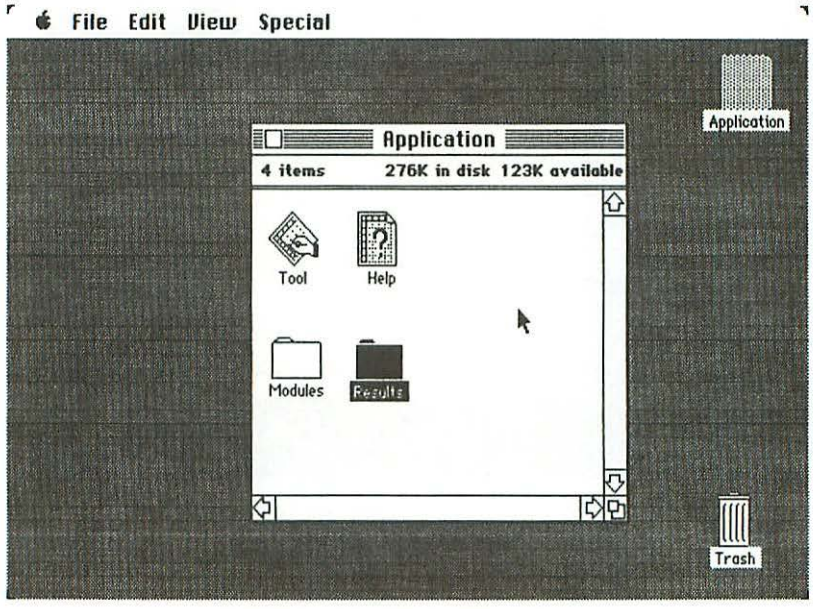
Since the "Application" disk is selected, the Eject command is now valid.

Another simple manner of selecting the "Application" disk is to close the directory window, either by clicking the close box or by choosing the Close command from the File menu. In that case, the directory window disappears, and the highlighted "Application" icon — which is no longer hollow — reappears as in the following illustration:



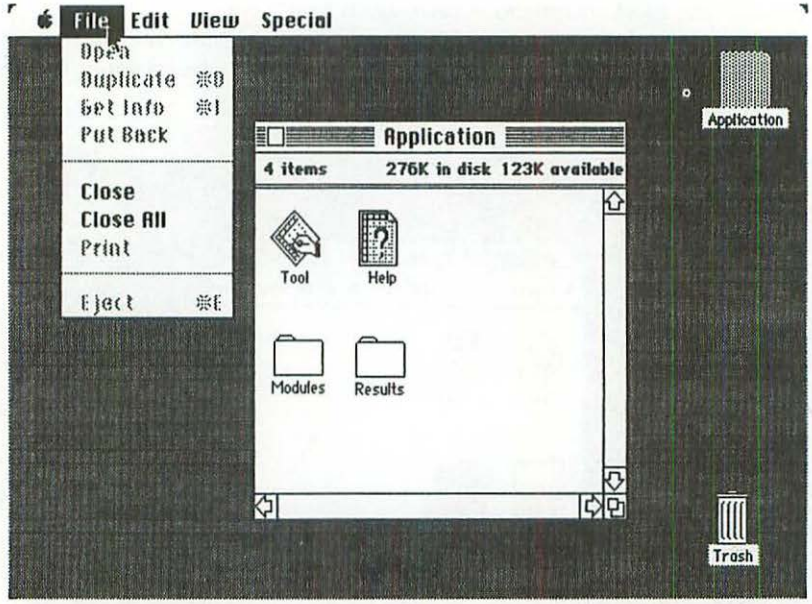
As you can see, the Eject command is once again valid.

There are still other ways of achieving the same result. Let's return to the situation represented by the following illustration, with the pointer located as shown:

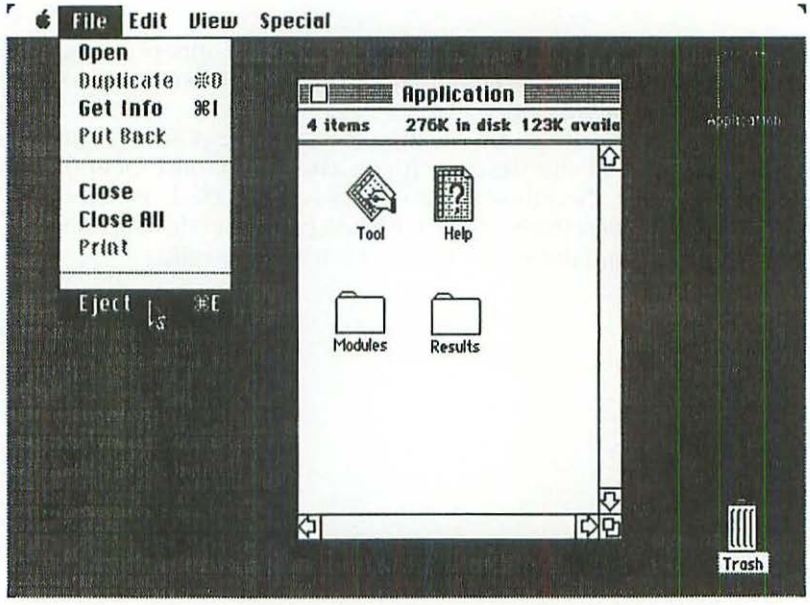


If you click the mouse button now, with the pointer in this position, then the "Results" icon is deselected, and the hollow "Application" icon turns from white to black, indicating that it is selected.

Let us finally imagine the same situation, but with the pointer located out on the gray no-man's-land of the desktop. If you click the mouse button, with the pointer in this position, then the "Results" icon is deselected, as before, but the "Application" icon is *not* selected. In fact, every icon on the desktop is white (see following illustration), and the Eject command is no longer valid.



But, since we wish to get the "Application" disk out of the machine, let us imagine that we have selected the icon as required, and chosen the Eject command, as indicated in the following illustration:



As soon as you release the mouse button, Macintosh purrs like a contented cat for several seconds, while making sure that all the files have been correctly closed, and then the "Application" disk suddenly pops out of the front of the drive, like a candy bar in a vending machine.

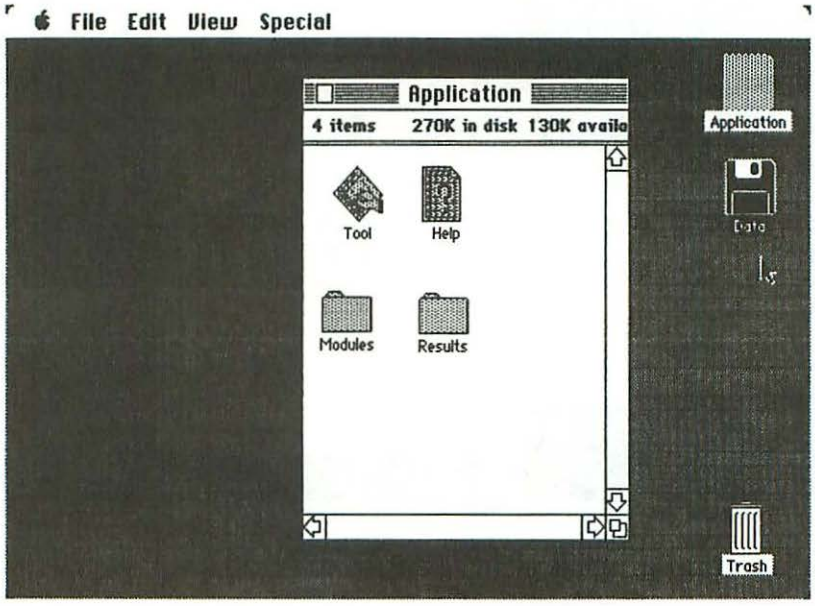
Disk-to-disk transfers

With most professional microcomputers, there must be a procedure that enables the user to copy files from one disk onto another.

In the case of Macintosh, this need arises, for example, when we create a large volume of documents using an application program. Normally there won't be much room left on the startup disk that houses the application program, so the best solution is to transfer documents, as they are created, from the startup disk onto a *data disk*.

Let us imagine that we have a completely blank disk called "Data", and that we would like to transfer the "Results" folder from the "Application" startup disk onto "Data".

The Macintosh disk drive is empty for the moment, since we have just ejected the "Application" disk. (*Warning*: Don't turn the power off, then back on, otherwise nothing in the next few pages will work as indicated!) Now, all we need to do — without touching either the mouse or the keyboard — is to insert the "Data" disk in the drive. After a bit of humming and purring — during which time a *wristwatch icon* appears on the desktop, indicating that Macintosh expects you to be patient — the following display appears on the screen:

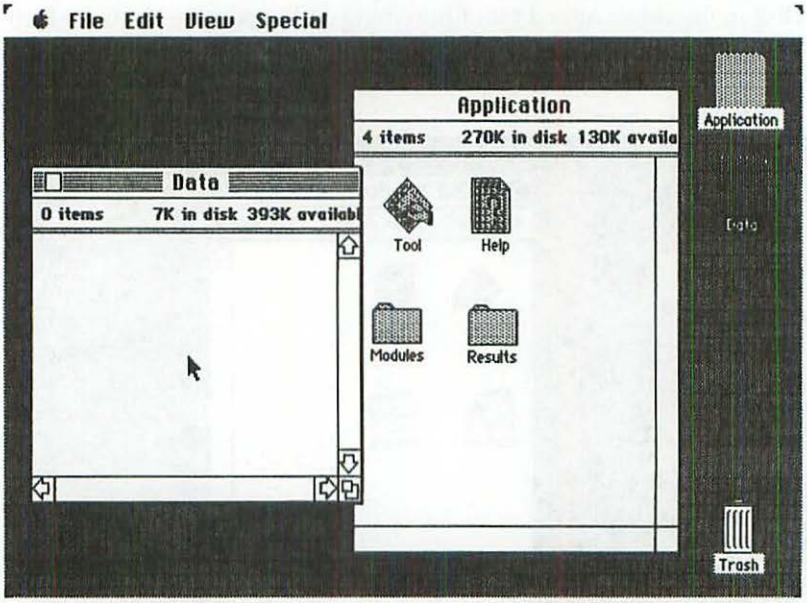


Notice carefully the different tones of the seven icons on the desktop. The "Data" icon is highlighted in black, to indicate that it is currently selected. The trash can is white, as usual, indicating that it is present on the desktop (which might appear to certain readers as a curious location for a trash can!), but not selected. But look at the "Application" icon and the four icons in the directory file window. They have all turned light gray.

This is the code used by Macintosh to inform the user that the "Application" disk is no longer present in the drive — which you knew, in any case — but that it still plays the role of the startup disk. (Don't forget that "Data", as we have pointed out already, is *not* a startup disk, since it does not contain all the software components that we observed earlier on in the folder called "Modules".)

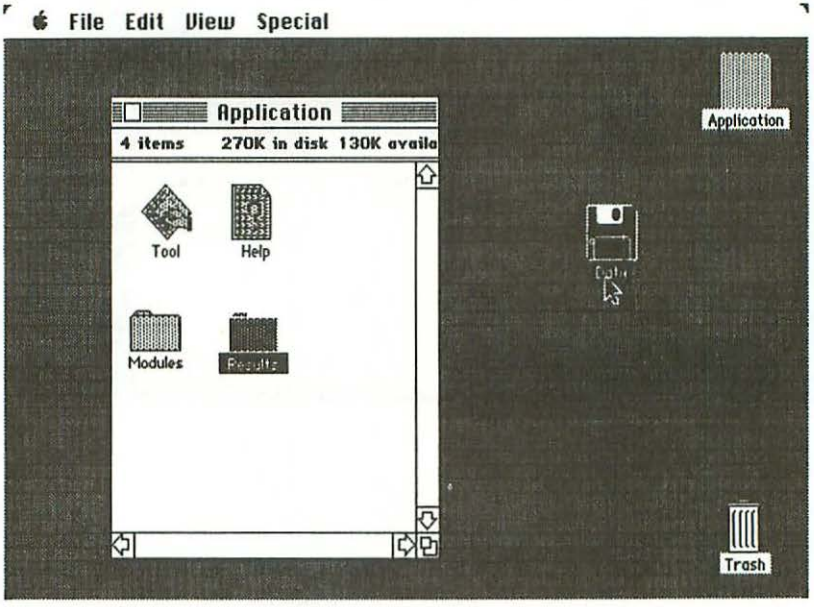
In a way, you might say that Macintosh considers that the "Application" disk and its contents are *virtually present* in the machine, even though the disk has been physically removed, temporarily, from the drive, and replaced by the "Data" disk. If ever you had turned off the power to Macintosh after ejecting the "Application" disk, it would *not* have been possible to turn the power back on and insert the "Data" disk. If you try to do this, Macintosh will display an image of a growling face, because the machine is not satisfied until it has "digested" a startup disk.

Double-click the "Data" icon. Its directory window — which is completely blank — will appear on the screen, as shown in the following illustration:



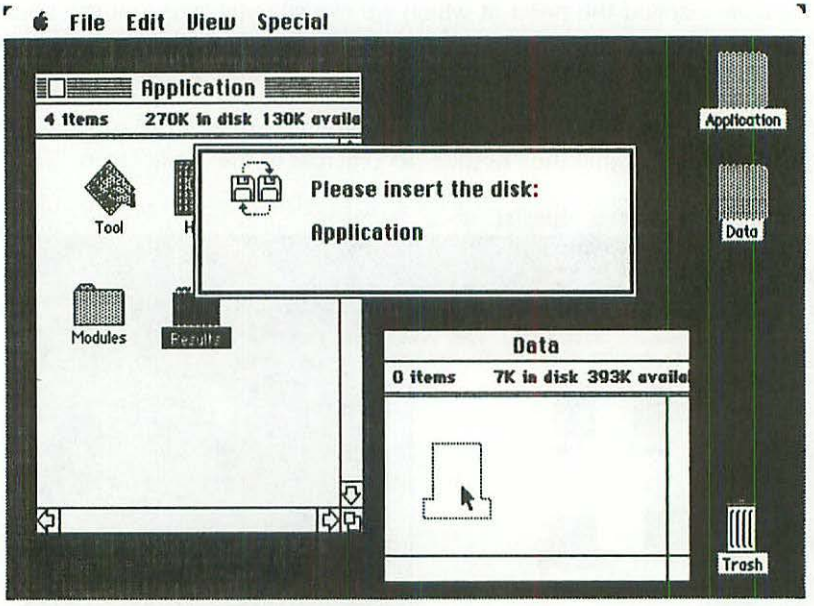
We have now reached the point at which we can actually carry out the transfer of the "Results" folder and its contents from one disk to the other. As before, when inserting the three new documents into the folder, there are two ways of going about this task.

The first method (see following illustration) consists of closing the "Data" window and then dragging the "Results" icon across to the "Data" icon.



The second method consists of leaving the "Data" window open on the desktop and then dragging the "Results" icon over onto this open window. As soon as you release the mouse button, Macintosh does several things. First, it informs you that three files (the number of documents in the "Results" folder) have to be copied from one disk to the other. Next it pops the "Data" disk out of the drive. And finally it displays a message, as in the following illustration, requesting you to insert the "Application" disk in the drive. Don't be disturbed by the curious little icon in the message; it's merely a visual aid designed to inform you that one disk has to be removed from the drive, and another inserted in its place.

At this point, Macintosh starts to behave in a manner that invariably surprises newcomers to the machine. It has just popped out the "Data" disk and requested you to insert the "Application" disk. Well, once you do as told, Macintosh immediately pops out the disk you just inserted, and asks you to put back the "Data" disk. No sooner do you do this than the machine pops out the "Application" disk and asks you to reinsert the "Data" disk. And Macintosh carries on like this for two more rounds, until it finally settles down — apparently satisfied — with the "Application" disk in its drive.

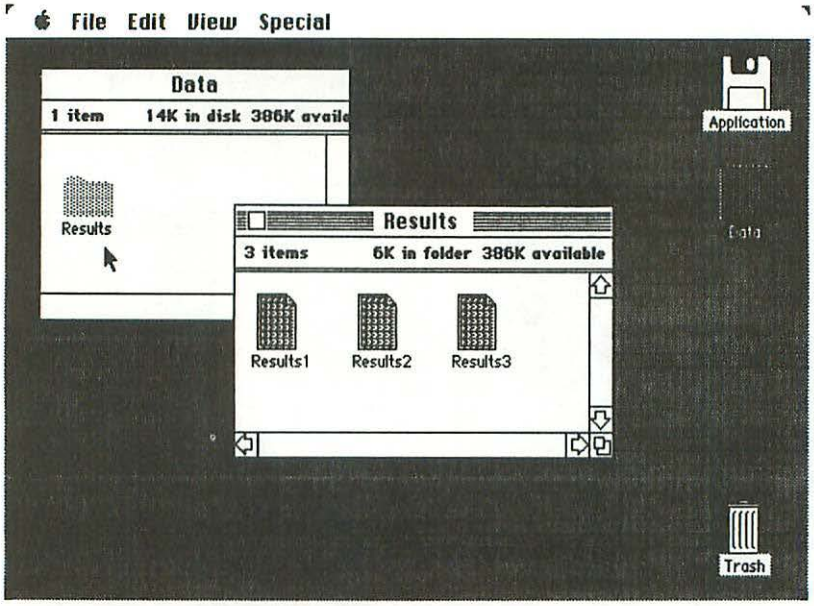


People who see Macintosh acting in this way for the first time generally believe that the machine has run amok, and that it can no longer make up its mind about which disk it wants. But the truth of the matter is simply that Macintosh is obliged to carry out its disk-to-disk information transfers in a series of relatively short steps, moving the data block by block. Don't forget that Macintosh is basically a single-disk machine with a main memory of 128K. Data that is to be moved from one disk to another must transit by the main memory, which explains why the process must be carried out in a series of steps.

All this swapping of disks would be tedious on any other machine besides Macintosh, particularly if the user had to constantly open and close the gate of the disk drive. But Macintosh makes the job effortless, because it physically ejects the disk to be removed, and you then simply push in the other disk with a slight movement of the hand. Admittedly, at times the user feels a little like a robot, sitting in front of Macintosh and mechanically obeying the machine's instructions to insert disks into its drive. But that's a small price to pay for the reassurance that Macintosh is taking care of the copying job in a faultless manner.

(Naturally, if you happen to have a second disk drive plugged into the back of your Macintosh, then this minor problem of disk swapping simply ceases to exist.)

Once the transfer of the "Results" folder is completed, you can double-click the icon in the "Data" window to verify that the three documents have in fact been copied onto the data disk. The result is shown in the following illustration:



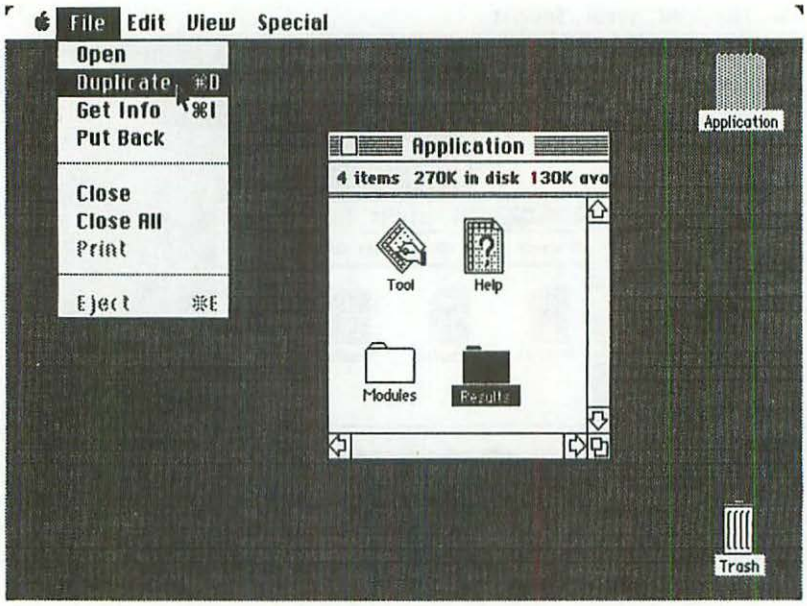
If you now open the "Application" window on the desktop, you'll see that the original "Results" folder and its contents are still in fact present on the startup disk. In other words, the process that we've been describing as a disk-to-disk *transfer* of information is in fact a *copy* operation. Remove the original folder on the startup disk by dragging it to the trash can.

Instead of copying a whole folder of documents from one disk to another, we could use this same technique for copying a single document such as "Results1", or an application program such as "Tool".

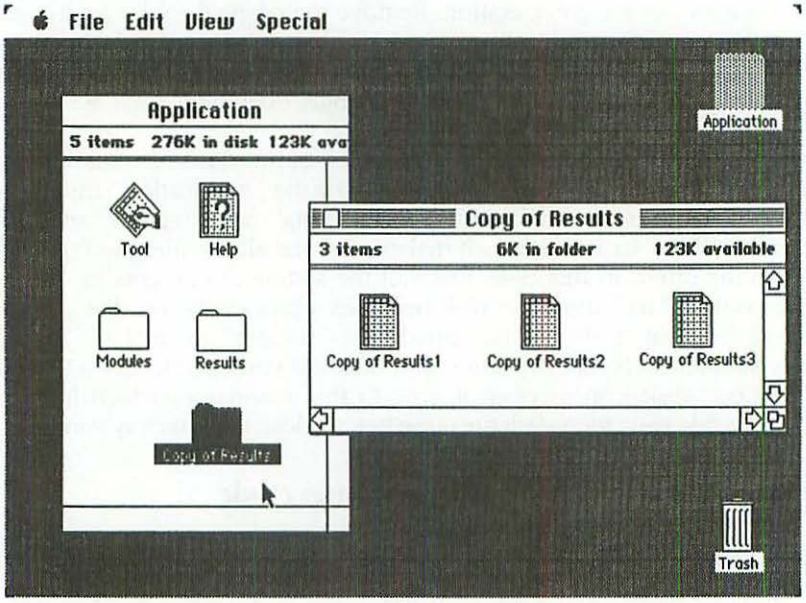
The same technique can be used to copy an entire disk. For example, if you want "Data" to contain an entire backup copy of the "Application" disk, then all you have to do is to eject "Application", insert "Data", and drag the "Application" icon onto the "Data" icon. Macintosh makes sure that all the files are copied from one disk to the other. In this case, since all the system documents in "Modules" are copied onto "Data", the latter disk becomes a proper startup disk . . . and so maybe you feel that it should be called, say, "Backup" instead of "Data". It's extremely simple to change the name of a disk. All you have to do is to double-click the "Data" disk icon to open it, type in the new name (which immediately appears in the title bar), then click the close box to close the directory window.

Copying documents on the same disk

As a final example of an important housekeeping feature of the Finder, let us look at the **Duplicate** command in the File menu, as shown in the following illustration:

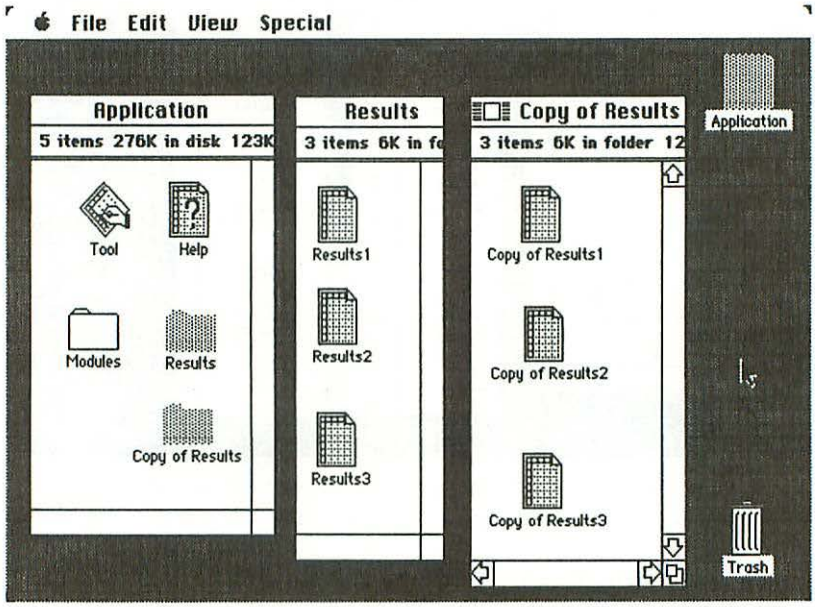


Since the "Results" icon has been selected, it is this folder and its contents that will be duplicated, as shown in the following illustration:



Notice that the prefix "Copy of" has been added to each new icon name.

We can even open the original folder and the copy, simultaneously, on the desktop, as shown in the following illustration:

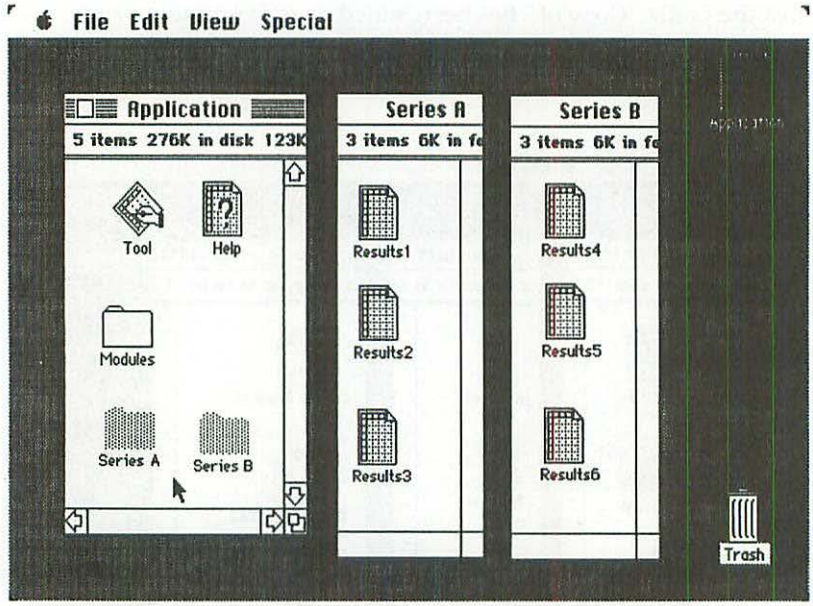


We have already seen (page 68) that it is very easy to change the name of a folder. In the same way, you can easily change the name of a document such as "Copy of Results1". To do so, move the pointer — which assumes the form of an I-beam — just to the right of the current name. Next, click the mouse button and use <Backspace> to delete the current name. Then, type in the new name. The operation is terminated by moving the pointer away from the new name, and clicking the mouse button.

In other words, we can create any number of folders we like, simply by duplicating existing folders, replacing their names, and changing their contents (which may involve throwing the old contents into the trash can).

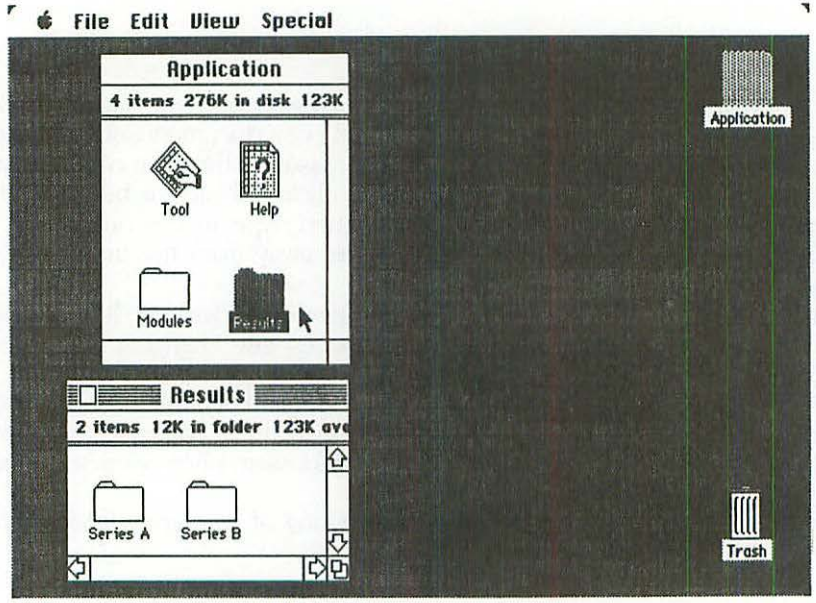
Furthermore, folders can contain, not only documents, but . . . other folders. After all, that's how we organize information in real-life contexts. In a lawyer's office, for example, we might discover that a big dossier, once opened, contains a pile of smaller folders, and so on.

Here's an example (see following illustration) of two open folders labeled "Series A" and "Series B".

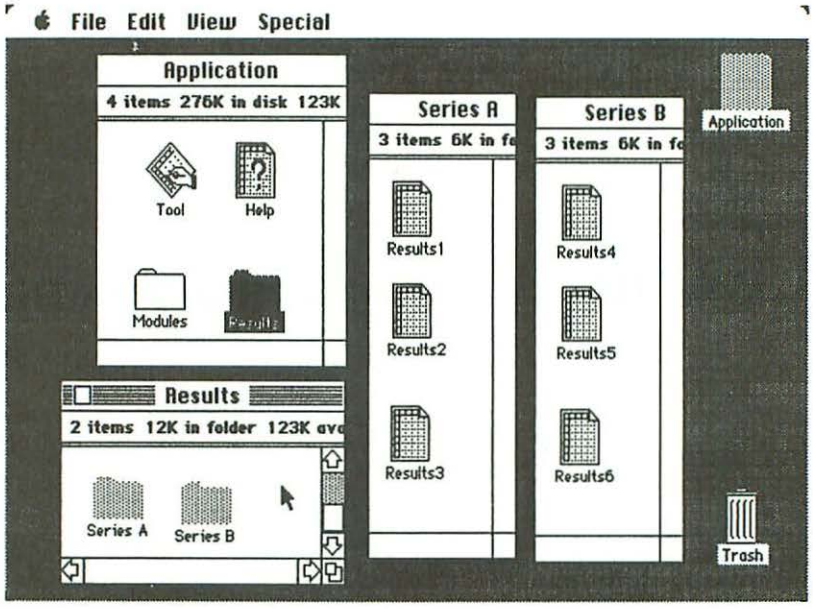


The folder named "Series A" contains three documents: "Results1", "Results2" and "Results3". Likewise for the folder named "Series B": "Results4", "Results5" and "Results6".

Let us suppose now that we store away these two folders in a big folder labeled "Results", as shown in the following illustration:



Nothing prevents you from opening simultaneously *all* these folders on the desktop, as shown in the following illustration:



So, this Macintosh housekeeping concept — involving documents, folders, copy functions and a removal function (the trash can) — is not only powerful and easy to use, but also foolproof. It's most unlikely that you'll ever lose an important document by inadvertence, because the presence of the icons on the desktop allows you at all times to actually *see* what objects you're working with.

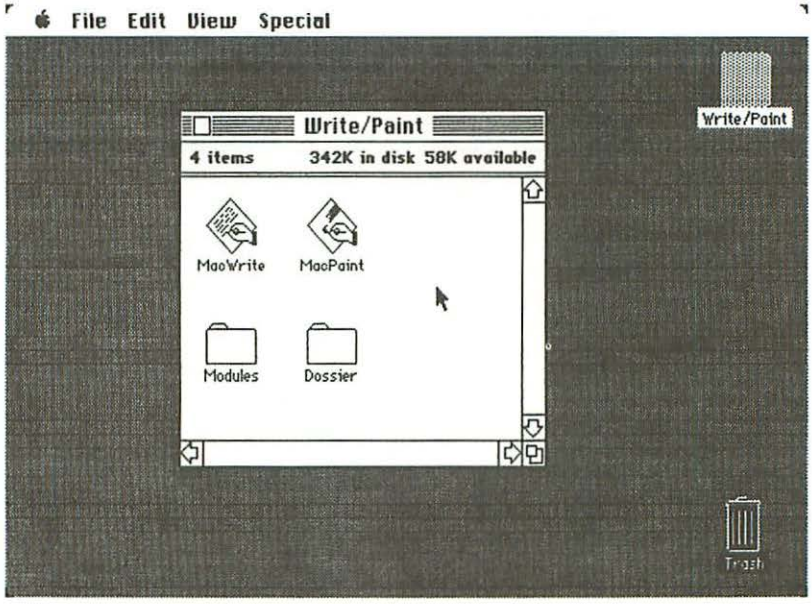
chapter 5
"MacWrite" — Word Processing

Word processing, today, has become one of the most popular applications of microcomputers, and so it is not surprising that *MacWrite* was one of the first major programs developed for Macintosh.

The design of MacWrite has been based upon two avant-garde word processing concepts:

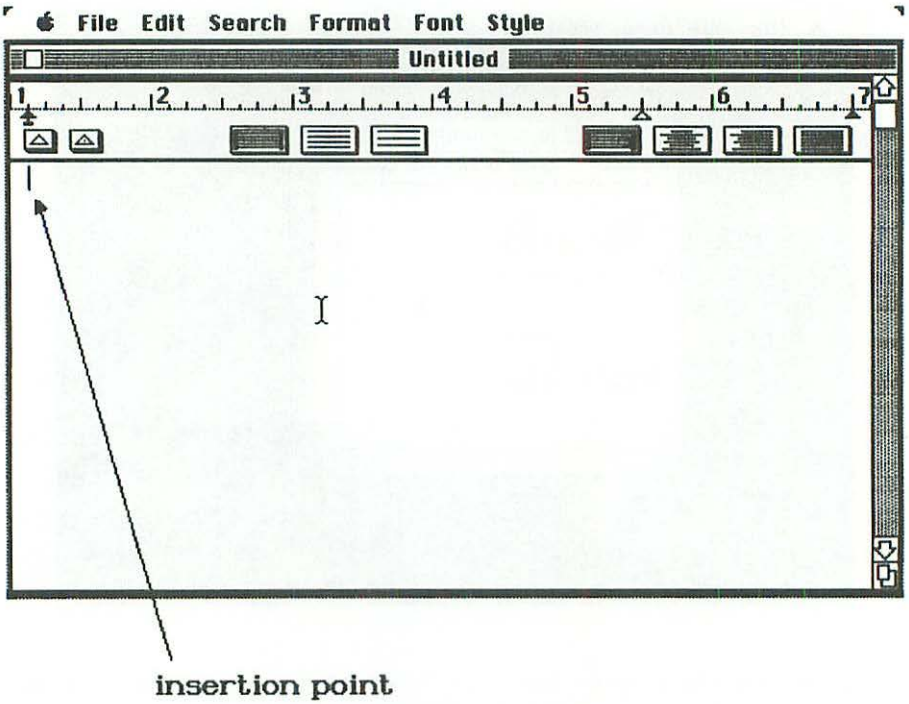
1 When you are creating a document with MacWrite, its appearance on the Macintosh screen is exactly the same as it will look once it has been printed on the Imagewriter printer. This means that there are never any unpleasant surprises when you ask Macintosh for a hard copy of what you see on the screen. A corollary of this essential fact is that the text displayed on the screen does not contain any of the embedded commands that burden most word processing systems.
2 The screen on which MacWrite documents are created is essentially a *graphic* device, because of the bit-mapped display mode used by Macintosh. Now, MacWrite utilizes this property of the screen in two spectacular manners. On the one hand, for text, you are offered a wide range of type styles and fonts. On the other hand, when the text needs to be enhanced with graphic material such as illustrations and charts, these can be created by means of MacPaint and/or MacDraw, and then incorporated effortlessly into MacWrite documents.

Both MacWrite and MacPaint are generally housed together on the same disk, called "Write/Paint", whose directory window might appear as in the following illustration:



Here again, as in our previous examples, "Modules" is a folder containing half-a-dozen software components that enable "Write/Paint" to function as a startup disk, whereas "Dossier" is an empty folder that can be used later on for holding documents created by either MacWrite or MacPaint.

Double-click the icon labeled "MacWrite" to start the program running. After several seconds, the screen will appear as in the following illustration:

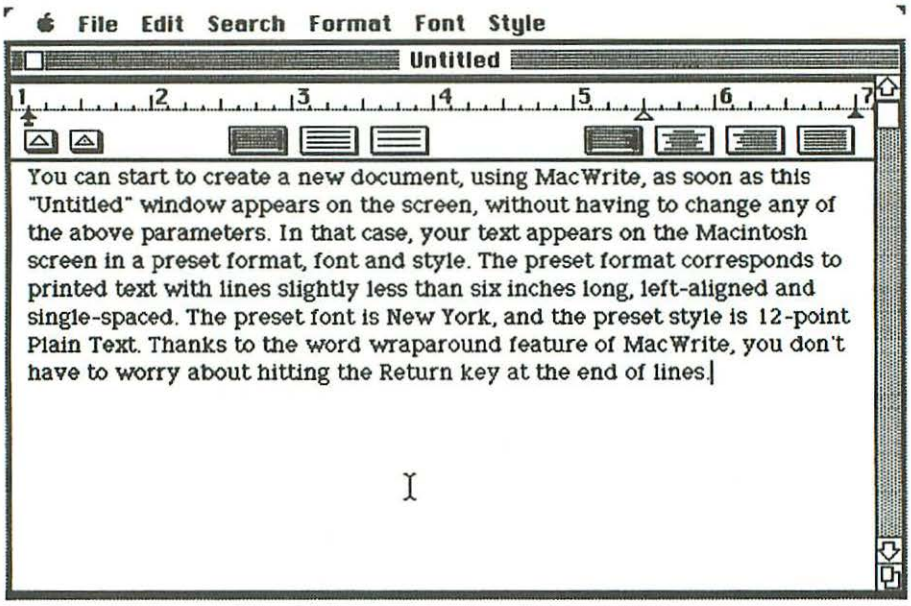


If the window is labeled "Untitled", this is merely to indicate that it is a new MacWrite window that does not yet contain any information. Naturally, as soon as we start to create a document in the window, one of the first things we'll do is to change the term "Untitled" into a meaningful title for our document.

Notice that there is a blinking vertical bar that indicates, at all times, the *insertion point* for the character that you are about to type.

There is also an *I-beam* pointer on the desktop, and you can click it to change the insertion point.

Now, MacWrite has the reputation of being an extremely "friendly" word processing tool . . . and one of the keys to that friendliness is the fact that you can start to use it immediately, as soon as this window appears on the screen, without having to get involved with any of the menus or strange-looking symbols up at the top of the screen. So, let us do exactly that, by typing the text that is shown in the following illustration:



MacWrite has a *word wraparound* feature, which means that you don't have to worry about hitting the <Return> key at the end of lines. As soon as MacWrite realizes that the word being typed cannot fit into the room left at the end of the current line, the entire word jumps to the beginning of the following line. Nothing prevents you, though, from using the <Return> key if you particularly wish to start a new line before reaching the right margin of the current line.

The fact that we have paid no attention to the things up at the top of the screen, when entering this short text into the machine, merely means that MacWrite has made use of a number of *preset parameters* to govern the way in which our text appears in the window.

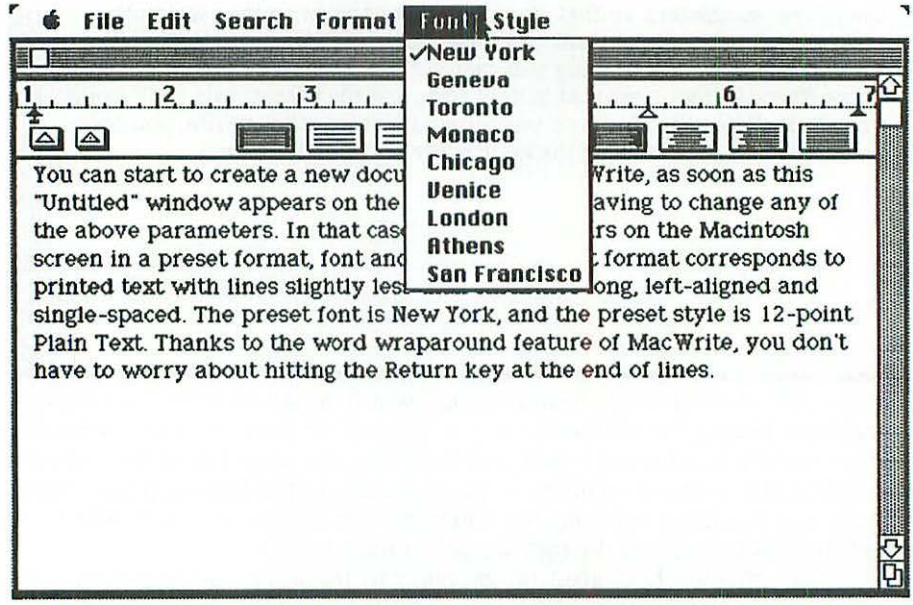
First, there are the parameters dealing with the *format* of the text. The preset format corresponds to printed text with lines slightly less than six inches long, left-aligned and single-spaced. Just below the word "Untitled" there is a scale marked in inches from 1 to just over 7. If you look closely, at either end of this scale, you'll see tiny black arrowheads. These are the left and right *margin markers*, and you can check that they are just under six inches apart.

Next, look at the group of three rectangular striped boxes below the figure 3. These are the *space boxes*. Notice that the one on the left is highlighted by a black background. It's the single-space box.

Look now at the group of four rectangular striped boxes below the figure 6. These are the *alignment boxes*. Here again, the one on the left is highlighted by a black background. It's the left-alignment box.

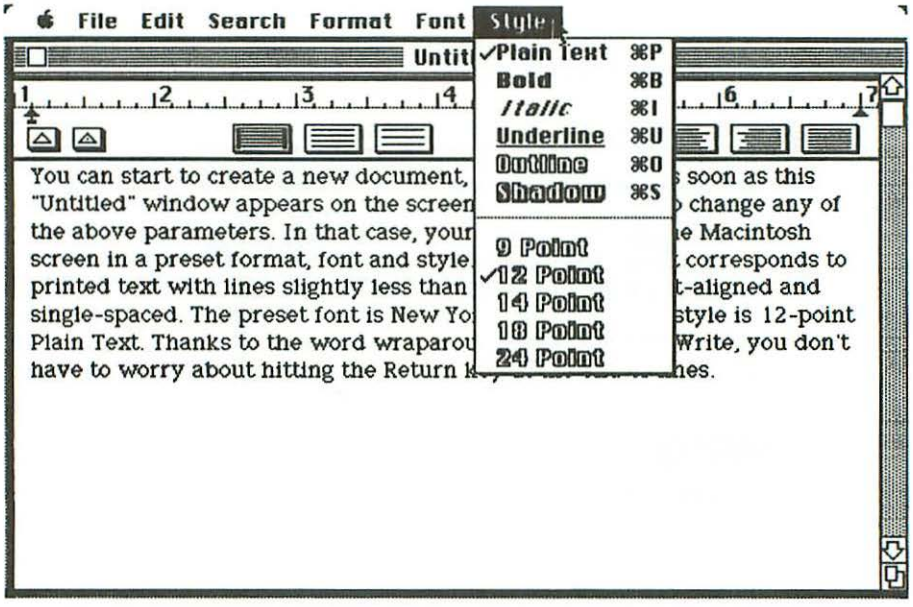
This entire section — including the scale in inches, the margin markers, the various boxes, etc. — is referred to as a *ruler*, and we shall be returning to this important format device later on.

Another preset parameter in the text that we just typed is the *font*. If you pull down the **Font menu** (see following illustration), you can verify that the name of the preset font — the one with the tick alongside — is New York.



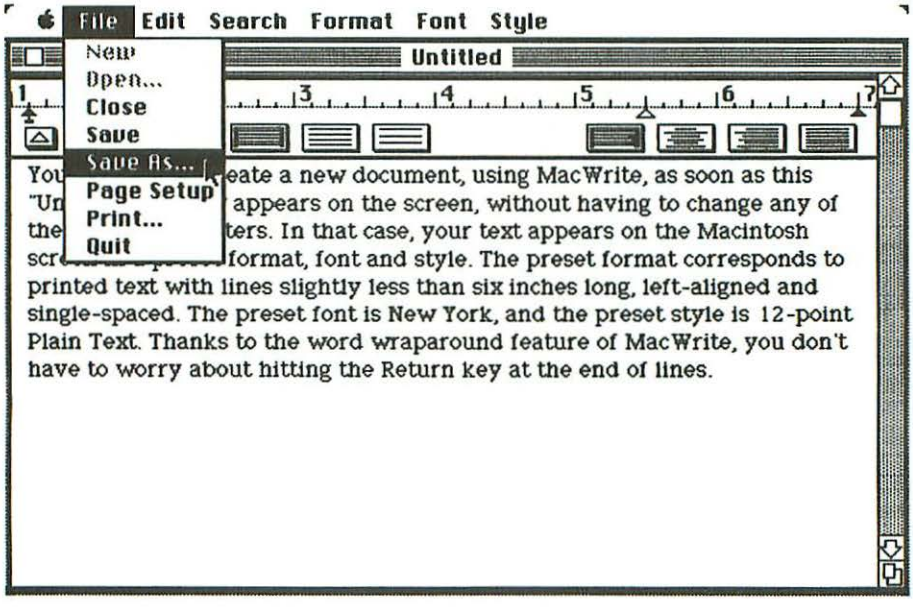
There are eight other fonts we could choose from, but let's not worry about them for the moment.

The remaining preset parameters concern the *style* of our text. If you pull down the **Style menu** (see following illustration), you can verify that the preset style is 12-point Plain Text.

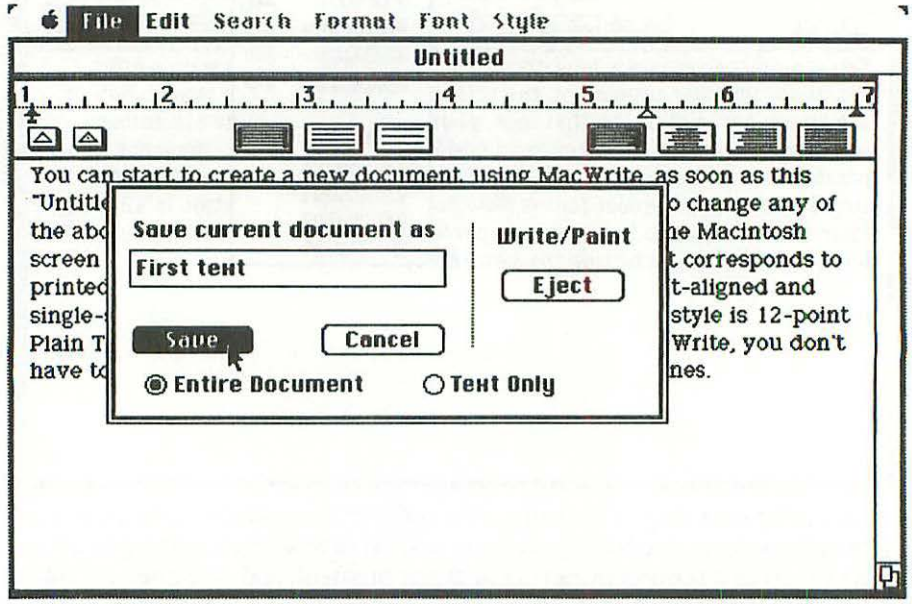


Here again, let's not stop to investigate the other style possibilities, because a more urgent task awaits us: we have to *save* our text on disk, with a meaningful title. (An accident such as a power cut can occur at any moment, and one must get into the habit of regularly saving new documents: once every twenty minutes or so.)

To do this, we pull down the File menu, as indicated in the following illustration, and choose the Save As command:



At this point, Macintosh reacts with a new type of message (see following illustration) known as a *dialog box*, because the machine needs to communicate with the user in order to obtain several elements of vital information.



Since we are saving this document for the first time, Macintosh asks us to supply a name, to be typed in the wide rectangle inside the dialog box. Let's call our document "First text".

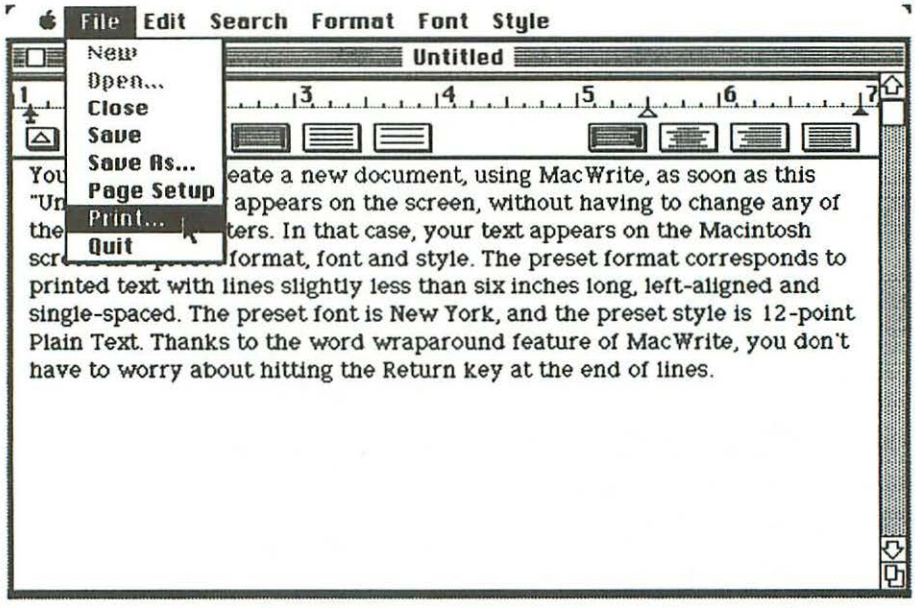
The three round-cornered rectangles inside the dialog box are referred to simply as *buttons*, whereas the two small circles down the bottom are called *radio buttons* (pressing one releases the other). The general idea is that we can dialog with Macintosh simply by using the hardware button on the mouse to click various software buttons in the dialog box.

In this particular dialog box, the **Entire Document** radio button is preset, which is what we want. The button that interests us, of course, is **Save**, which is highlighted as soon as we click it. If we had clicked the **Cancel** button, Macintosh would have reacted just as if we had never chosen the Save As command in the File menu. In other words, the **Cancel** button can be used to move back to a previous state if ever we happen to choose the wrong command.

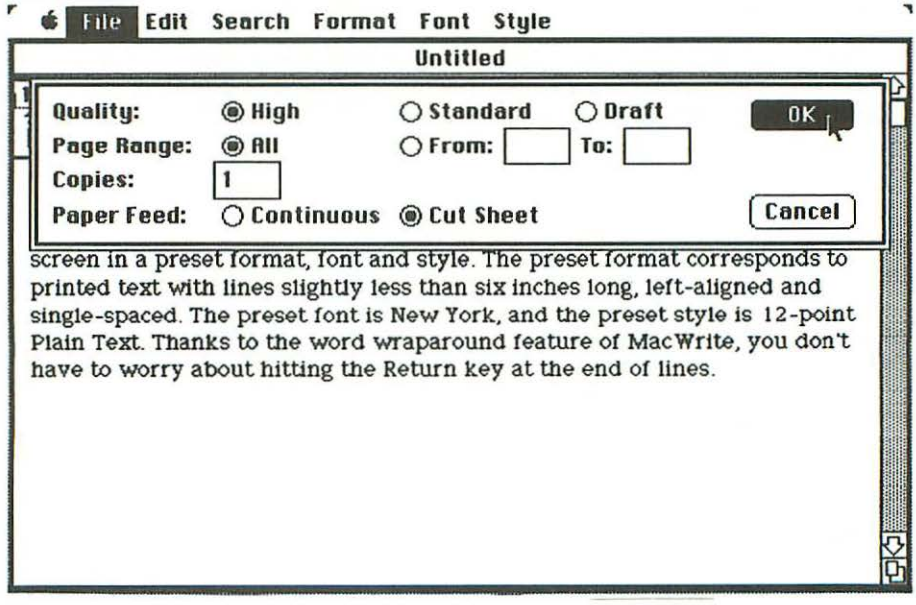
As for the **Eject** button, it can be clicked whenever you want to save your document, not on the "Write/Paint" startup disk, but on some other disk: maybe a data disk with no application software, only documents.

Printing using the Imagewriter

Now that our "First text" is safely stored on the "Write/Paint" disk, maybe we can think about getting a hard copy of it. This is achieved by pulling down the File menu and choosing the **Print** command, as indicated in the following illustration:



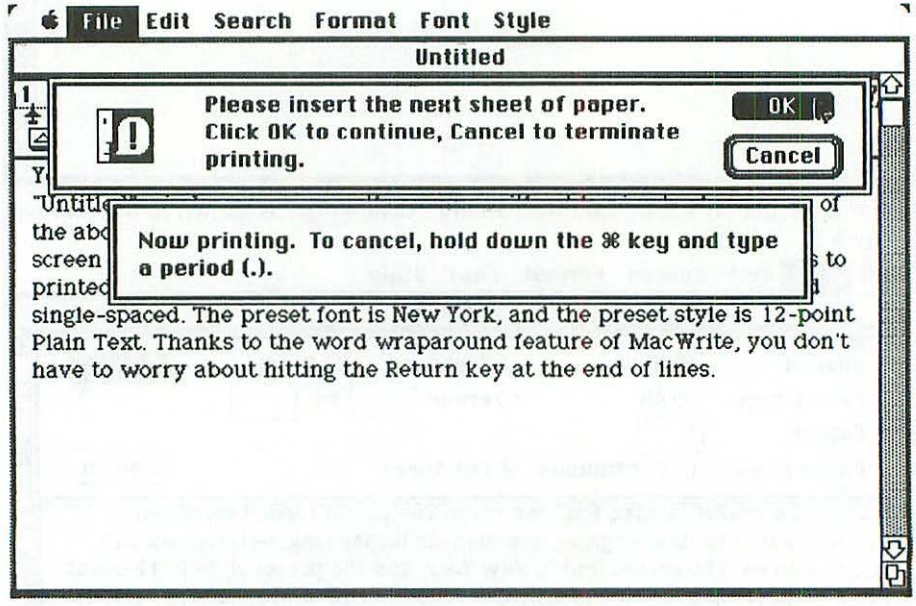
We are immediately confronted with another dialog box, as shown in the following illustration:



There are radio buttons and boxes for four different aspects of the printing operation that is about to take place:

1 Quality: **High** quality printing on the ImageWriter is elegant but slow. **Standard** quality, faster than **High** quality, is the preset option. It provides the same degree of resolution as the Macintosh screen. **Draft** quality, the fastest, can only be used for text.
2 Page Range: **All**, the preset option, indicates that the entire document is to be printed. Otherwise, click the **From/To** button and indicate the bounds of the part of the document that is to be printed.
3 Copies: The preset value is 1.
4 Paper Feed: Click the **Cut Sheet** option if you intend to hand-feed single sheets of paper into the Imagewriter. The **Continuous** option, which is preset, concerns fanfold stationery.

In our case, we have asked for a single high-quality copy of our document on a normal sheet of paper. Clicking the **OK** button tells Macintosh that we've selected all the appropriate options in the dialog box, and that we would like the printing to start. But Macintosh (see following illustration) is not the sort of creature who likes to rush things. . . .

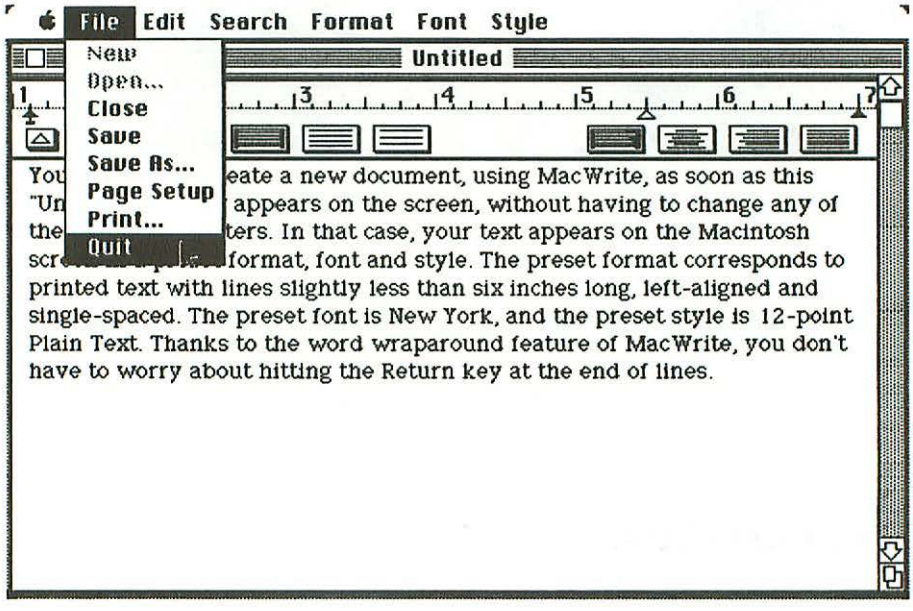


Two more dialog boxes appear on the screen. In the upper one, the little icon with a human face and an exclamation mark reminds us that we have to put a sheet of paper in the Imagewriter. Click **OK** to let the machine know that we've done so. Then the second dialog box provides us, in a friendly manner, with an easy means of halting the machine if ever something goes wrong during the printing process.

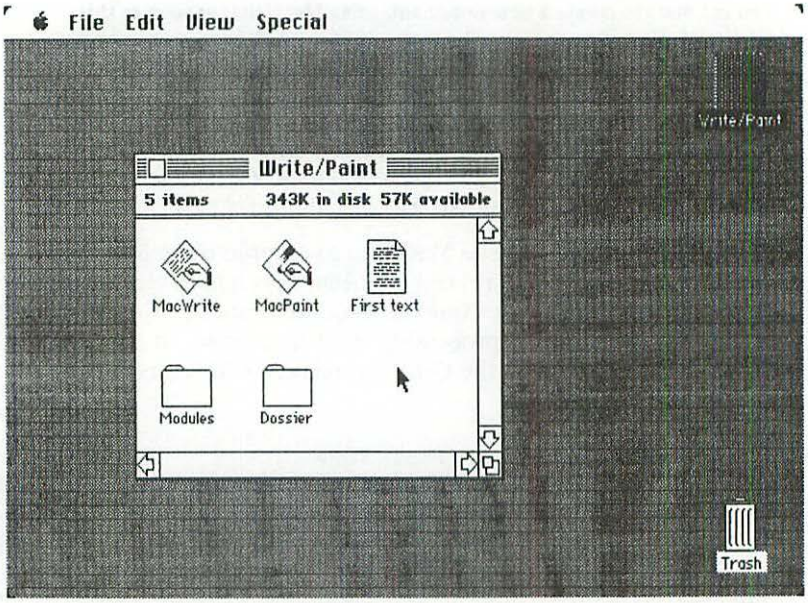
Finally we obtain the following printout of our document:

You can start to create a new document, using MacWrite, as soon as this "Untitled" window appears on the screen, without having to change any of the above parameters. In that case, your text appears on the Macintosh screen in a preset format, font and style. The preset format corresponds to printed text with lines slightly less than six inches long, left-aligned and single-spaced. The preset font is New York, and the preset style is 12-point Plain Text. Thanks to the word wraparound feature of MacWrite, you don't have to worry about hitting the Return key at the end of lines..

At this point we have therefore used MacWrite to complete the entire sequence of events from the typing of a text up until its printout as a hard copy, and including its safe storage on disk. Maybe we could consider that our day's work is done, and that we can set aside our word processing until tomorrow. In that case, we pull down the File menu and choose the Quit command, as indicated in the following illustration:



The text disappears from the screen, and the directory window of the "Write/Paint" disk reappears on the desktop, as shown in the following illustration:



Notice that there is a new icon in the window, named "First text": the MacWrite document that we have just created.

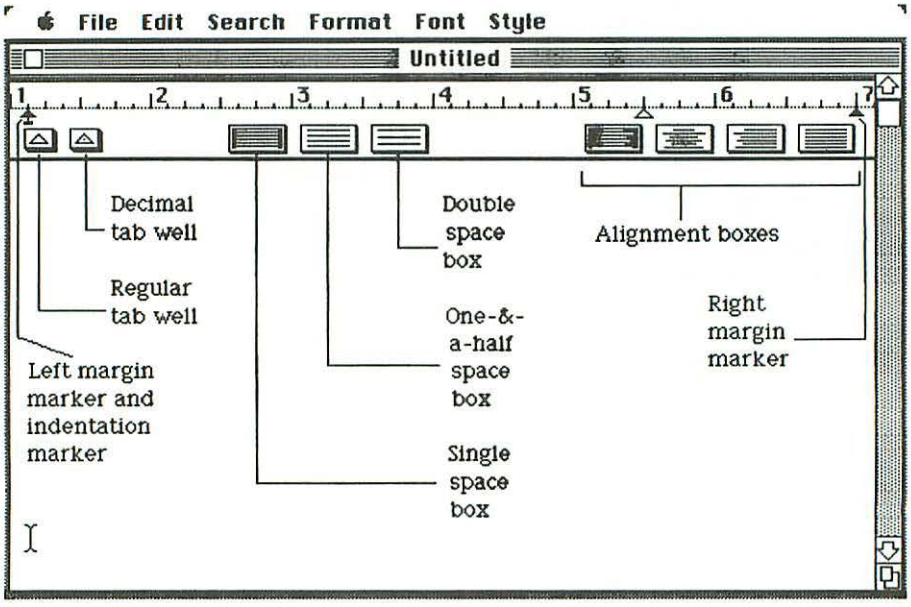
To physically eject the "Write/Paint" disk from the drive, pull down the File menu — as usual — and choose the Eject command.

Format modifications

In order to discover the possibilities of MacWrite, let us imagine that, the following day, we decide to carry out a whole series of modifications on the text that we have just written.

It's best to start out with *format modifications*. They are the easiest to understand, in the sense that they alter merely the *form* of the document, but not its *content*.

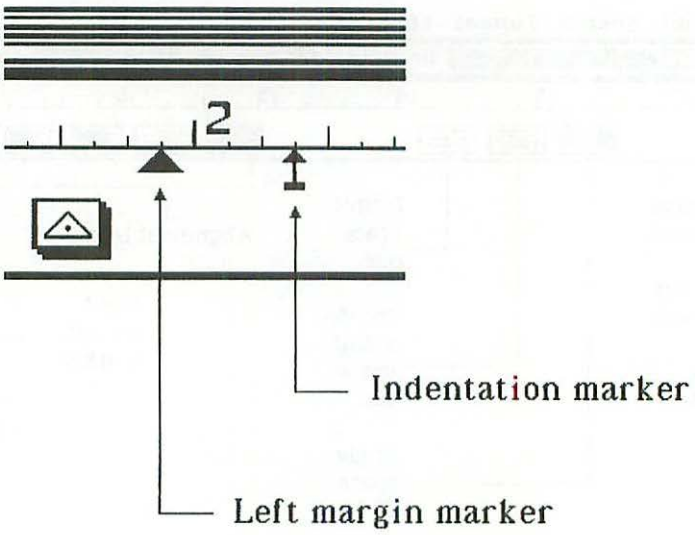
The following illustration identifies most of the format devices contained in a MacWrite *ruler*, some of which have already been mentioned:



MacWrite and the Imagewriter are designed to function with standard-sized paper, 8½ inches wide. The ruler scale starts at 1 and ends just after 7 because this provides for a left margin of about an inch and a right margin of about an inch and a half. (These margin widths are only meaningful in the case of continuous-feed fanfold stationery, held in a fixed position by the Imagewriter pins. Single sheets of paper can of course be placed at any position along the platen.) In any case, it is impossible to print a line wider than 6⅜ inches. Since the left and right margins are preset respectively at 1⅛ and 7 inches, this means that the preset width of a line is 5⅞ inches.

To vary the line width, either of the little black arrowheads in the ruler can be dragged along the scale by means of the mouse. But if you look carefully at these two symbols in the preceding illustrations, you'll notice that they're not identical. In fact, the one on the left is a *double* symbol, composed of two separate parts that can be dragged along the scale in an independent manner. The triangular part at the top is the actual *left margin marker*, and it hides a tiny arrow which is the *indentation marker* for new paragraphs. We shall see examples of the use of these markers in a short while.

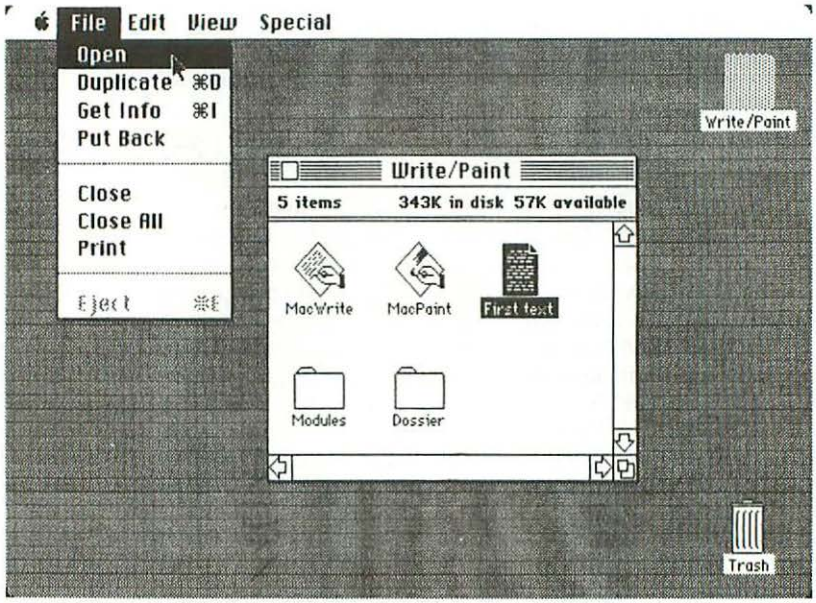
The two *tab wells* contain a stock of tabs that can be dragged out and placed underneath the scale. Notice that there is a preset tab at the 5½ inch mark. The tabs from the well on the left work for text, whereas those from the well on the right (which have a dot inside the triangle) function for numbers that have to be aligned with respect to their decimal points. To get rid of a superfluous tab, you merely drag it downwards and it disappears into thin air.



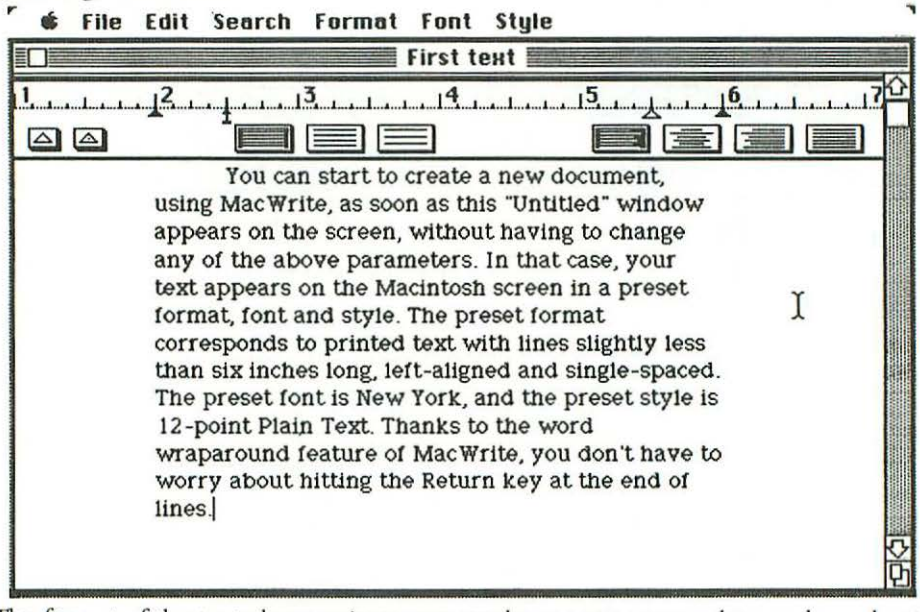
The three *space boxes* provide different spacings between lines, as indicated. You select the desired spacing by clicking the corresponding box, which is immediately highlighted in black.

The *alignment boxes* offer four possibilities: left-alignment, center-alignment, right-alignment or full-justification.

Now, in order to try out all these different formats, let us reopen the MacWrite document called "First text". To do this, we do *not* need to touch the "MacWrite" icon, for the Open operation is carried out on the "First text" icon itself. We either select the "First text" icon and choose the Open command from the File menu (see following illustration), or else we simply double-click this same icon. To understand why this is so, you have to get back to the intuitive notion of *objects* and *icons* that pervades the whole Macintosh concept. At the beginning, when we wanted to create "First text", the object that we needed was the tool called "MacWrite", and so that's why we opened that particular icon. But now that "First text" actually exists, *it* has become the object that interests us. It is nevertheless true that, in opening the document called "First text", we shall also gain access once again, in a transparent manner, to the MacWrite application program.

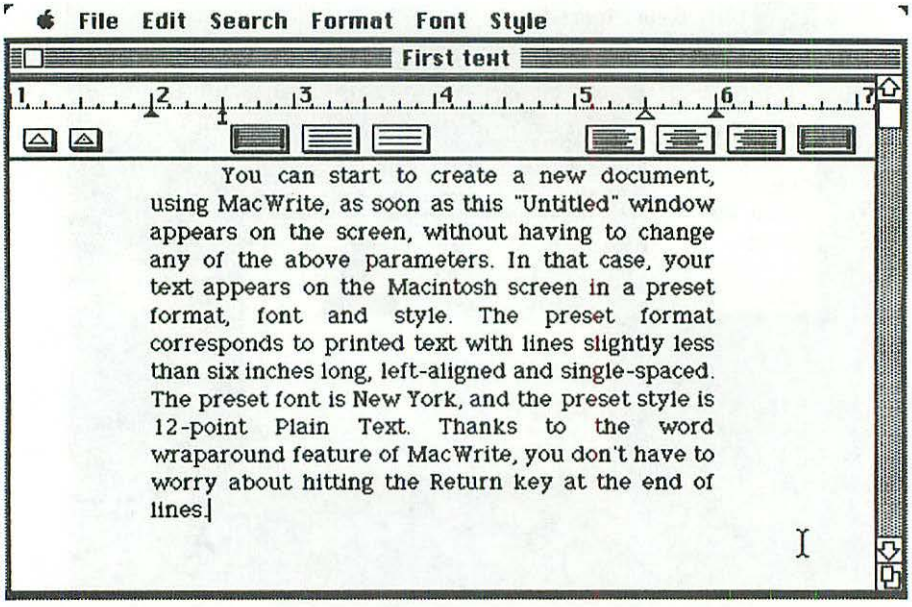


Once our familiar text is back on the desktop, let us start out by changing the position of the indentation marker and the left and right margins, as shown in the following illustration:

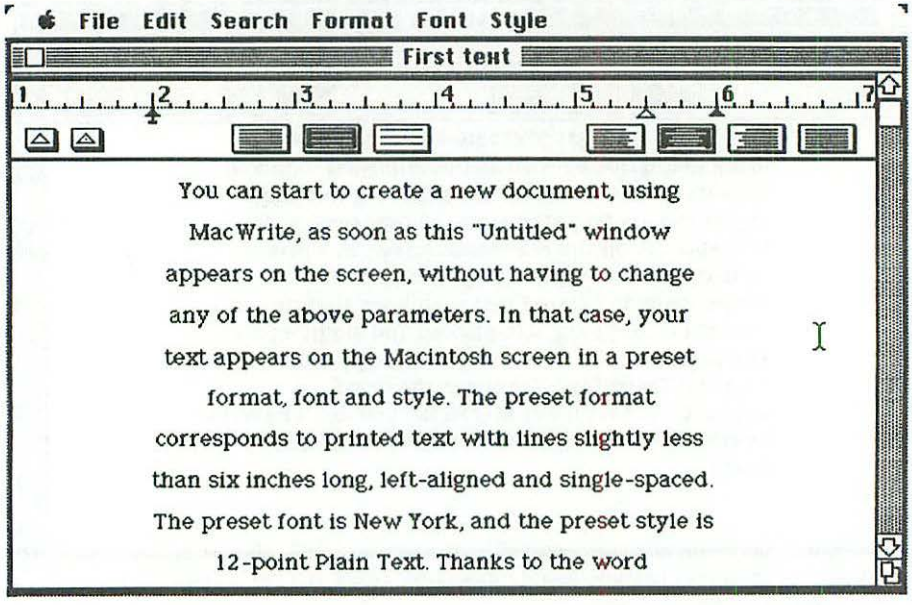


The format of the text changes instantaneously, as soon as you drag each marker to its new position along the scale.

This time, we'll keep the markers in their current positions, but we'll click the full-justification box, at the righthand end of the ruler. The result is as follows:



Finally, we'll drag the indentation marker back to the same position as the left margin, to avoid paragraph indentation, and we'll click the middle space box and the center-alignment box, which gives the following result:



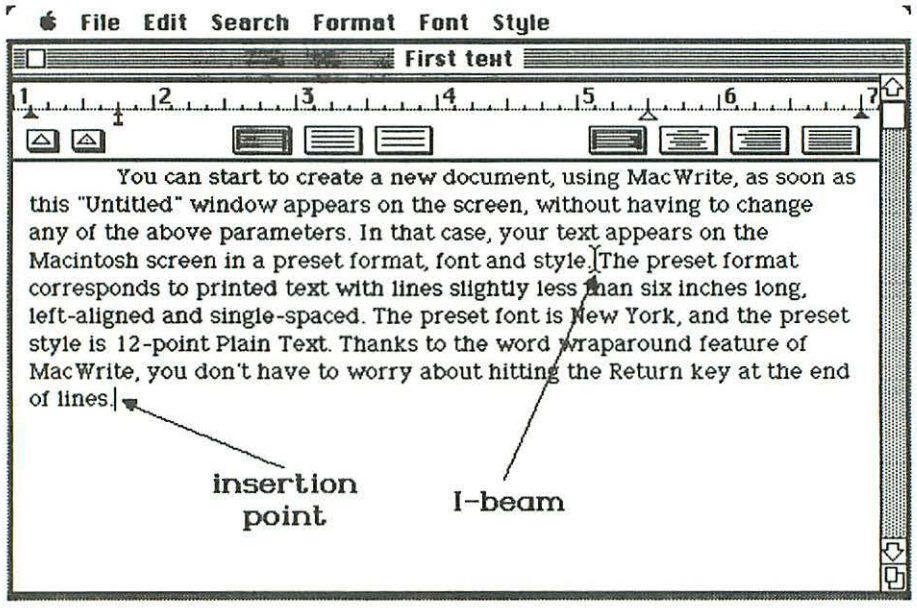
The last few lines of the text have now moved down below the lower edge of the document window, but you can access them easily by means of the vertical scroll bar.

Insertions and deletions

The MacWrite manipulations that we shall be looking at now are capable of changing the actual contents of previously-created documents.

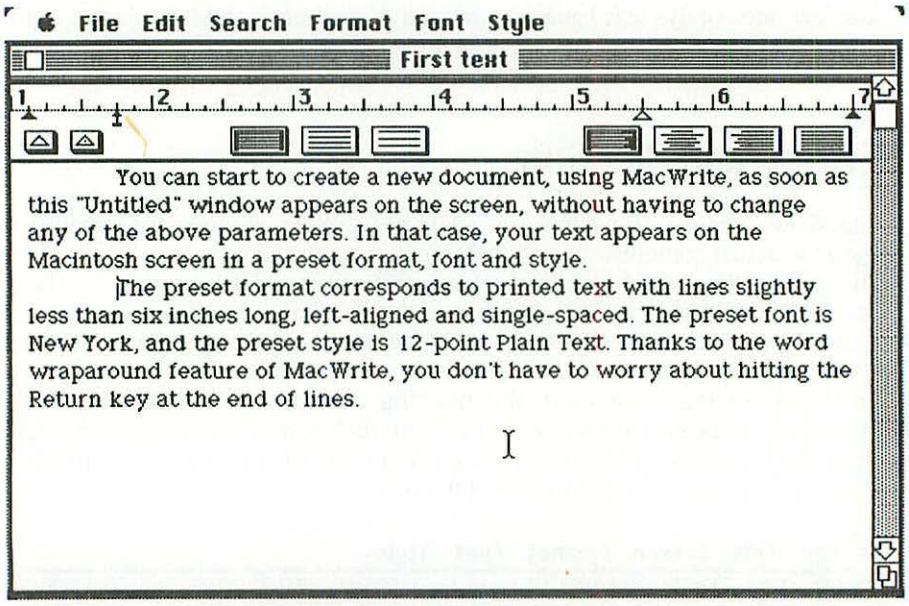
In many of the MacWrite illustrations up until now, the pointer has been present, at various places on the desktop, in the familiar form of an arrow. But, if you move the pointer onto the part of the MacWrite document window where text is entered, it changes into an *I-beam*.

In the following illustration, the blinking vertical bar that represents the insertion point is located at its normal position: that is, just after the last character that was typed (a full stop); and the I-beam pointer has been moved, by means of the mouse, to a position in the middle of the text.



If you click the mouse button, with the I-beam in this position, the blinking vertical bar will disappear from its present location and move to that spot on the screen. This means that, if you then start to type on the keyboard, the characters will be inserted at this new position, not at the end of the text.

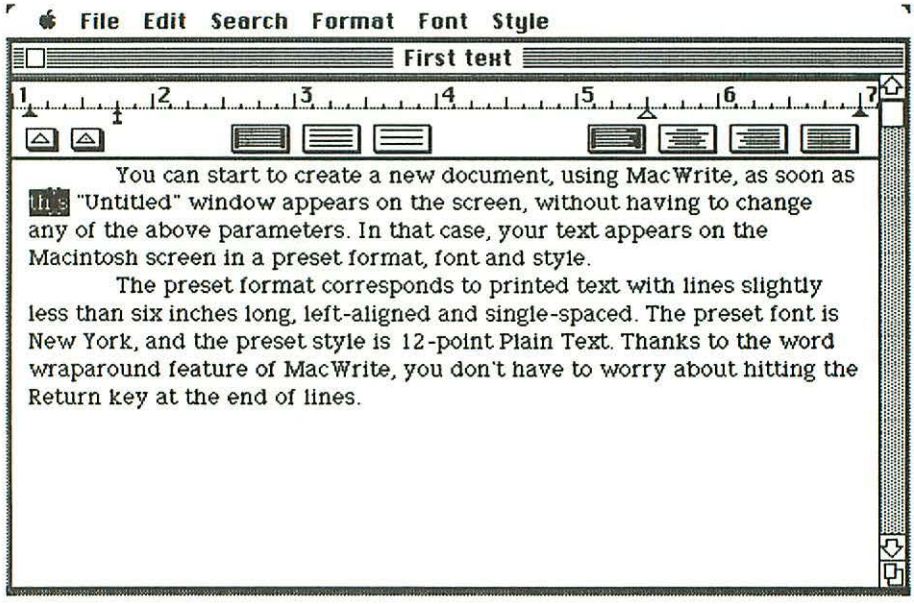
The following illustration shows what happens if we move the insertion point to this new position, and then hit the <Return> key:



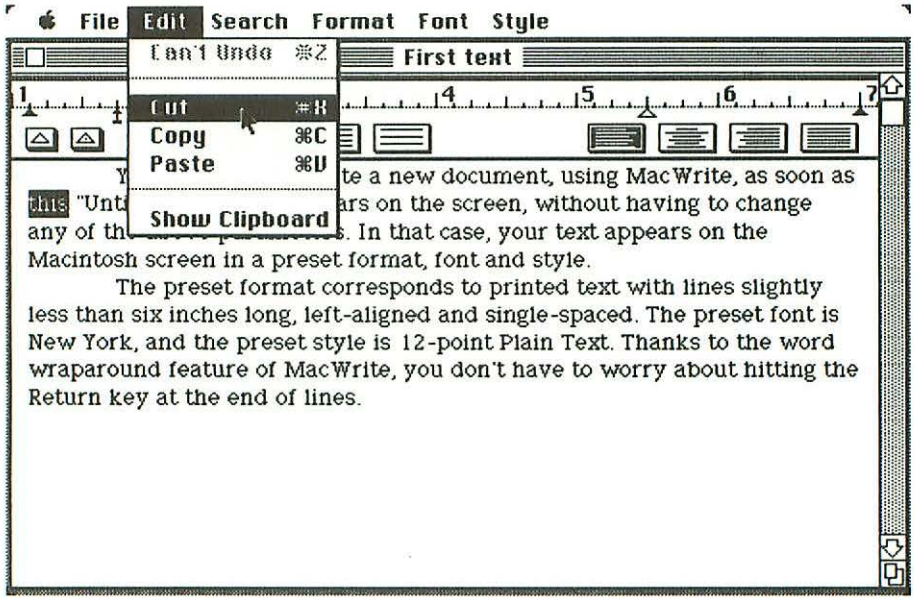
Hitting <**Return**> on the keyboard tells MacWrite that you want to start a new paragraph. The operation that we have just carried out can be described as *insertion* of a character into the existing document. Notice that the insertion point is still located just to the left of the word "The", whereas the I-beam pointer has been moved down to the lower region of the window. If you clicked the mouse button, with the I-beam in this location, the insertion point would move back to its original position, just after the final full stop in the text.

At the beginning of the second line of the text, we have used the demonstrative pronoun "this" when referring to the "Untitled" window. This is no longer coherent, since the name of the current window is "First text". So, to make the sentence comprehensible, let's carry out a *deletion* operation on the word "this", then we'll insert "the" in its place.

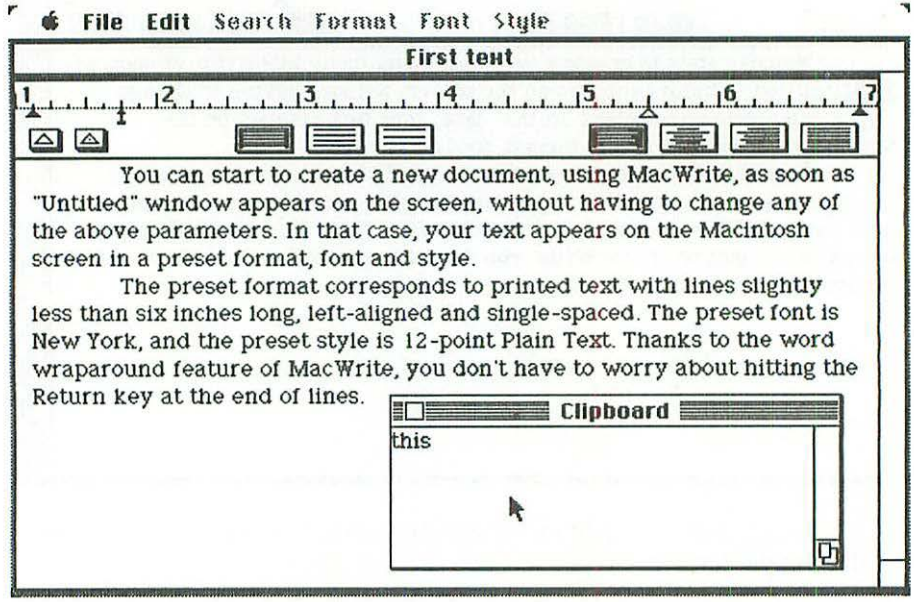
The deletion of a word is a two-step process. First, you use the mouse to position the I-beam pointer somewhere in the middle of the word to be deleted, and you double-click the mouse button. The word is immediately highlighted in black, as shown in the following illustration:



Second, you pull down the Edit menu and choose the Cut command, as shown in the following illustration:



The word "this" disappears instantly from the document (see following illustration), and the remaining text is automatically shifted to the left to fill up the hole left by the removal of the word.

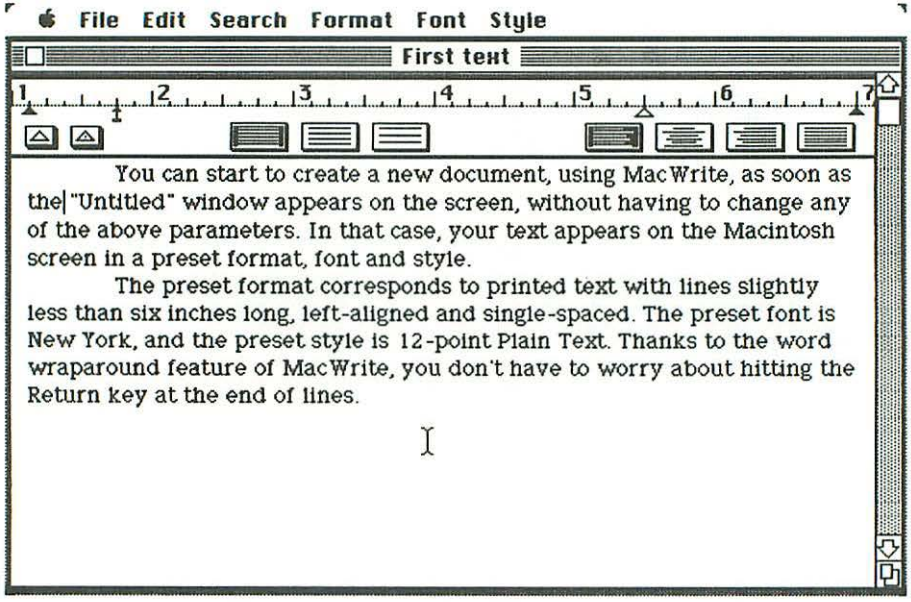


If you pull down the Edit menu and choose the Show Clipboard command, you can verify (see preceding illustration) that the Clipboard contains the word that has just been cut out of "First text".

Incidentally, it's interesting to note that there are no less than *four* ways of now removing the Clipboard from the desktop:

1 Click the Clipboard close box.
2 Pull down the File menu and choose the Close command.
3 Click the MacWrite document window alongside the Clipboard.
4 Pull down the Edit menu and choose the Hide Clipboard command.

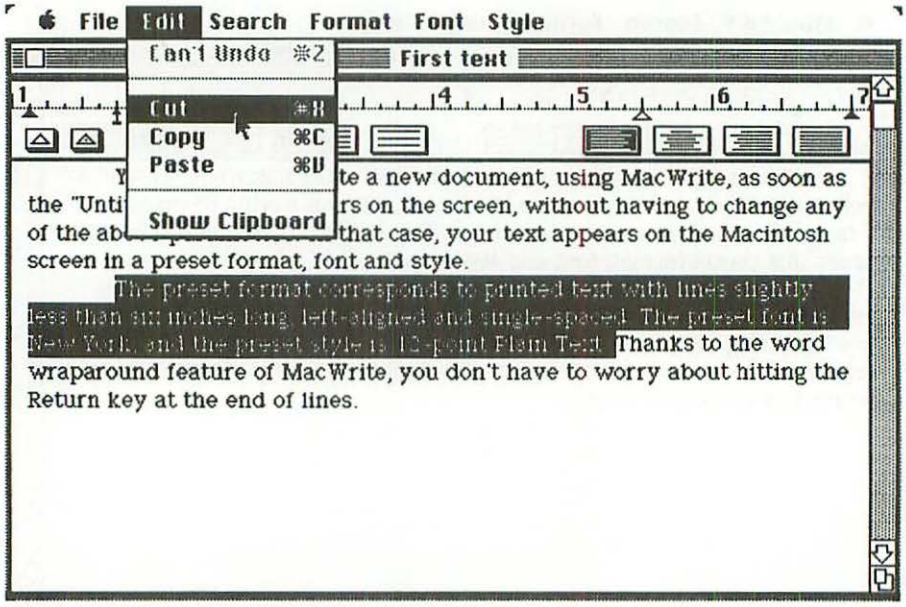
As soon as the Clipboard disappears from the desktop, and the MacWrite document window becomes active once again, we find that the blinking vertical bar reappears at the very end of the first line of our text, after the word "as". So we merely have to type in the word "the" on the keyboard, and we obtain the following result:



These two operations — deletion of "this" followed by insertion of "the" — can in fact be combined, enabling us to speak of *replacement*. Only two steps are involved:

1 Use the mouse to position the I-beam pointer somewhere in the middle of the word to be replaced, and double-click the mouse button.
2 Type the new word.

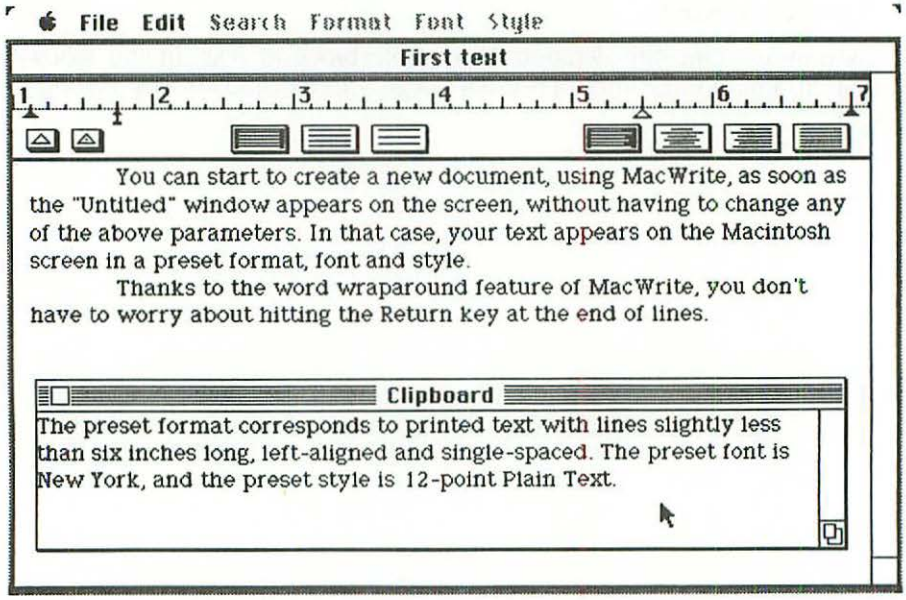
Let us now examine the deletion of a whole block of text. In the following illustration, almost three lines of text have been selected, and are now highlighted in black:



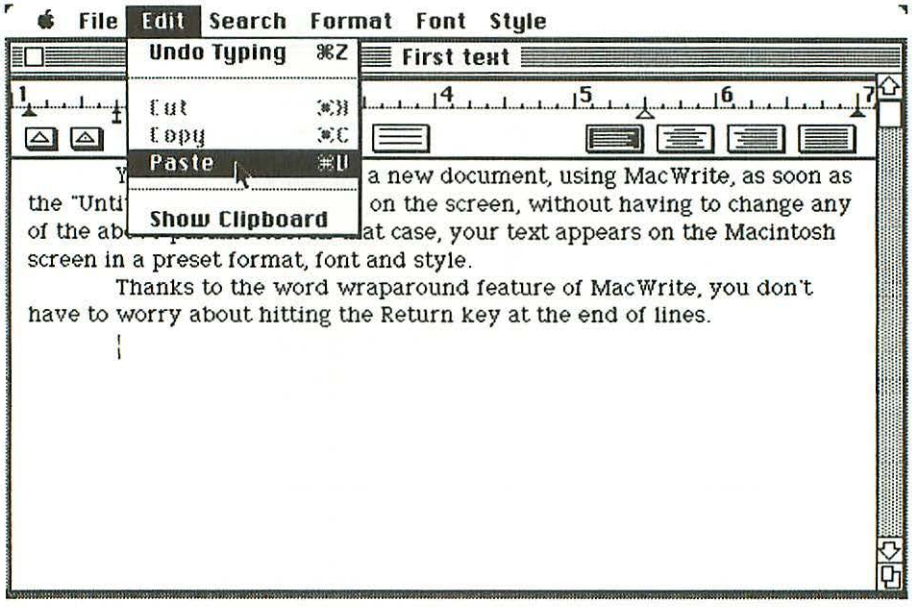
This selection is carried out — as explained already in the chapter on the cut & paste concept (page 48) — by dragging the I-beam pointer down alongside the three lines, and then across to the spot in the third line where the selection is to end.

The Cut command, from the Edit menu, is used to carry out the deletion of the selected text.

If we open the Clipboard (see following illustration), we rediscover the deleted text.



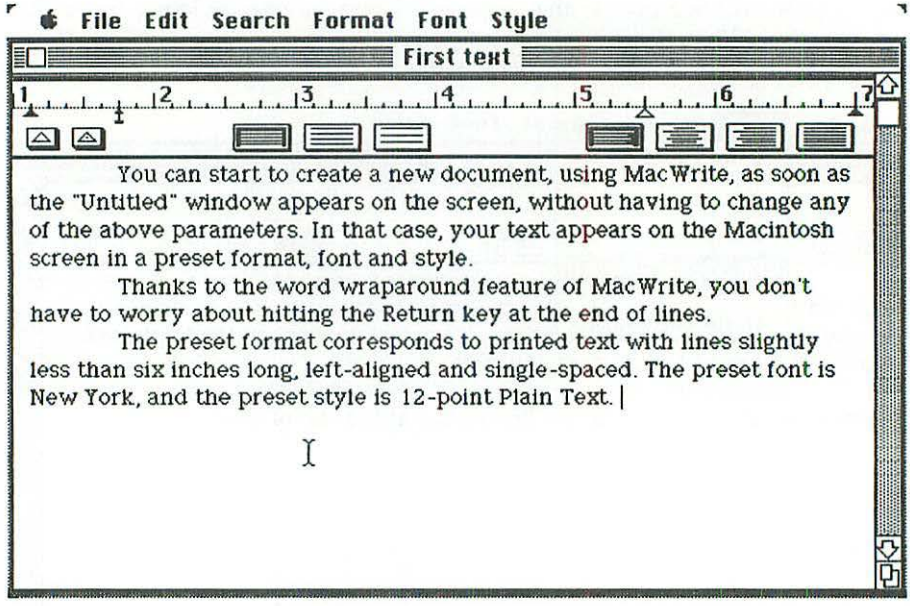
Now suppose that we decide that, instead of permanently deleting these three lines, we would like to *move* them to another place in the "First text" document: say, at the end. The following illustration shows how this is done:



We want the displaced lines to appear as a third paragraph, so the blinking vertical bar had to be set at the right position, as indicated in the preceding illustration. This operation took place in two steps. First, the I-beam was moved to the lower region of the window, and clicked, which set the insertion point just after the final full stop. Second, we hit <Return> , which moved the blinking vertical bar to its present location.

Now we pull down the Edit menu and choose the Paste command, so that the contents of the Clipboard will be transferred to the document window, starting at the insertion point.

The final result is shown in the following illustration:



If you were to open the Clipboard now, you would find that the three lines are still there. In other words, the same Clipboard information could be pasted into several different places in a document. The Clipboard will lose its current information, of course, when the next Cut or Copy command is carried out.

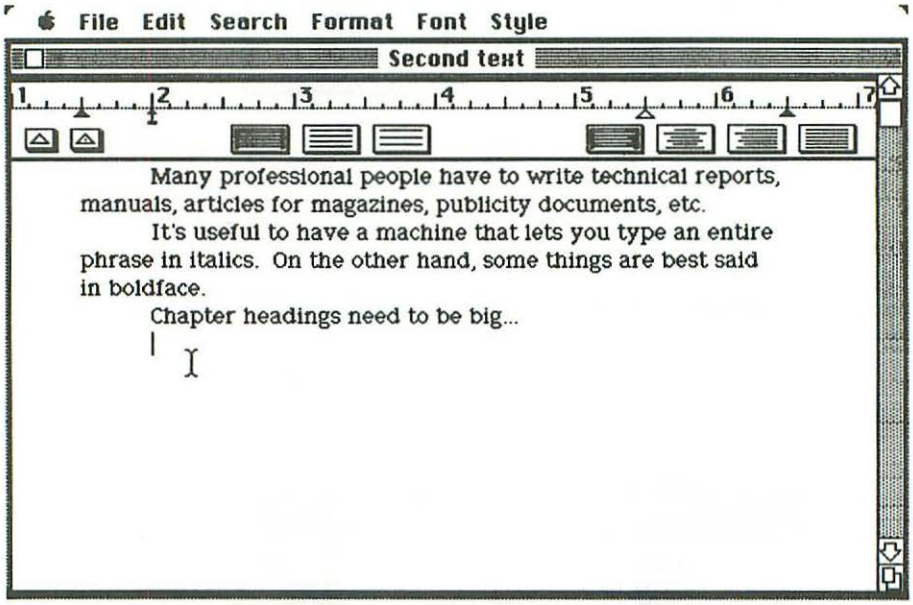
There is another way of deleting characters in a MacWrite document. You simply position the I-beam pointer at the righthand end of the text to be deleted, and you use the <**Backspace**> key. But the deleted characters are not sent to the Clipboard in this case. On the contrary, they are permanently lost.

For a newcomer to MacWrite, probably the most difficult operation to get used to is the selection of fragments of text (as in the preceding example) that are to be deleted, replaced or moved. If you don't have a steady hand on the mouse, the black highlighting has a tendency to run off in the wrong direction. Thankfully, there is a very handy trick, in two steps, for highlighting precisely and rapidly. First, you position the I-beam pointer to the immediate left of the text to be selected, and you click the mouse button. Then you position the I-beam pointer to the immediate right of the text to be selected, you hold down the <**Shift**> key and you click the mouse button, whereupon all the text between the two bounds is highlighted in black.

Pretty print

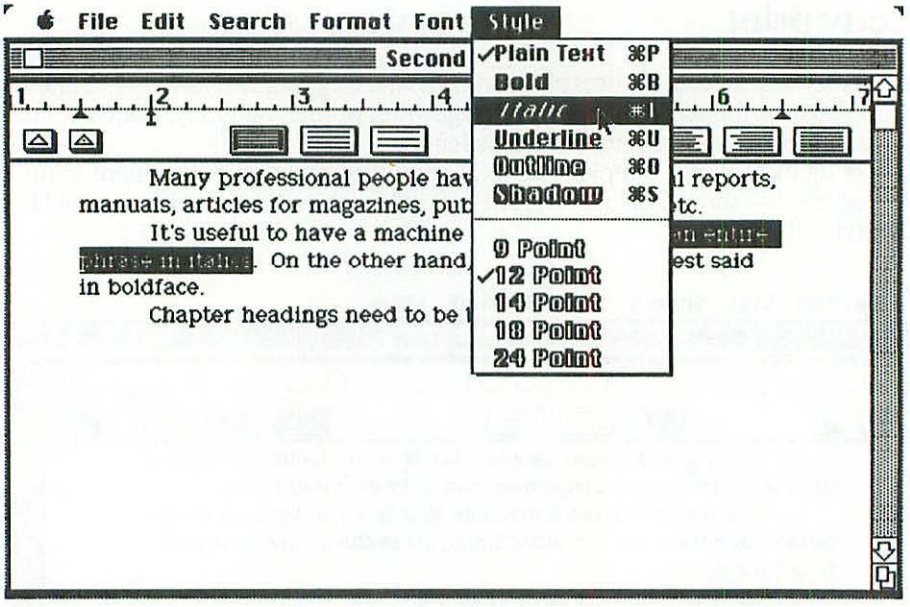
One of the most striking features of Macintosh is the elegance and variety of its graphic output, both on the screen and on the Imagewriter printer. Naturally, MacWrite takes full advantage of this property of the machine.

Let us look at an example. The following illustration — a document entitled "Second text" — shows half-a-dozen lines of ordinary New York font in 12-point Plain Text style:

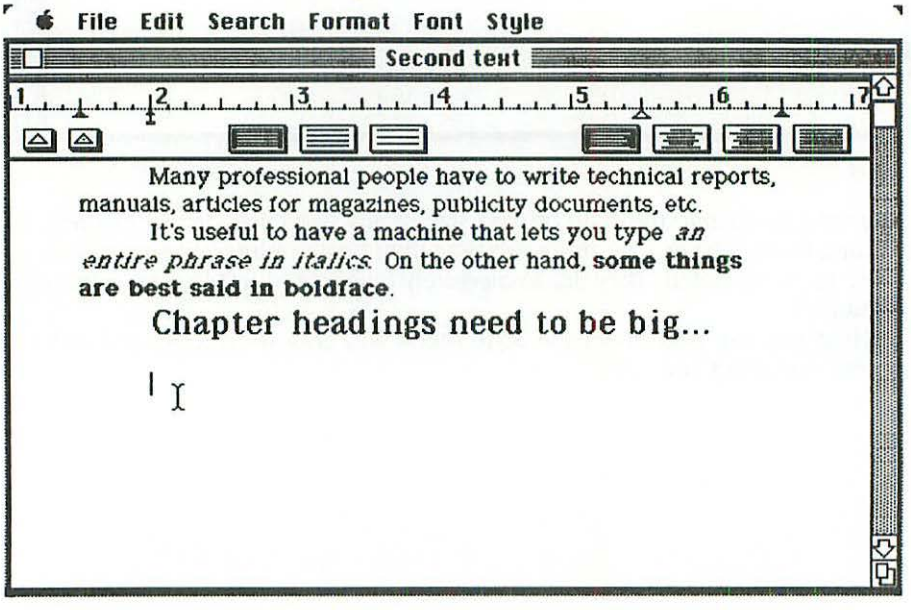


If you want to change the style or type size of any particular part of this text, the first thing to do is to use the mouse to drag the I-beam pointer over the characters that are to be modified. They are immediately highlighted in black (see following illustration).

Then you can pull down the **Style menu** and choose from several different typographical styles and sizes.



The following illustration shows the same document presented in an assortment of Plain Text, italics and boldface:



Here are samples of several different fonts and styles that are offered by MacWrite:

This is New York font in 12-point Plain Text style.
This is New York font in 12-point Bold style.
This is New York font in 12-point Italic style.
This is New York font in 12-point Bold Italic style.
This is New York font in 12-point Outline style.
This is New York font in 12-point Shadow style.

This is Toronto font in 12-point Plain Text style.
This is Toronto font in 12-point Bold Style.
This is Toronto font in 12-point Italic style.
This is Toronto font in 12-point Bold Italic style.
This is Toronto font in 12-point Outline style.
This is Toronto font in 12-point Shadow Style.

This is Chicago font in 12-point Plain Text style.
This is Chicago font in 12-point Bold style.
This is Chicago font in 12-point Italic style.
This is Chicago font in 12-point Bold Italic style.
This is Chicago font in 12-point Outline style.
This is Chicago font in 12-point Shadow style.

This is London font in 18-point Plain Text style.

This is Venice font in 18-point Bold style.

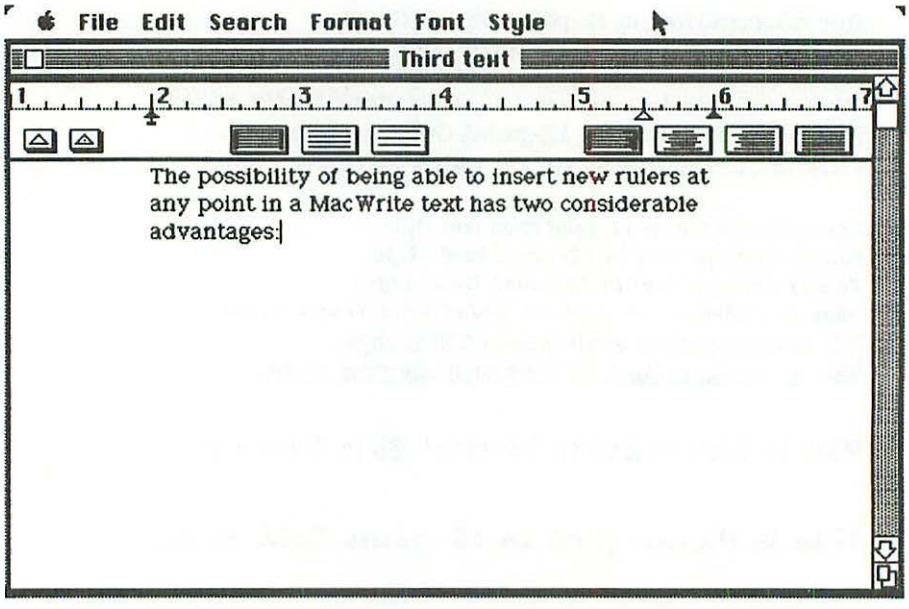
This is San Francisco font in ...

For the author of technical reports, articles or books, it's hardly an exaggeration to say that working with a Macintosh, MacWrite and an Imagewriter is a little like having your own private printing press tucked away in a space no bigger than a travel bag.

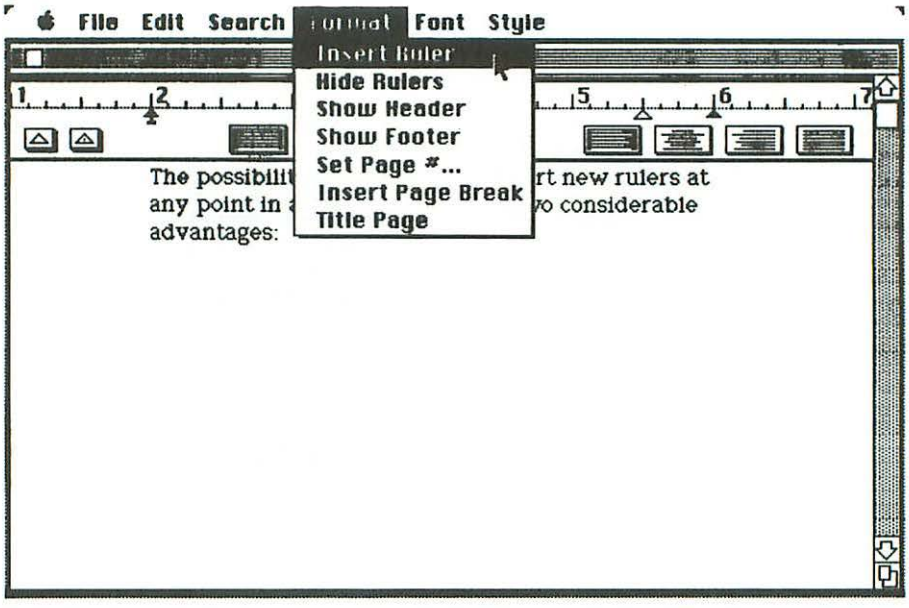
Rulership

Up until now, our MacWrite examples have included a single ruler at the beginning of the document. In fact, nothing prevents us from inserting any number of different rulers in the document being created by means of MacWrite.

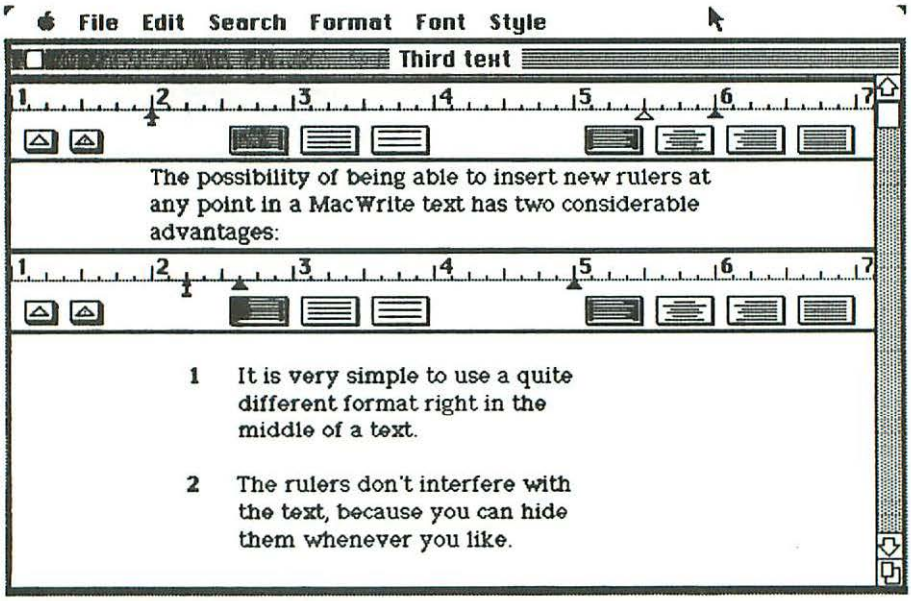
Suppose that we are producing a document called "Third text", that starts out as shown in the following illustration:



At this point, judging from the text, we are about to write two subsidiary paragraphs, describing the "two considerable advantages" of ruler insertion in MacWrite. In order to give these paragraphs a distinctive layout, let's insert a new ruler. This is done by pulling down the **Format menu** and choosing the **Insert Ruler** command, as shown in the following illustration:

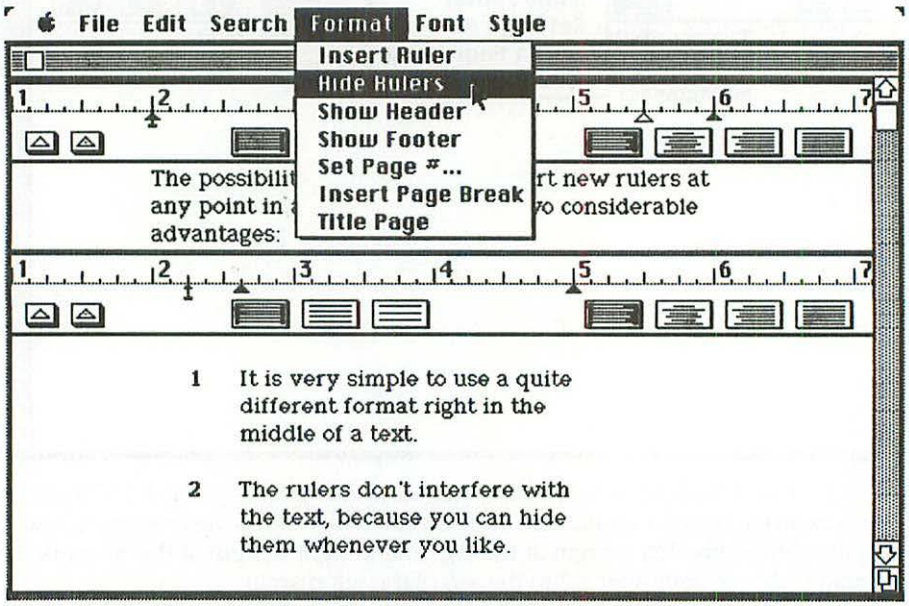


The first ruler set the left margin at the 2″ mark and the right margin at the 6″ mark, and there was no paragraph indentation. The new ruler (see following illustration) can be set quite differently: left margin at the 2⅝″ mark, right margin at the 5″ mark, and paragraphs that actually start ⅜″ to the *left* of the left margin.

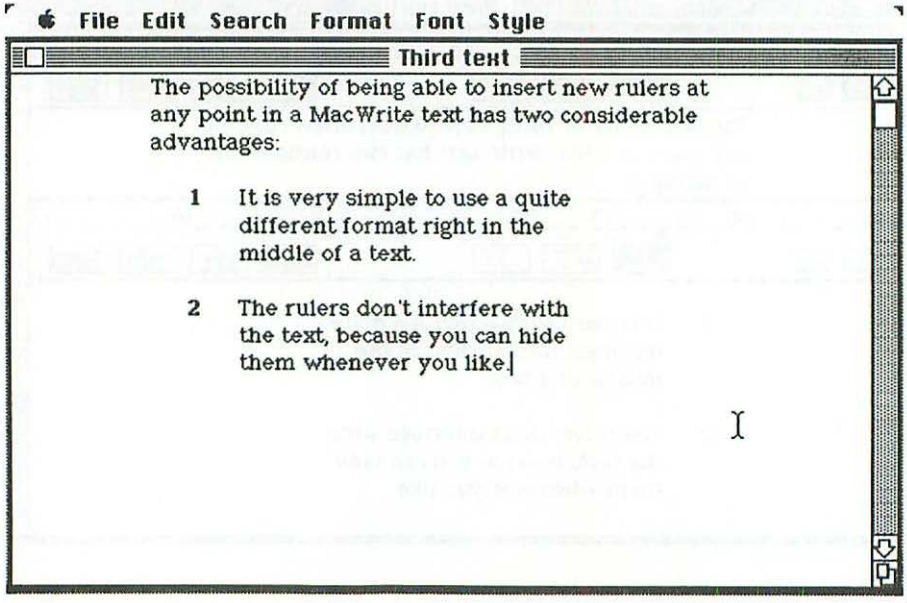


The two numbered paragraphs correspond perfectly to the new format.

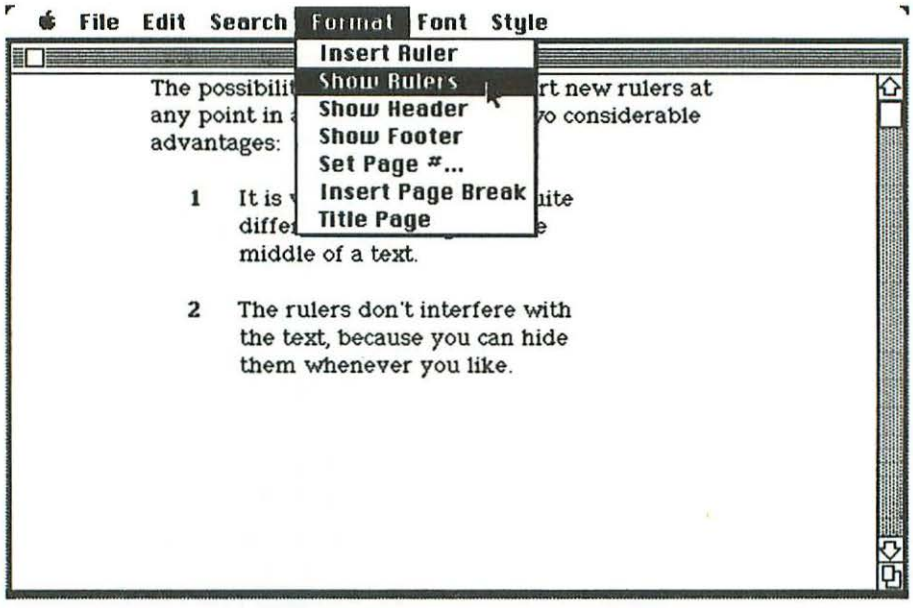
We would now like to see the complete document without the two rulers. This is achieved by choosing the **Hide Rulers** command in the Format menu, as shown in the following illustration:



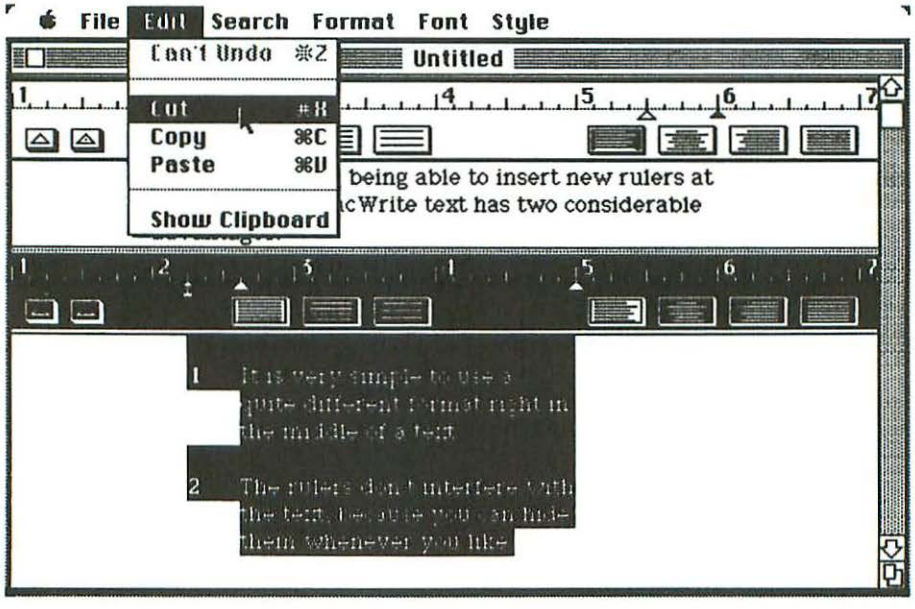
The final result is shown in the following illustration:



If we now wish to put the rulers back into the displayed text, we simply choose the **Show Rulers** command in the Format menu, as shown in the following illustration:



Maybe we might decide to remove both a ruler and a block of text. In that case, we drag the pointer down over the ruler and the text to be removed, in order to highlight them (see following illustration), and then we choose the familiar Cut command in the Edit menu.

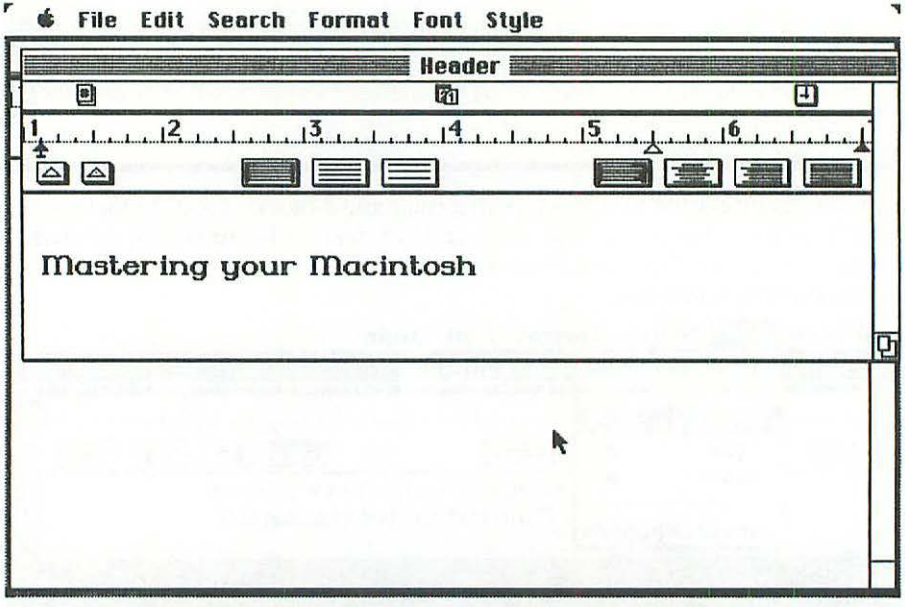


To be precise, there's only one thing you cannot cut out of a MacWrite document: namely, the *first* ruler.

People who have worked with the old-fashioned methods of layout specifications in word processing, using all kinds of embedded codes, will surely agree that the MacWrite concept of rulers is ingenious. As usual, you can actually *see*, instantaneously, the effect of a new ruler upon the layout of your document, and so there is no possibility whatsoever of ambiguity.

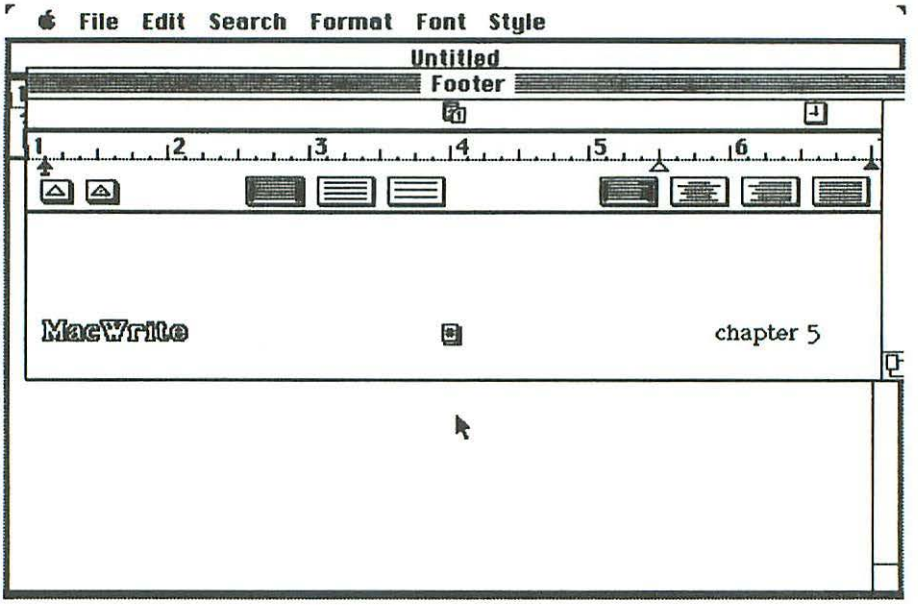
Another remarkable feature of MacWrite concerns the top and bottom page margins, called *headers* and *footers*. In many conventional word processing systems, the handling of this subject — that might be referred to as "page breaks" (for want of a more precise term) — is frankly messy. MacWrite, on the other hand, proposes the simple and elegant solution of autonomous *windows*, with *rulers*, for both the header and the footer.

To get at such goodies, you use one or other of the powerful commands of the Format menu, that have appeared in several of the preceding illustrations. **Show Header**, for example, gives rise to the following window:



We have imagined here the case of an author who would like the title of his book — "Mastering your Macintosh" — to appear at the top of every page of the typescript.

Here's an example of the outcome of choosing the **Show Footer** command:

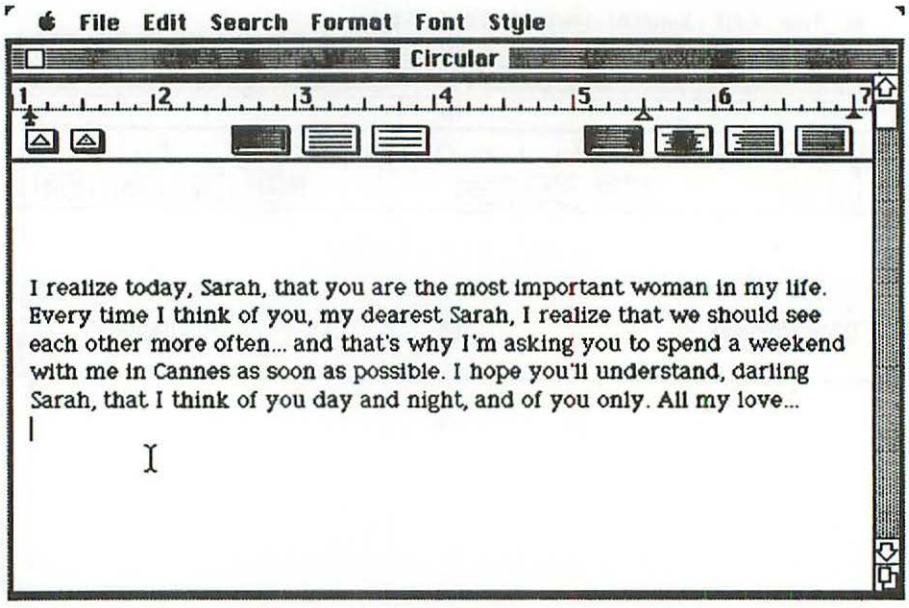


For both the header and footer windows, any quantity of standard text can be typed in . . . such as document titles, chapter names, etc. In addition, the user can drag down any of the three little icons just below the title bar: page number, calendar date and clock time. So, the author can print out numbered pages that are stamped with the references of the exact moment at which they were produced. What more could you ask for?

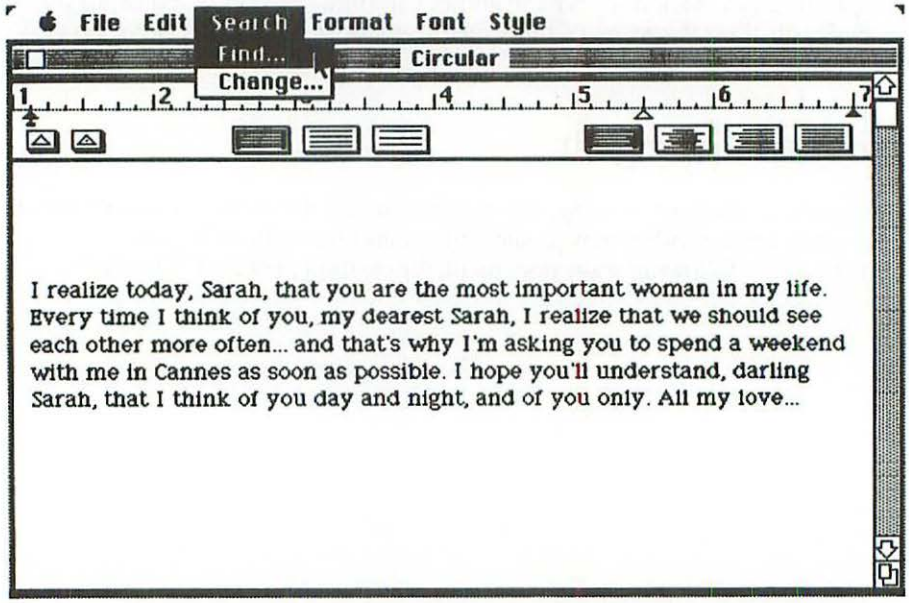
Find it and change it

Quite often, in word processing, we want to find all the occurrences of such-and-such a term, maybe with a view to substituting another term in its place.

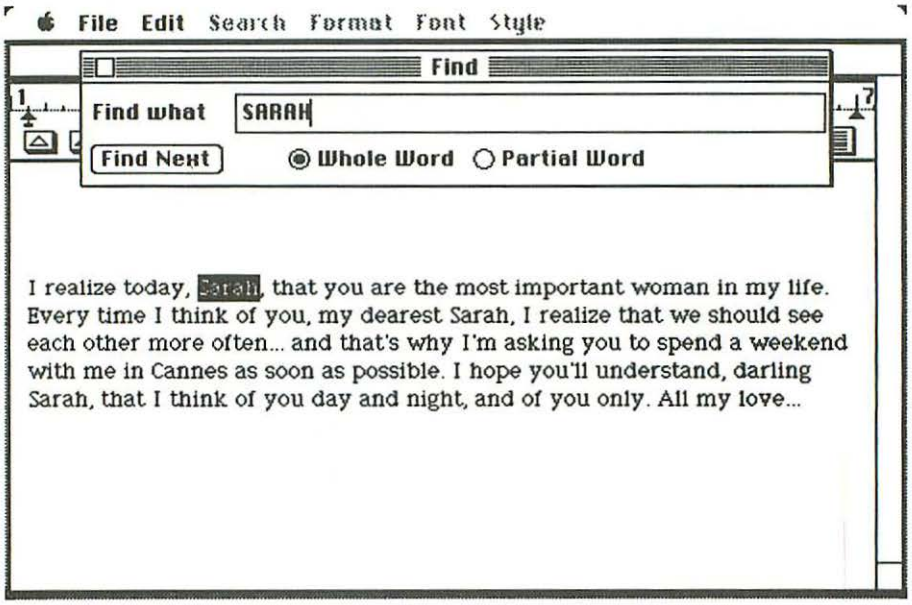
Look at the following short document, for example, entitled "Circular":



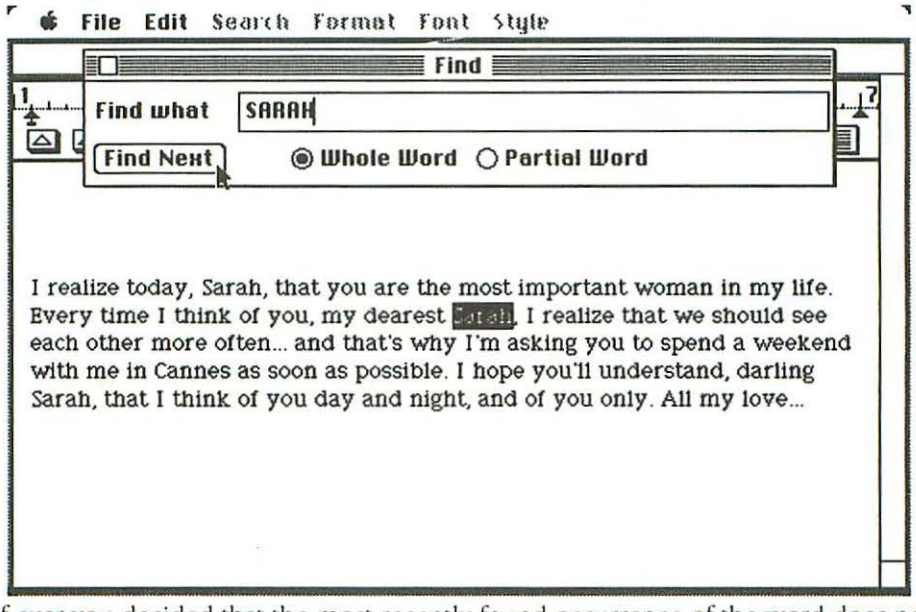
Let's suppose that we want to locate all the occurrences of the word "Sarah". To do so, we start out by pulling down the **Search menu** and choosing the **Find** command, as shown in the following illustration:



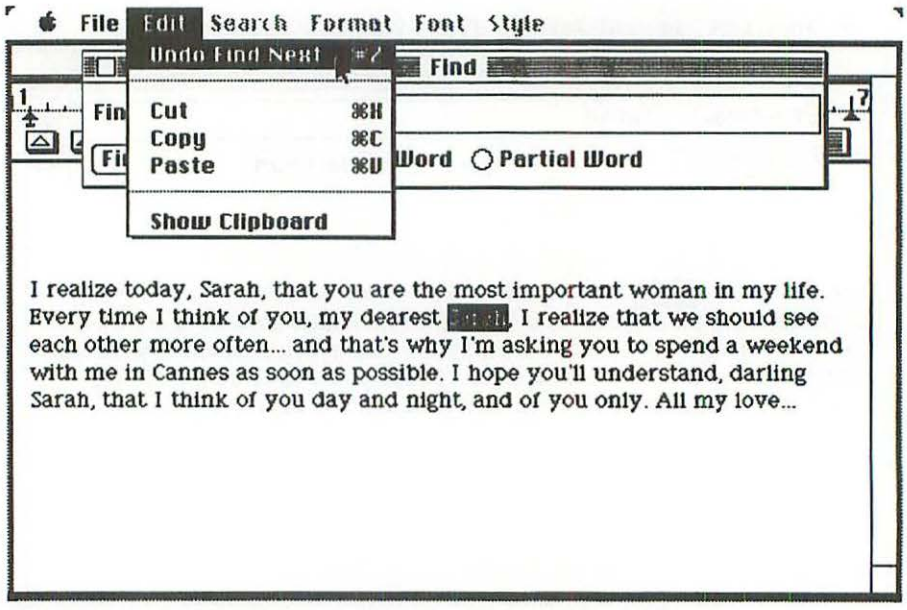
A dialog box appears on the desktop (see following illustration), asking us to spell out what it is we want to search for in the document. Notice that it doesn't matter whether you use upper or lower-case letters.



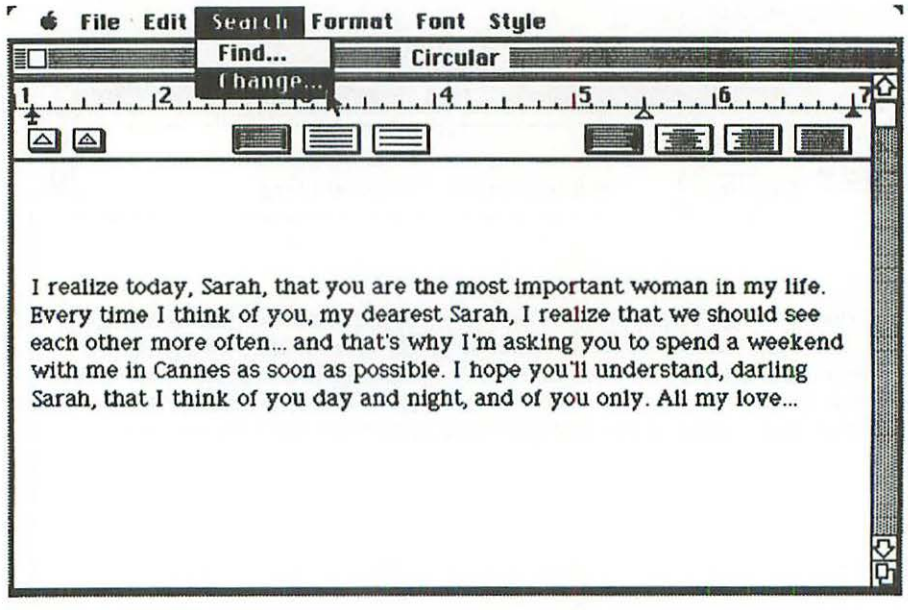
The first "Sarah" is highlighted in black. You can then click the **Find Next** button to carry on searching for further occurrences of the same word, as shown in the following illustration:



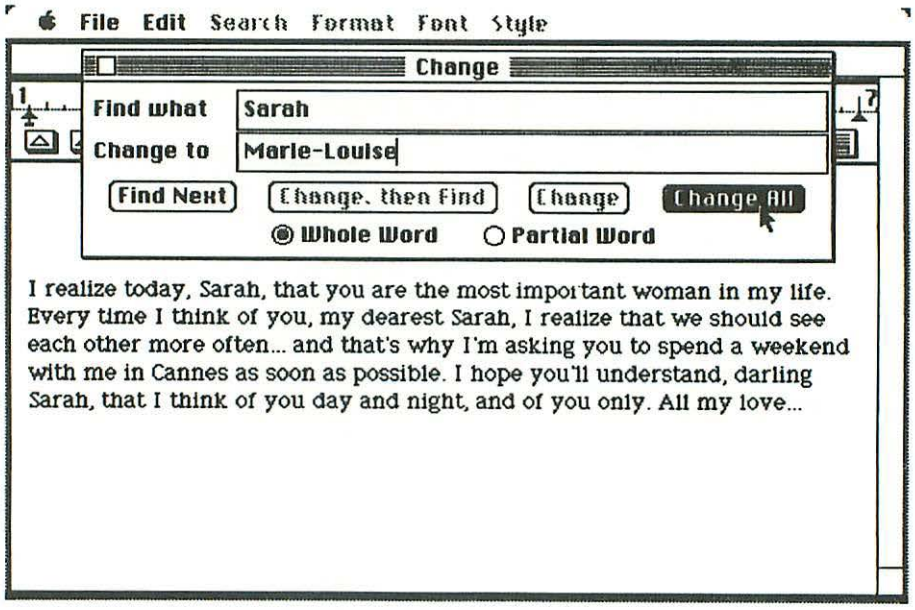
If ever you decided that the most recently-found occurrence of the word does not interest you, it's possible to call upon the **Undo Find Next** command in the Edit menu (see following illustration) in order to return to the previous occurrence.



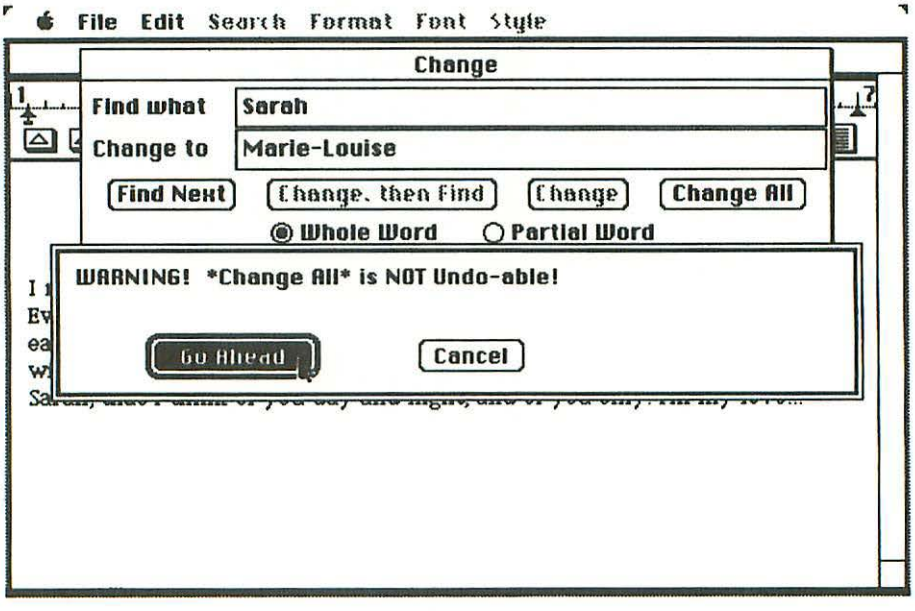
There is also a **Change** command in the Search menu, as shown in the following illustration:



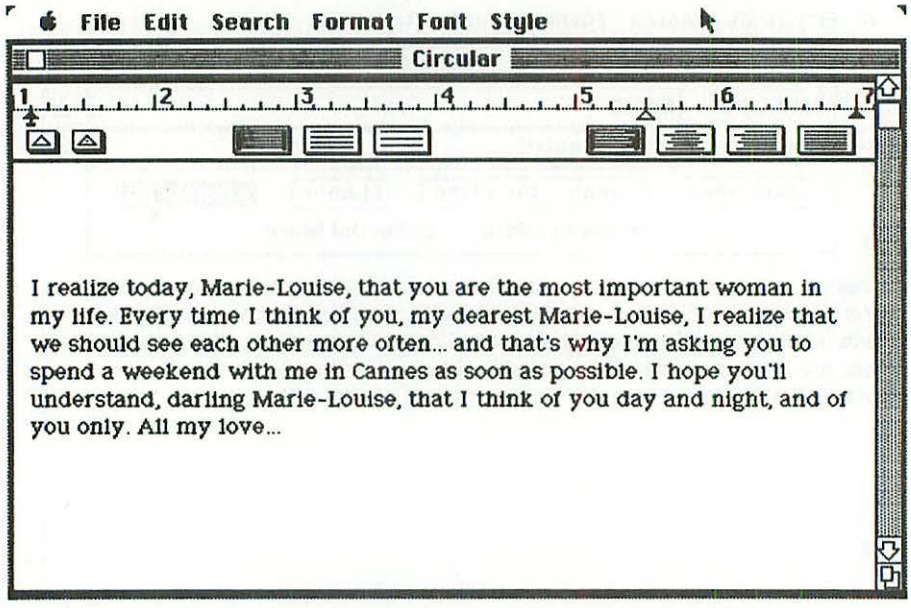
Let's suppose that we intend to change every occurrence of "Sarah" into "Marie-Louise". The precise manner of doing this is shown in the following illustration:



Clicking the **Change All** button in the dialog box means that we want to change *every* occurrence of the argument word, not just the first one. Macintosh, ever conscious of the dangers of behaving in a foolhardy manner, asks us (see illustration below) if we're really sure that we know what we're doing. This kind of warning is called an *alert box*.



As soon as we click the Go Ahead button, the irreparable modification is carried out, and we obtain the following elegant result:



That brings us to the end of our presentation of MacWrite. We shall nevertheless be returning to this subject later on, to show how images created by MacPaint and MacDraw can be incorporated into MacWrite documents.

No doubt the most astonishing thing about MacWrite is the fact that, in a text processing environment, so much can be carried out by means of the mouse alone, without having to use the keyboard. Once you've gotten into the habit of using MacWrite and the mouse to churn out correspondence and professional documents, it's practically impossible to revert to antiquated methods of text preparation. There is no longer any sense in carrying out guessing games about the marketing potential of Macintosh, but it's probably true to say, retrospectively, that this machine, armed with MacWrite, could have been offered to the public as no more than an avant-garde word processing system, and already it would have become a best seller.

chapter 6

"MacPaint" — Image Processing

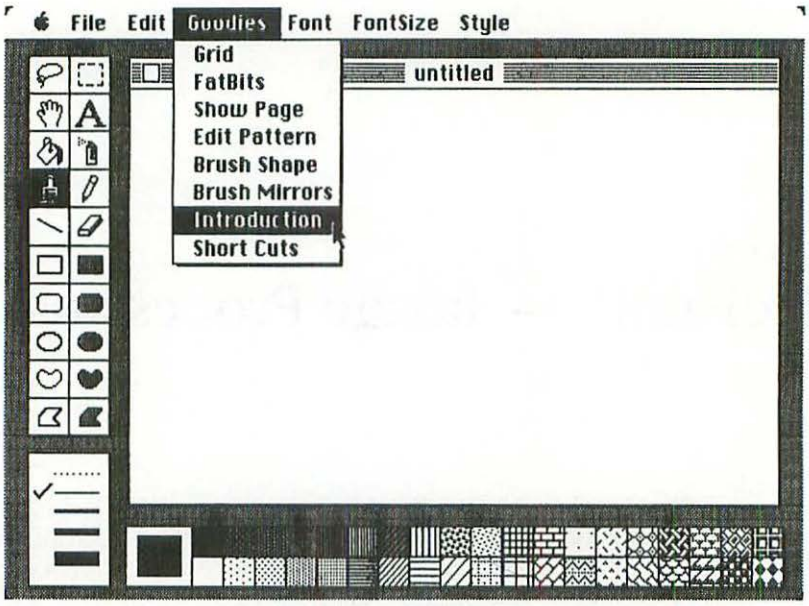
Apparently it was Napoléon Bonaparte who once declared that a freehand sketch provides more information than a thousand words. He was right, for once, but it remains a fact — even today — that professional people have never really acquired the habit of inserting sketches into their correspondence and reports. Why is there this reluctance to use images? Maybe one explanation is that we have become accustomed to using simple machines —such as the typewriter and the telephone — which make it easy for us to communicate by means of written or spoken words, whereas the idea of producing sketches has always seemed to be a relatively onerous task that is best left to graphics artists. In particular, there has never been a simple machine, like the typewriter, that would enable people who are not necessarily artists to produce attractive sketches that could be inserted into their typed documents.

That machine now exists, and its name is . . . Macintosh. The processing of graphic material is a fundamental concept in the case of this new machine, and it is no accident if Macintosh has appeared on the market with two splendid software tools in this domain: *MacPaint* for freehand image processing, and *MacDraw* for structured technical drawing.

Electronic painting

MacPaint is housed on the same disk as MacWrite, and it is opened in the normal manner by double-clicking the directory icon labeled "MacPaint".

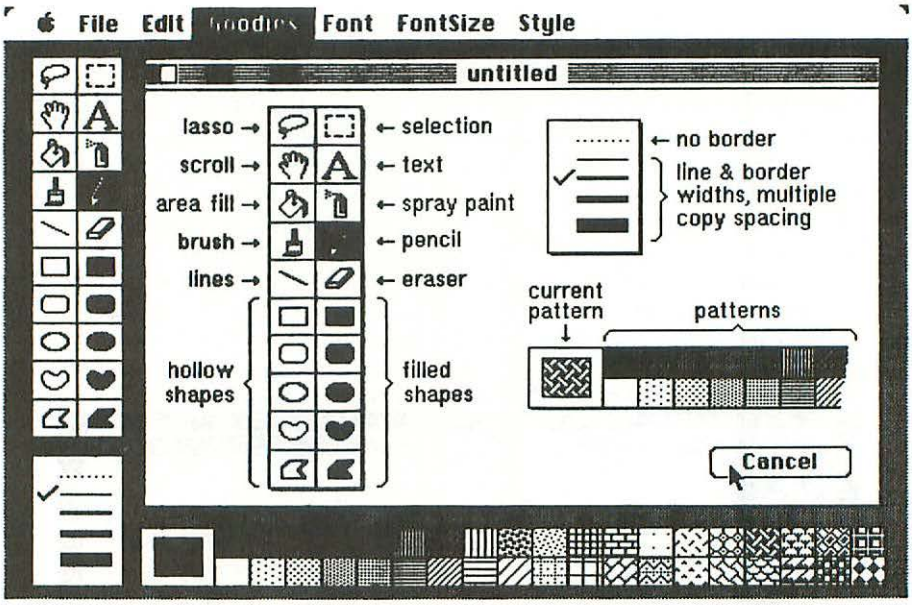
After a few seconds, the distinctive MacPaint window arrives on the screen, as shown in the following illustration:



It is a fact that about 90% of the possibilities of MacPaint can be grasped intuitively once you sit down in front of a Macintosh and start to play around with this software tool. Even a child with no training whatsoever in the manipulation of a personal computer can rapidly attain this level of performance. (Maybe that last sentence should be changed to read: "*Especially* a child with no training whatsoever . . .", because children are often more successful than adults in getting used to a revolutionary concept such as electronic painting.)

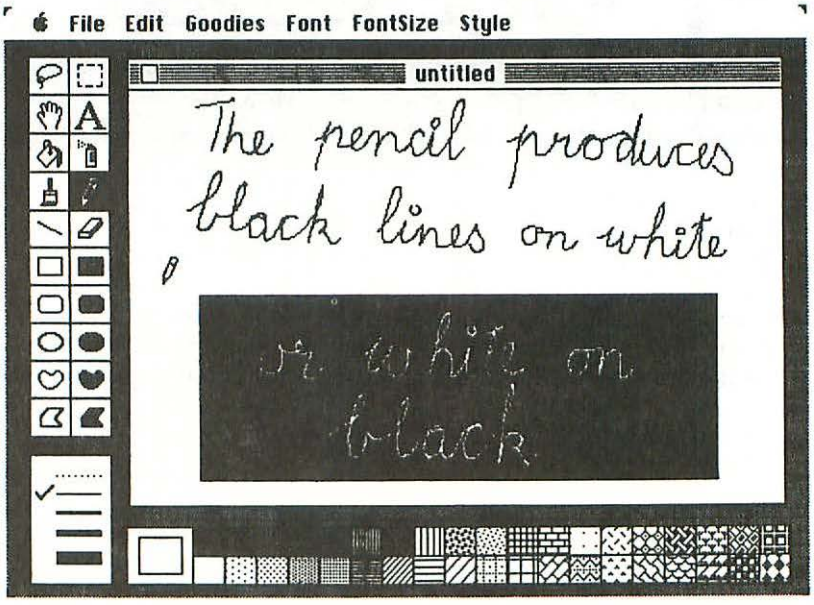
To the left of the main window, there's a block of 20 icons that represent *painting tools* of many different kinds. Underneath the main window, there's a block of 38 *patterns* that you might think of as your palette of painting colors. And the lower left corner of the screen proposes several alternatives for *line and border widths*.

In order to learn the names of the various MacPaint tools, you can pull down the so-called **Goodies** menu, as indicated in the previous illustration, and choose the **Introduction** command. The result is as follows:



To get rid of this display, click the **Cancel** button in the lower right corner.

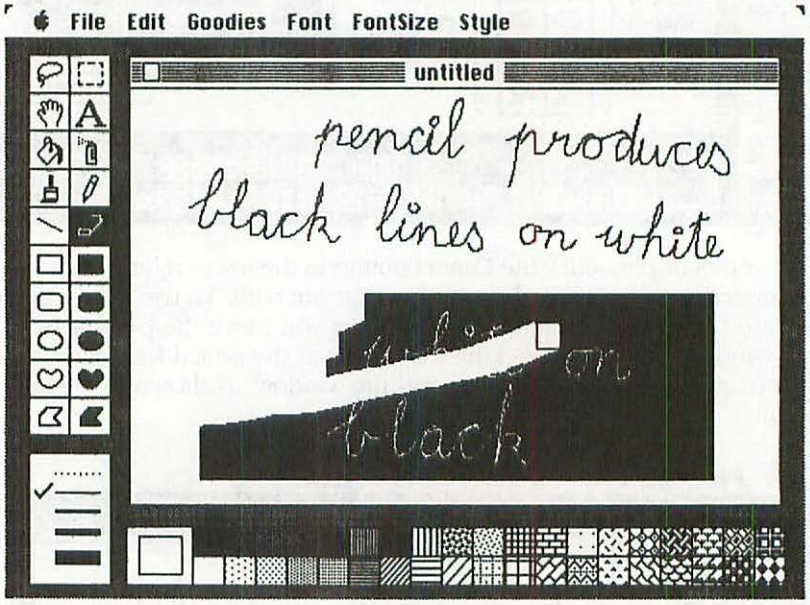
The *pencil* is probably the best tool to start out with. To use it, you move the pointer onto the pencil icon and click it. When you move the pointer back onto the main window, it has acquired the same form as the pencil icon, and you draw simply by dragging the pencil pointer over the window, as shown in the following illustration:



The pencil always draws lines of the same width, irrespective of the indications down in the lower left corner of the screen, and the color does not depend on which pattern has been selected. The rule for pencil colors is elementary. If you start dragging the pencil on a white background, the trace is black and it remains black as long as you keep the mouse button held down, even if the pointer moves over a black zone on the window. Inversely, if you start dragging the pencil on a black background, the trace is white.

Later on we shall see that the pencil has another very important role to play, when parts of an image are enlarged for point-by-point modifications.

To rub out parts of the screen, click the *eraser* icon, just below the pencil icon. The pointer takes the form of a white square (see following illustration) that you can drag across any part of the image that you wish to erase.

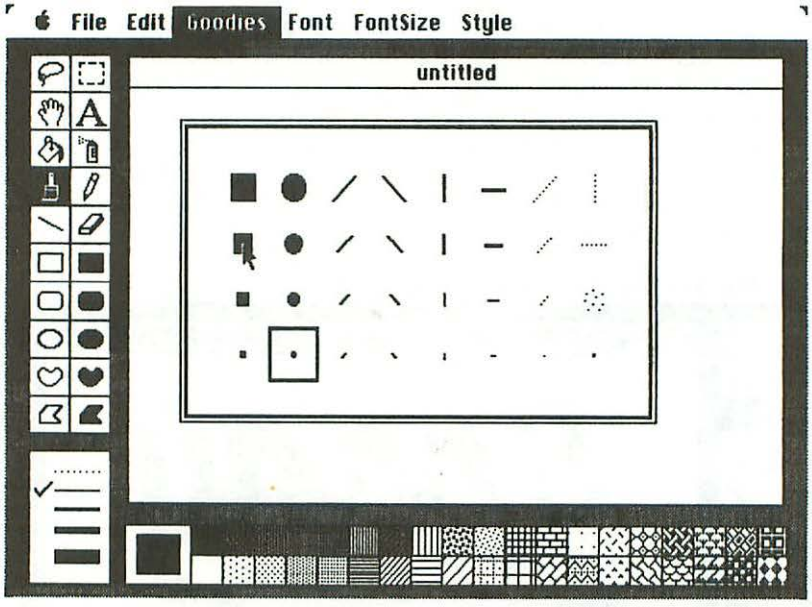


To erase the entire window, simply double-click the eraser icon.

Let us turn next to the *brush*, whose icon appears just to the left of the pencil.

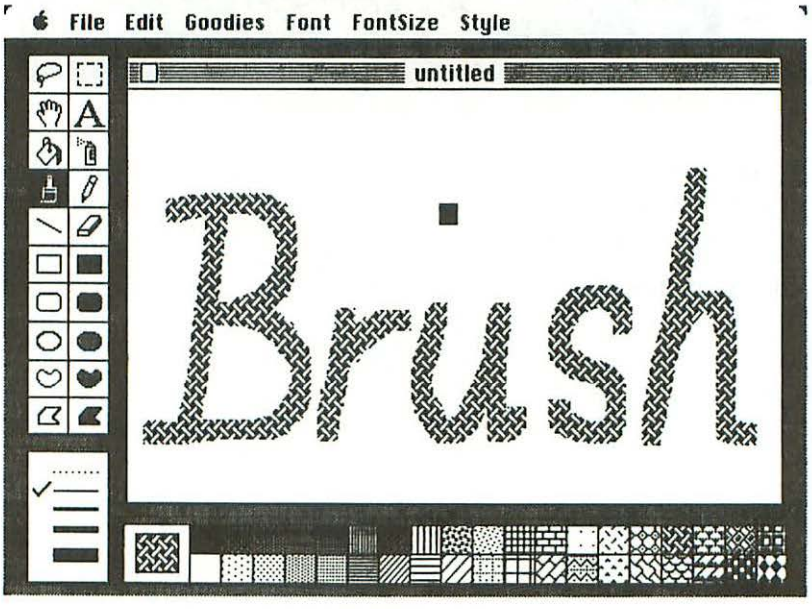
The brush is used for sketching in the same manner as the pencil, but it operates in a more complex fashion.

Before actually manipulating the brush, let us pull down the Goodies menu once again and choose the **Brush Shape** command. Alternatively, you can simply double-click the brush icon. The outcome is shown in the following illustration:



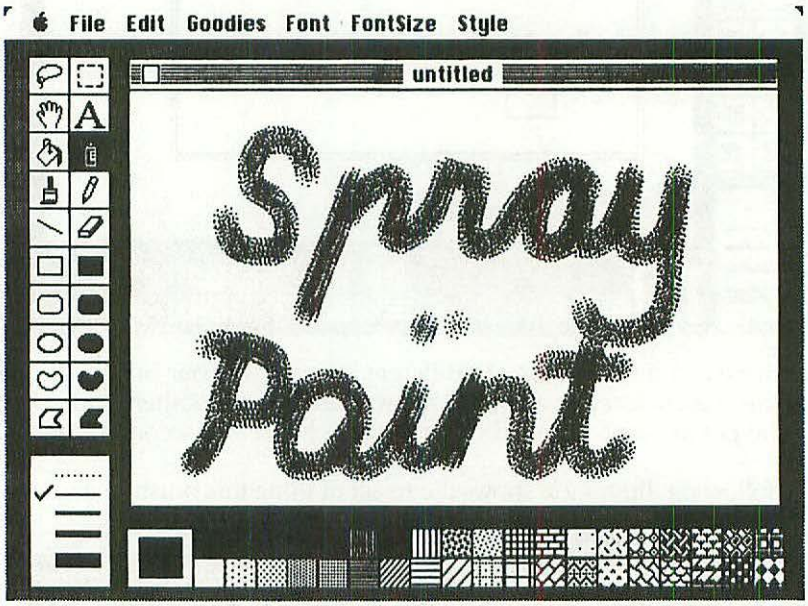
Here we have a choice between 32 different "shapes" for our brush. The present shape is the one enclosed in a square, but we can choose another shape simply by moving the pointer onto it and clicking it. Let's choose the second-biggest square shape.

The following illustration shows the result of using this brush to trace a word:

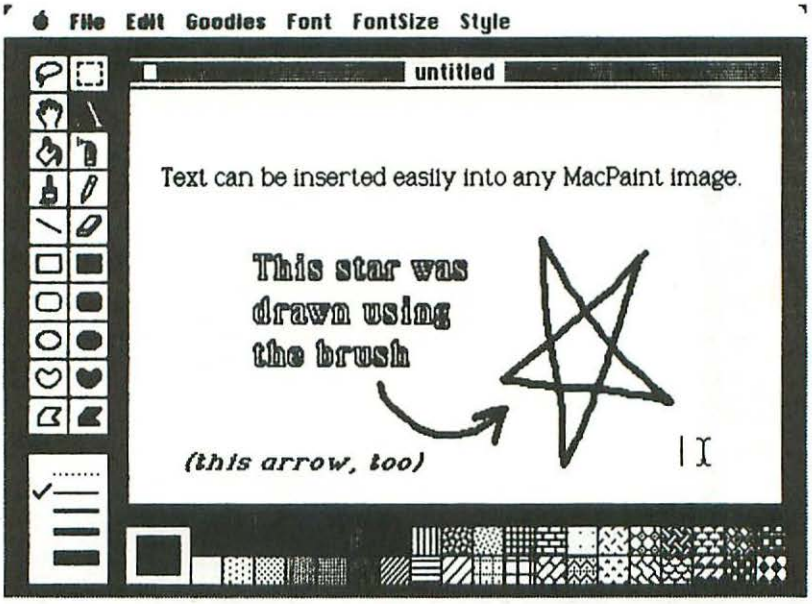


Before using the brush, we moved the pointer down to the pattern that is located four from the righthand end of the top line, and we clicked it, making it the *current pattern*. As you can see, the brush actually does its painting using the current pattern as its "color".

Let us now erase the contents of the window, and select the *spray paint* tool, whose icon is just above the pencil. And let's choose plain black paint down at the bottom of the screen. The following illustration shows what you can obtain by dragging the spray paint tool around on the window:



The tool with a big letter A for icon enables you to insert text into MacPaint images. In much the same way as in MacWrite, you have access to a **Font** menu, a **FontSize** menu and a **Style** menu. On the next page, there is an example of an image containing text. Once the text has been entered, MacPaint no longer "thinks" of it as text, but as just another part of the global image.



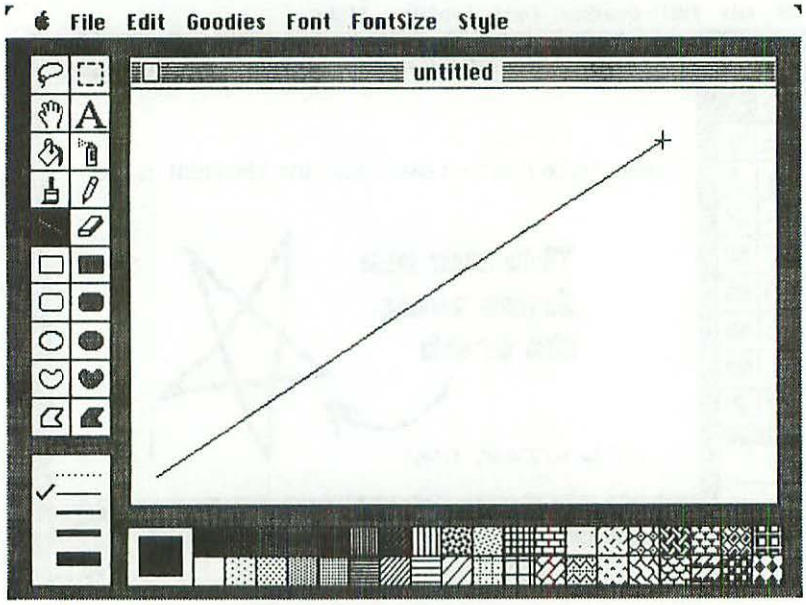
Rubber banding

In the Macintosh context, one frequently encounters the concept of *rubber banding*. What is meant by this expression?

Let's start out with the case of one-dimensional rubber banding. Imagine that you're holding on to one end of a stretched piece of elastic, the other end of which is pinned to the top of the table by a thumbtack. As you move your hand over the table top, the extended elastic changes direction, and lengthens and shortens.

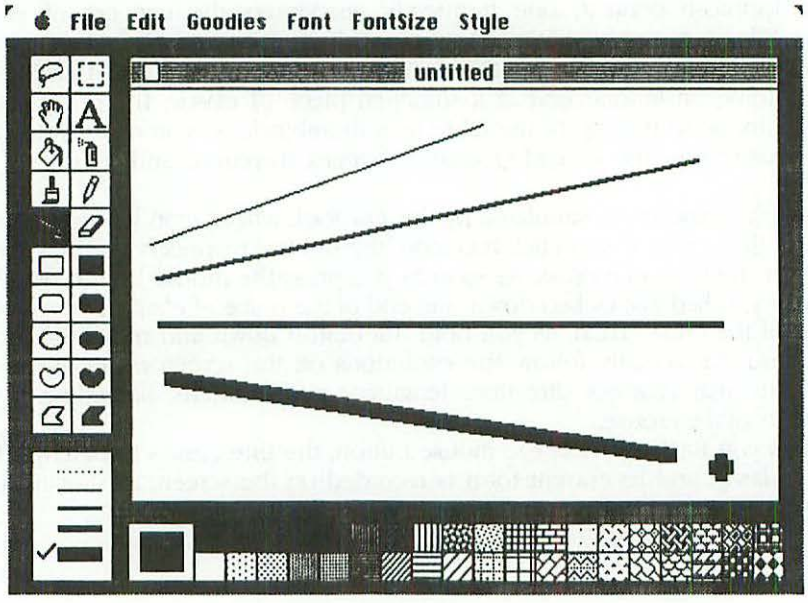
This phenomenon is simulated by the *line* tool, whose icon is located just to the left of the eraser. If you click this icon, the pointer reappears on the drawing window in the form of a cross. As soon as you press the mouse button, MacPaint reacts as if you had just tacked down one end of the piece of elastic at the current position of the cross. Then, as you hold the button down and move the mouse around, you can actually follow the evolutions on the screen of an elastic-like straight line that changes direction, lengthens and shortens, according to the movements of the mouse.

When you finally release the mouse button, the line ceases to behave like a piece of elastic, and its current form is recorded on the screen, as shown in the following illustration:



In this window, the position of the metaphorical thumbtack (where the line was started) is down in the lower left corner of the window, and the cross marks the spot where the mouse button was released and the line ended.

We have here a first example of a tool that takes into account the setting of the line width down in the lower left corner of the screen, as shown in the following illustration:



This illustration shows four lines traced with the different widths. To change the line width, you simply position the pointer on the desired width and click it, whereupon a check mark appears to the left of the selected line.

Incidentally, if you wish to draw lines that are either perfectly horizontal or vertical or inclined at 45°, then all you have to do is to hold down the <Shift> key while you're dragging the pointer.

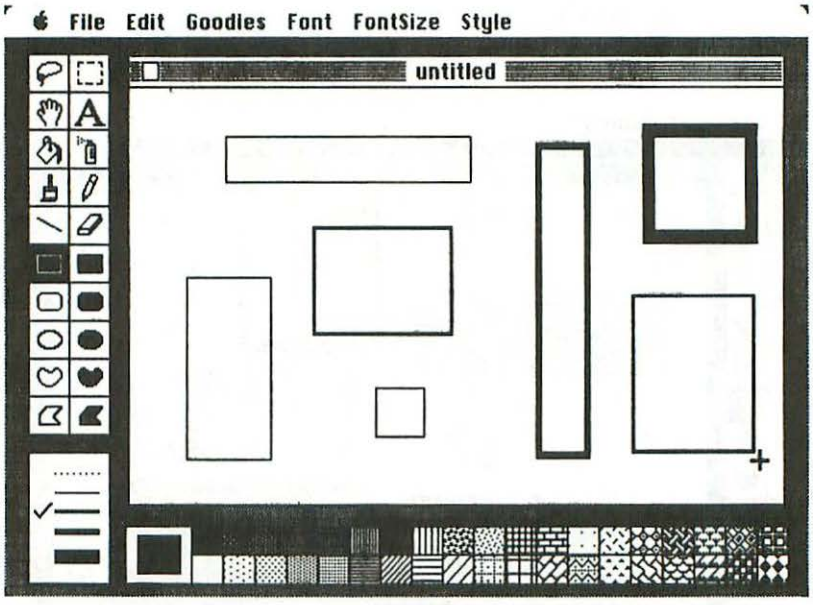
Let us now turn our attention to the case of two-dimensional rubber banding, which means that it's as if we were using a closed rubber band instead of a length of elastic. Here, the general idea is that the metaphorical rubber band is attached to one point, and that its fixed shape (rectangle, ellipse) can be enlarged or reduced simply by dragging the pointer.

This behavior is exhibited by some of the ten drawing tools whose icons appear in the lower half of the block.

To use any of these tools, click the desired icon, and then drag the pointer around the window, as usual, to draw. In the case of all these ten tools, the pointer itself is in the form of a cross.

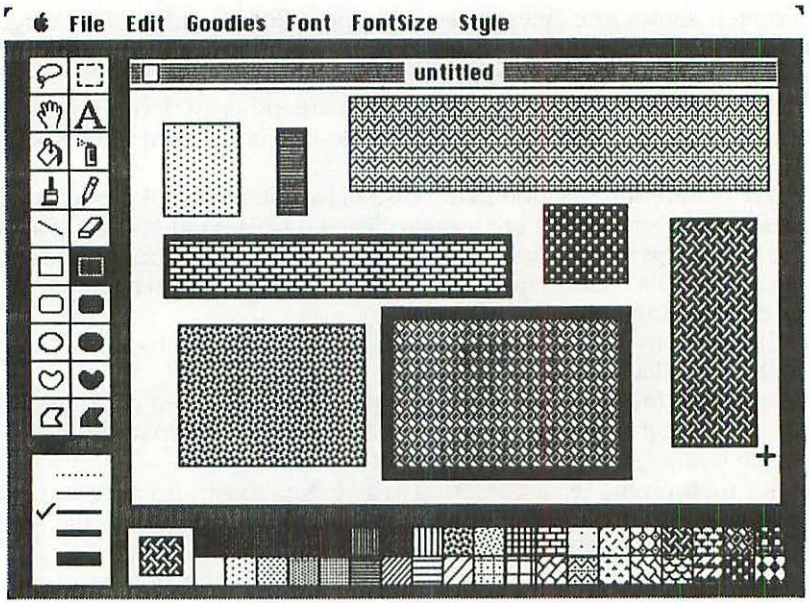
The five tools on the left are referred to as *hollow shapes*, because you end up with black line drawings on a white background. The width of the lines is determined by the setting down in the lower left corner of the screen.

The following illustration demonstrates the use of the hollow rectangular shape:

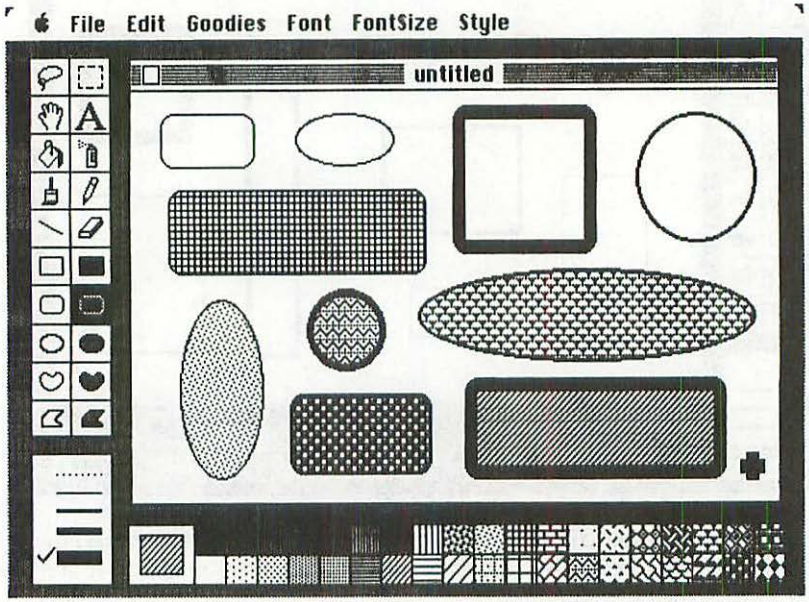


The five drawing tools whose icons appear below the eraser are referred to as *filled shapes*. They behave like the hollow shapes, except that the resulting line drawings are filled in with the current pattern.

The following illustration shows us various rectangles that are filled in with an assortment of patterns:



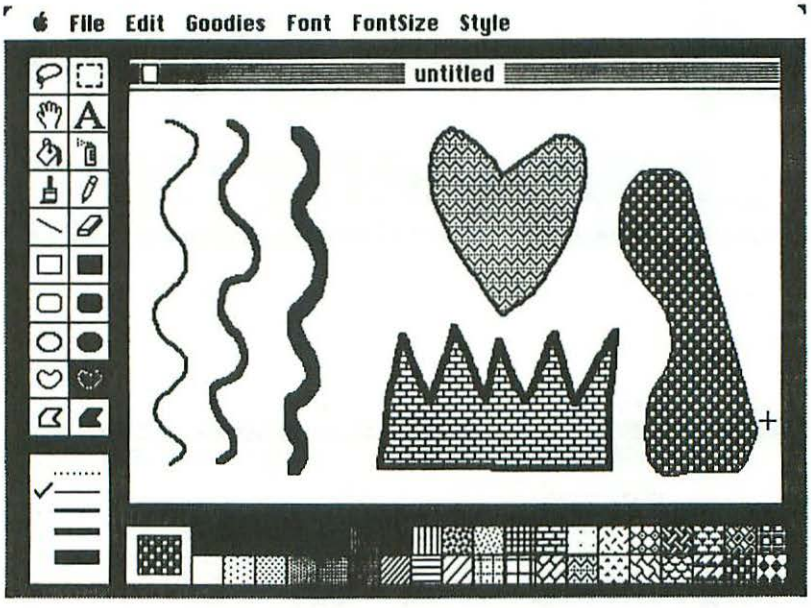
The *round-cornered rectangles* and *ellipses* behave in a similar manner. The following illustration shows us an assortment of hollow and filled shapes made with these tools:



Freehand drawing

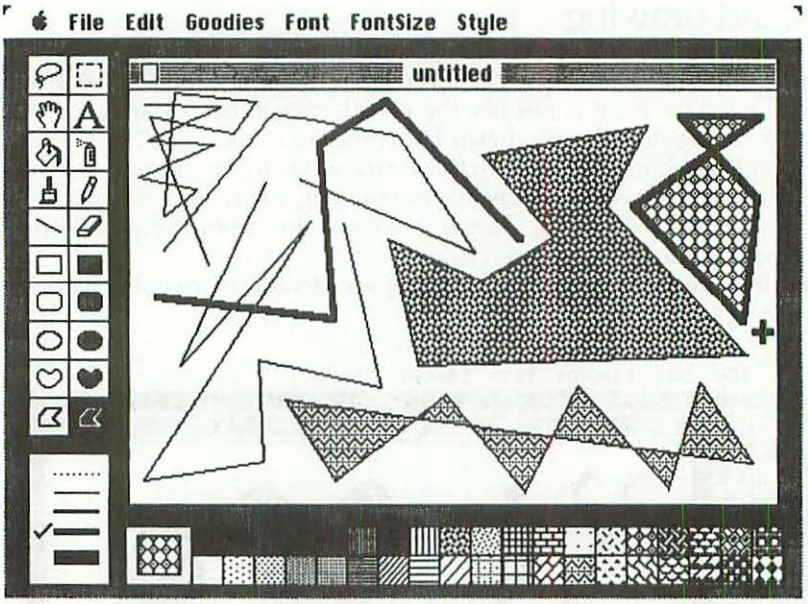
The heart-shaped icons just below the ellipses are used for sketching irregular shapes. The hollow heart is just like the pencil, except that it takes account of the line width setting down in the lower left corner of the screen. The filled heart is more complicated, in the sense that the shape has to be closed in order to be filled. So, as soon as you stop dragging the pointer, and release the mouse button, MacPaint closes the shape by tracing a straight line from the point where you stopped back to the beginning of the curve.

The following illustration shows samples of various irregular shapes drawn with these tools:

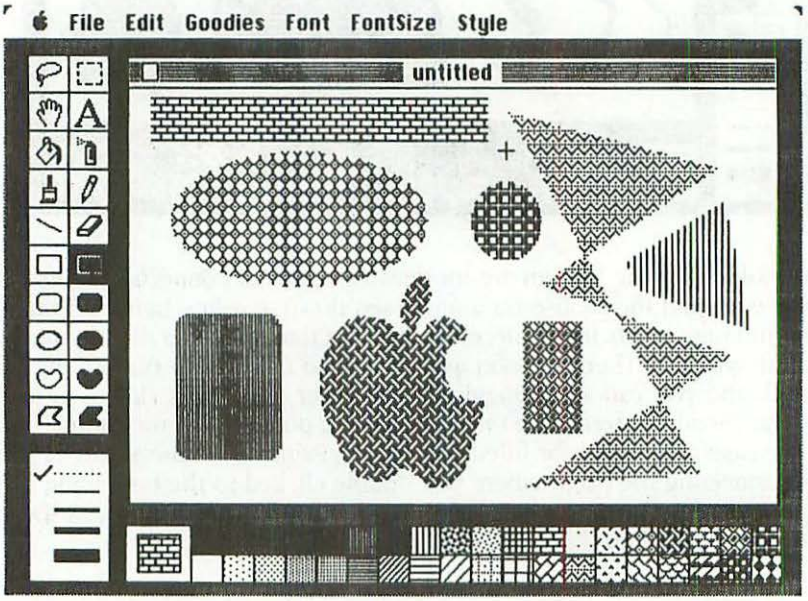


The two tools down the bottom are for drawing series of connected straight lines. As long as you hold the mouse button pressed down, the line being drawn at any particular moment reacts like a piece of elastic, in that it follows the pointer to any spot on the window. Then, as soon as you release the mouse button, the line is terminated, and you can start dragging the pointer in another direction to draw another line. Finally, to terminate the drawing, you double-click the mouse button.

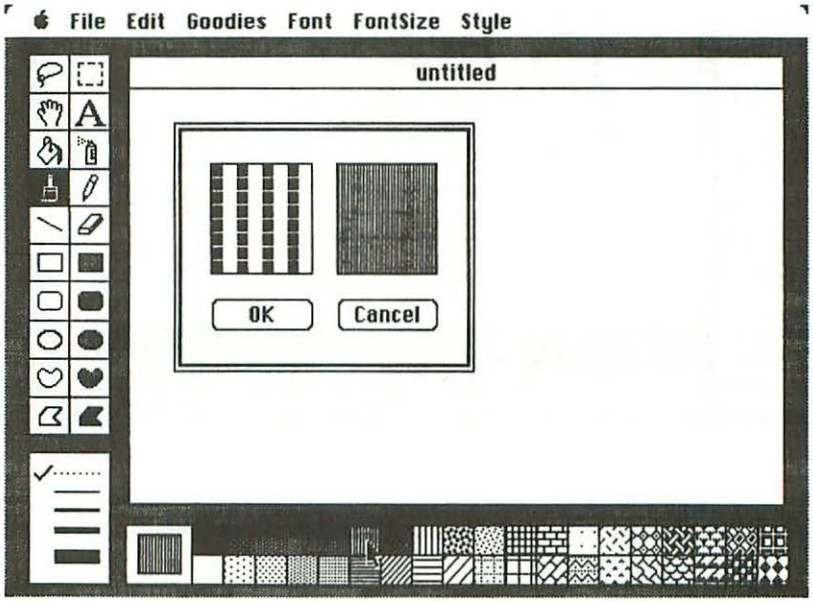
In the case of the tool for filled shapes, MacPaint closes the series of straight lines by connecting the point where you double-clicked to the beginning of your series of lines, and this gives rise to one or several painted polygons, as shown in the following illustration:



One final precision: If you select the dotted line down in the lower left corner of the screen, that means that filled shapes will be painted without borders, as shown in the following illustration:



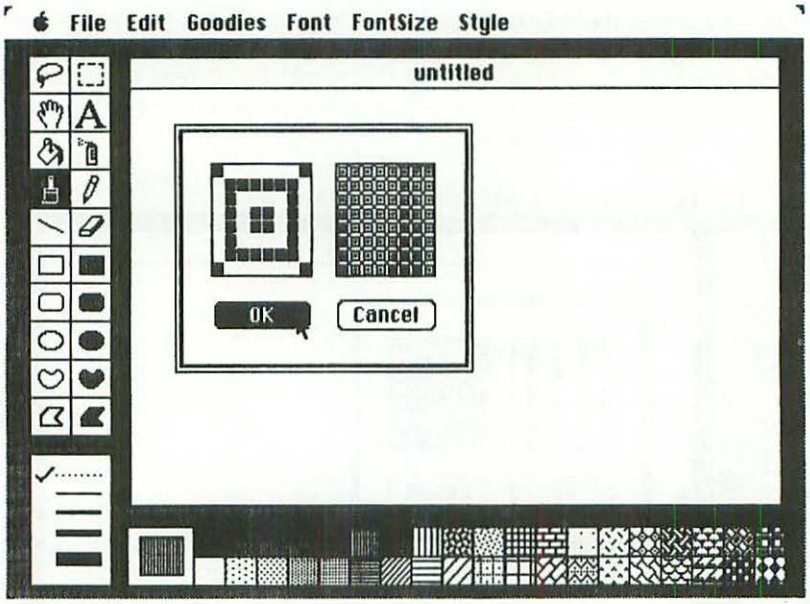
You can replace any of the 38 existing patterns by patterns that you yourself design. This is done either by pulling down the Goodies menu and choosing the **Edit Pattern** command, or by double-clicking the particular pattern that you wish to replace. Let's imagine that we want to replace one of the vertical-striped patterns, as shown in the following illustration:



The square on the left contains an enlarged fragment of the pattern that you see in the square on the right.

The black and white points in the square on the left are changed by pointing at them and clicking them. As the points are modified, the overall pattern evolves at the same time.

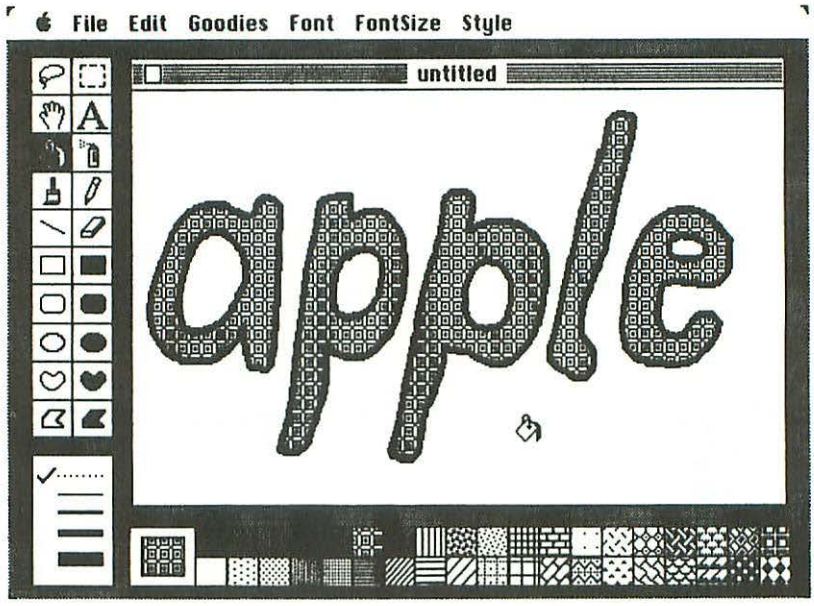
The final result might be as shown in the following illustration:



Click the **OK** button to confirm that we wish to retain this new pattern, which we shall now use to do some painting. The new pattern immediately replaces the old one in the box down at the bottom of the screen.

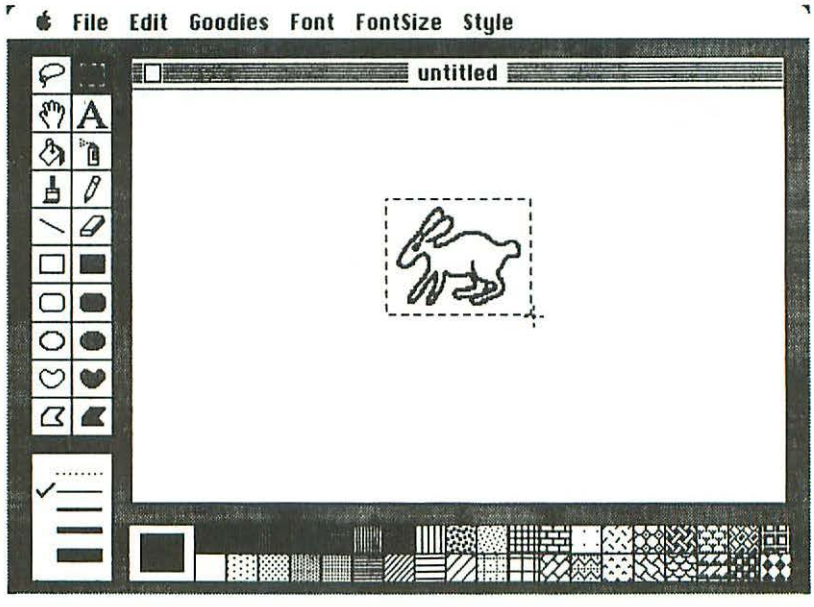
There's another tool called the *area fill* — represented by a pot of paint, just to the left of the spray paint icon — that can be used for painting.

First of all, let's draw the word "apple" (see following illustration), using the paint brush. We then select the pot of paint. Now, each time we move the pot of paint onto one of the letters in "apple", and click it, the letter is filled up with the newly-designed pattern.



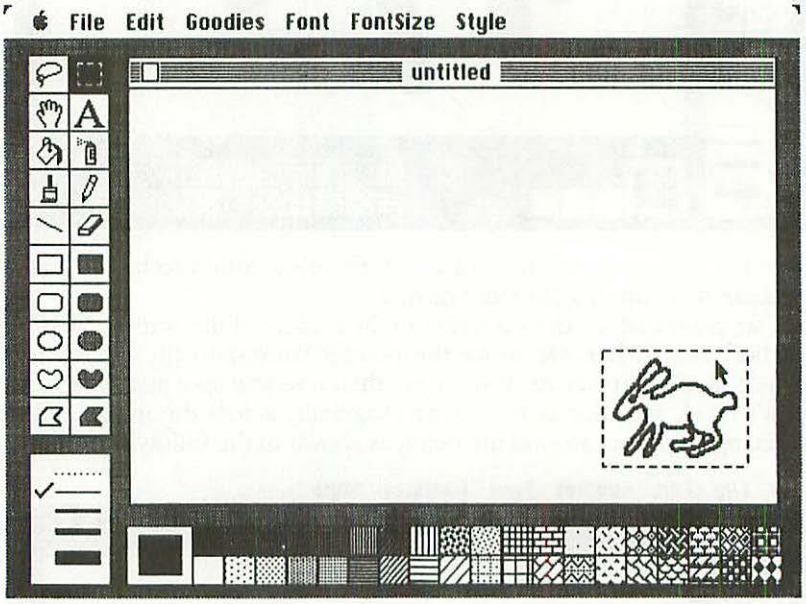
Just above the icon in the form of a letter A, there's a dotted rectangle that is used for the *selection* of part of a MacPaint picture.

Let's suppose that we draw a rabbit in the middle of the screen. Next we click the selection icon. When we move the pointer back onto the main window, it appears as a small dotted cross. If we move this cross to a spot just to the top left of the rabbit's head, and then drag it down diagonally across the image, a flickering dotted rectangle appears around the beast, as shown in the following illustration:

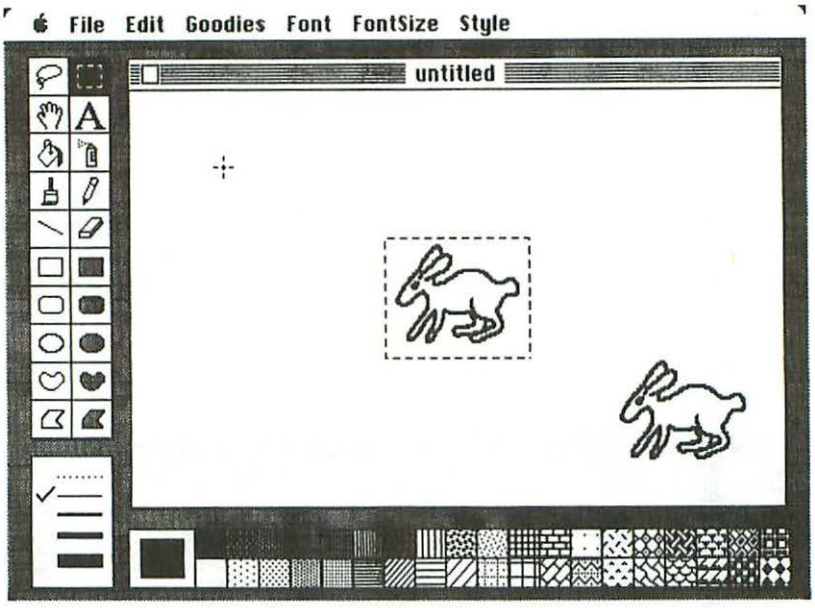


We now say that the rabbit is selected. It may happen that the position of the rectangle does not satisfy you; for example, you might have started the dragging from a position that's too low, and clipped off part of the ears. In that case you simply undo the selection by clicking at some position outside the present rectangle, which disappears instantly. Then you can reselect the rabbit.

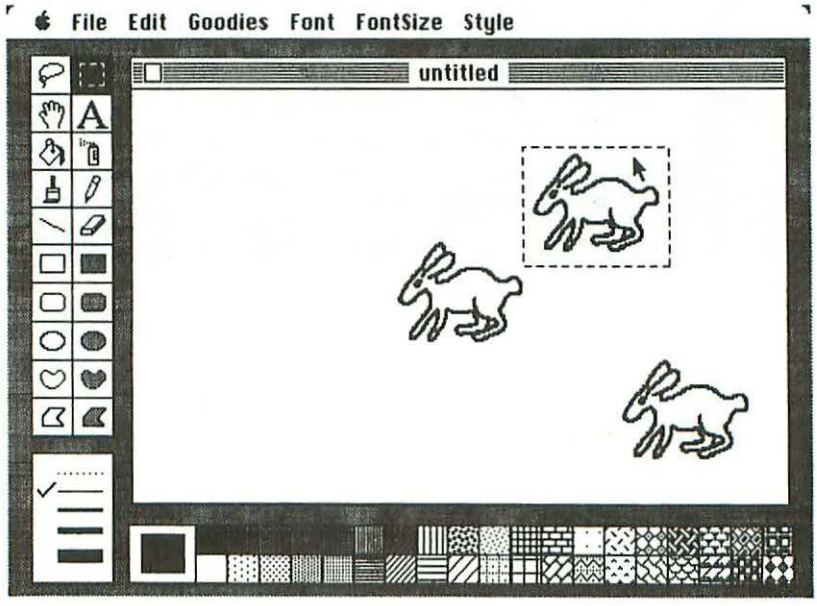
Once you've selected something in the MacPaint window, there are all sorts of things you can do with it. Let's start off with the simplest operation, which consists of shifting the image around on the window. To do this, you move the pointer inside the selection rectangle. In this position (see following illustration), the pointer ceases to be a dotted cross, and becomes an ordinary arrow. Now you can drag the image to any spot you like.



Another interesting thing you can do with a selected image is to use the cut & paste concept that we presented in Chapter 3. To do this, pull down the Edit menu and choose the Copy command. This means that the image of the rabbit is copied onto the Clipboard. Next, undo the present selection by clicking at some place outside the dotted rectangle. Then pull down the Edit menu once again and choose the Paste command. A selected copy of the rabbit reappears in the middle of the window, as shown in the following illustration:

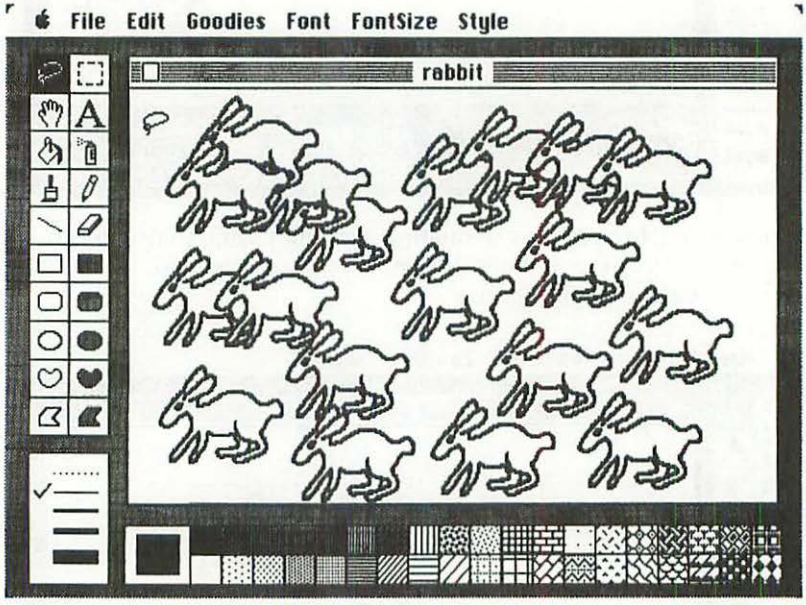


You can however get the rabbits to multiply without having to use the cut & paste technique. Simply hold down the <Option> key as you drag the image. The result is shown in the following illustration:



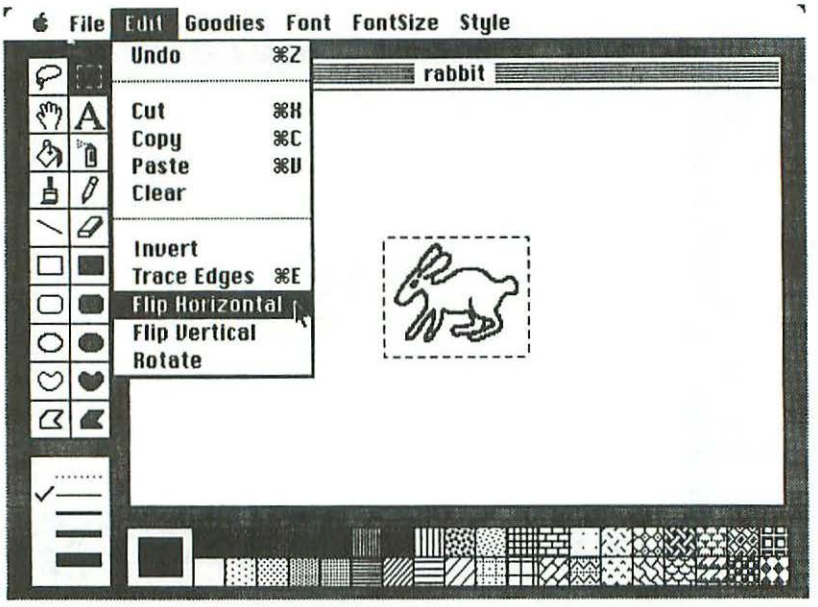
The *lasso*, to the left of the dotted rectangle icon, provides an alternative manner of selecting something in the MacPaint window. Instead of enclosing the selected image in a rectangle, the lasso places a flickering tightened "rope" around the selected item. From that point on, you can operate on the selected image in exactly the same way as for the rectangle.

Let's get back to a single rabbit in the middle of the window. There's still a copy of the original animal on the Clipboard, so all we have to do is to double-click the eraser to clear the window, then pull down the Edit menu and choose the Paste command. Next, let us select this solitary creature with the lasso. The outline of the rabbit, once selected, starts to shimmer in a curious manner, almost as if it were animated. Now we'll drag a copy of the selected image to various parts of the window, while holding down the <**Option**> key, as in the previous example. The result is shown in the following illustration:

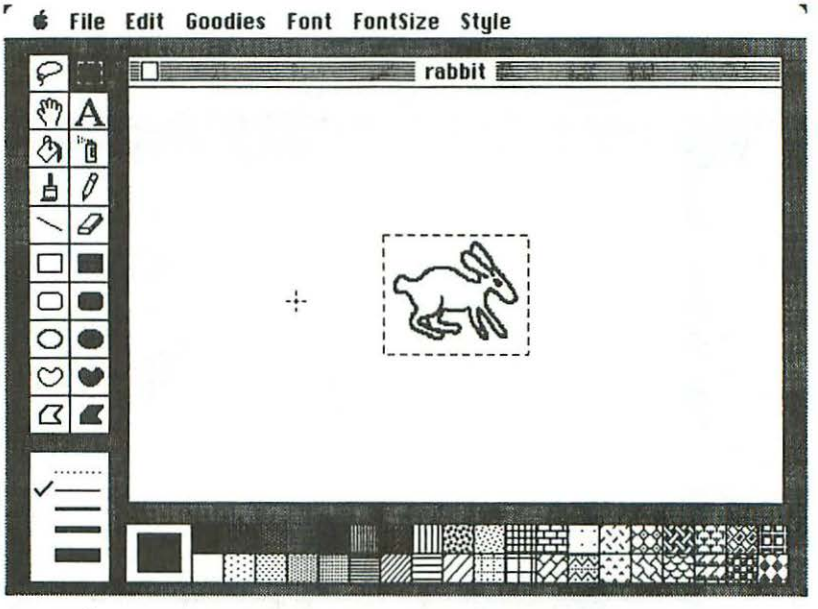


Let us double-click the eraser to get rid of this multitude of mammals, and return to a single specimen in the middle of the screen. It remains selected by a rectangle, because that's the way we copied it into the Clipboard earlier on.

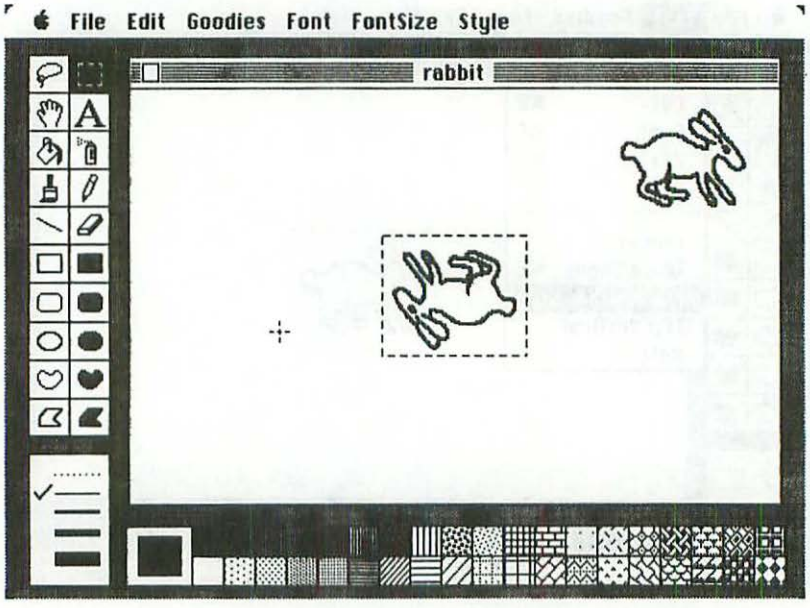
We can pull down the Edit menu and choose the **Flip Horizontal** command, as shown in the following illustration:



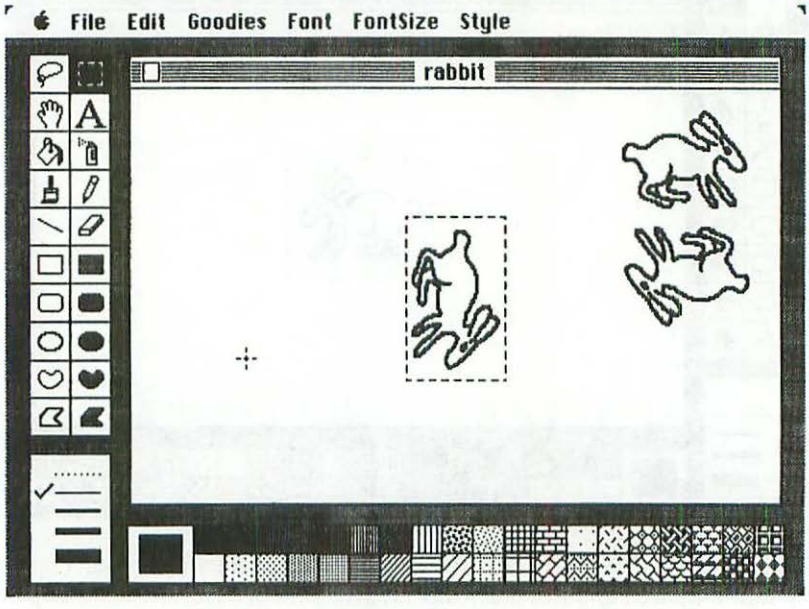
Here's the result:



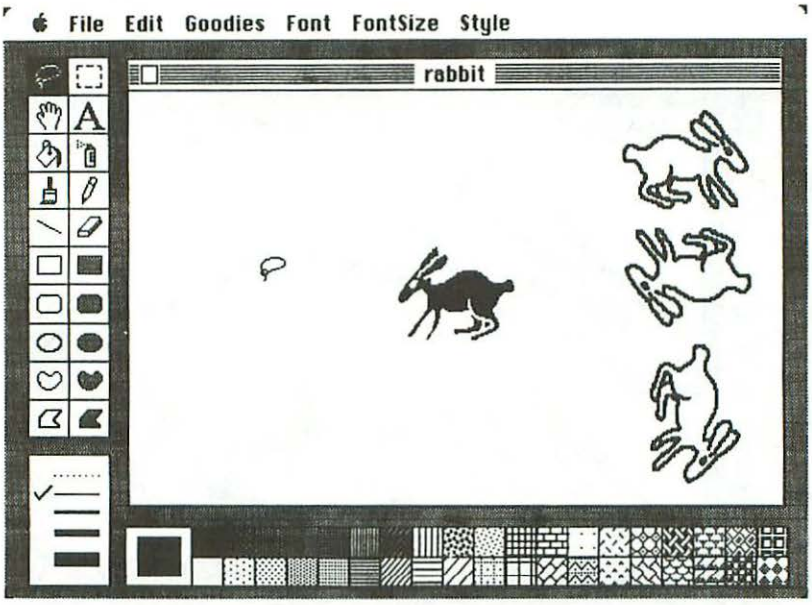
Let's drag him up to a corner of the screen, and paste in another copy of the original rabbit in the center of the window. This time we'll choose the **Flip vertical** command from the Edit menu, which produces the following result:



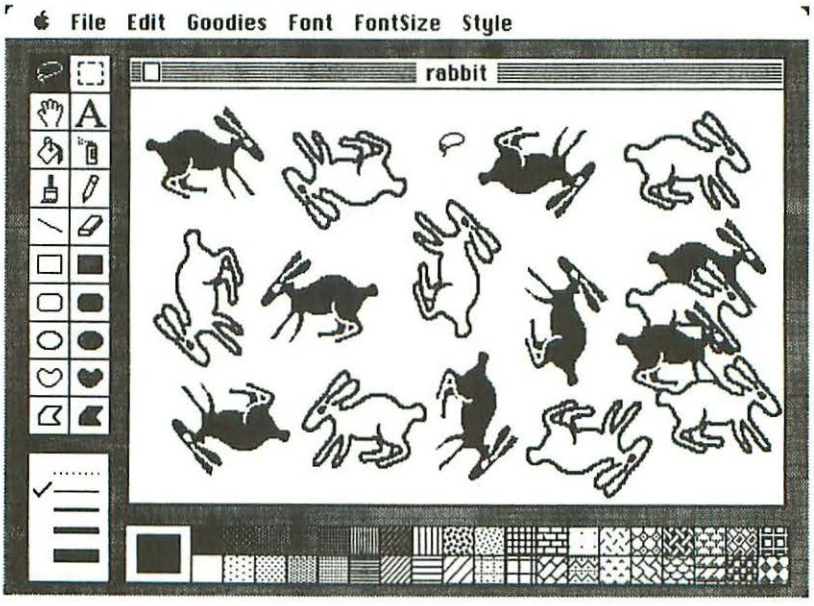
The <Option> key allows us to drag a copy of this inverted rabbit across to the right, then we'll reselect the one in the middle of the window and ask for the **Rotate** command from the Edit menu. The result is as follows:



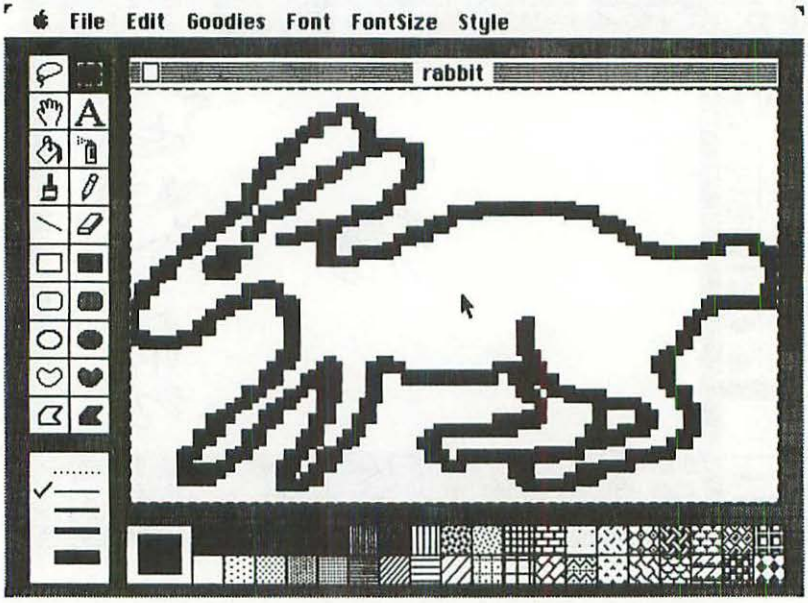
Having dragged this new specimen across to the right, let's paste in another copy of the original. This time we'll reselect him with the lasso instead of the rectangle, and then ask for the **Invert** command from the Edit menu. The result is a sleek black-coated species, as exhibited in the following illustration:



Using a random assortment of all these techniques, we can arrive at the sort of result shown in the following illustration:



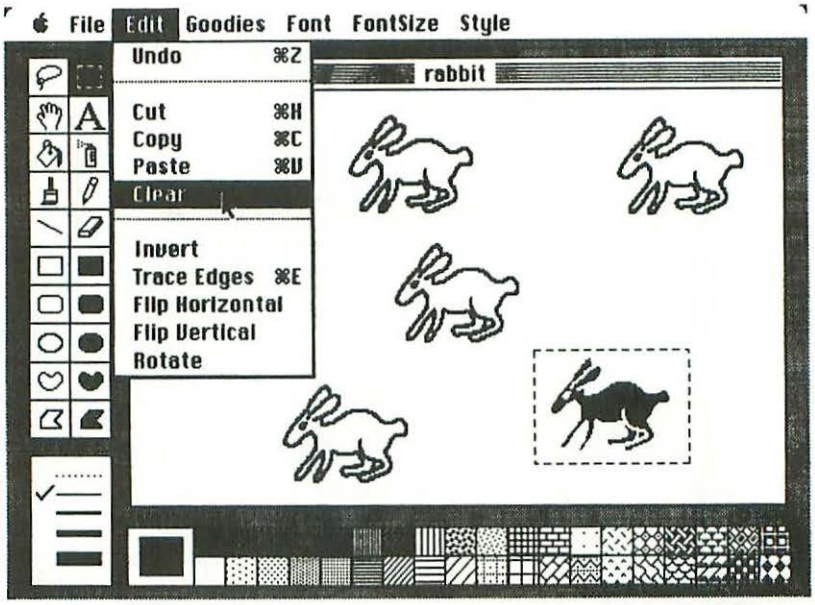
We come to yet another MacPaint possibility that is based upon the concept of selection: *stretching* and *shrinking*. Once again, let us erase the window and paste in a copy of the original rabbit. To change the dimensions of the selected image, you simply hold down the <⌘> (command) key while you carry out dragging operations. The following illustration shows us, for example, a stretched version of the image:



When you stretch or shrink an image in this manner, it is likely to become distorted with respect to the original form. If you wish to avoid this distortion, hold down both the <Shift> key and the <⌘> (command) key as you carry out your dragging.

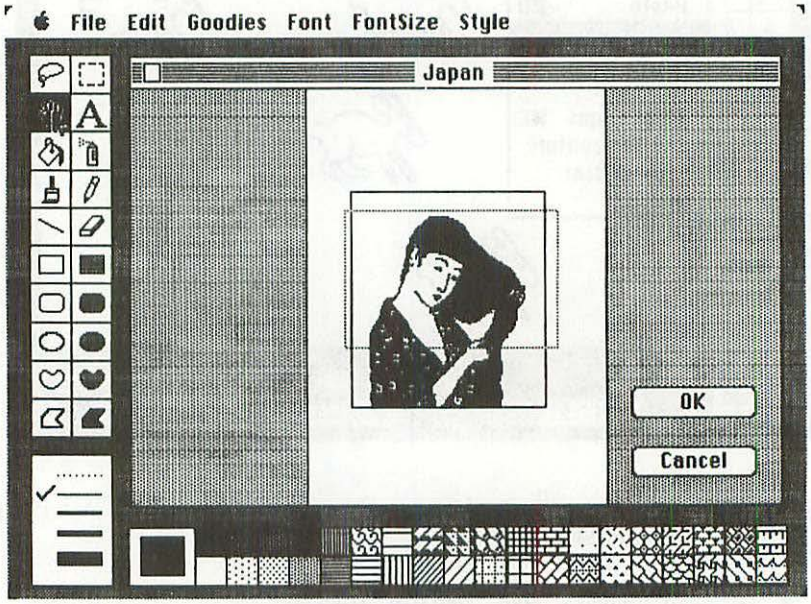
There is still another possibility that involves selection using either the lasso or the rectangle: the **Clear** command. Like Cut, it erases selected images, but it does *not* put the erased image onto the Clipboard.

In the following illustration, we have selected the black rabbit, and the Clear command is about to be used to remove this somber animal from the MacPaint window:



Let us abandon our rabbits now, and turn to the splendid sample of "MacPainting" that is shown in the following illustration:

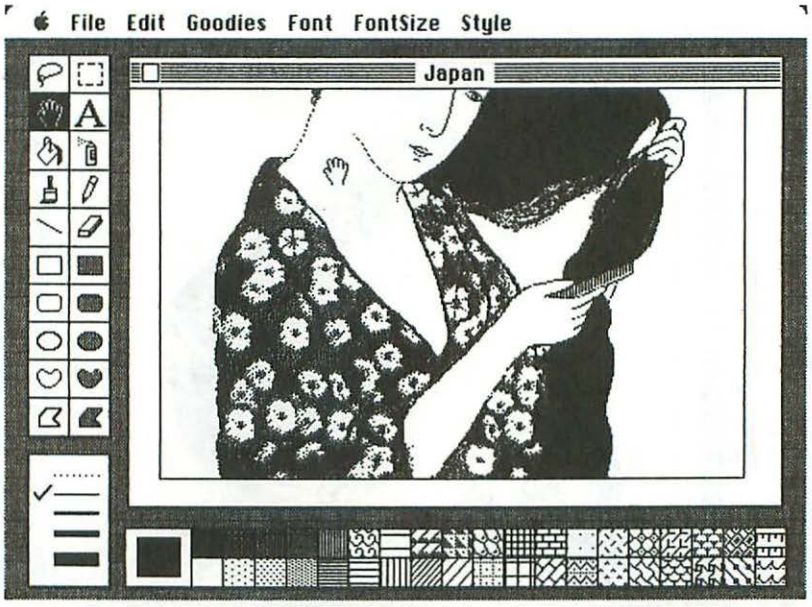
The MacPaint window provides us, at any particular moment, with little more than half of the total painting. To see a reduced version of the entire picture, either pull down the Goodies menu and choose the **Show Page** command, or simply double-click the hand-shaped icon that is located just below the lasso. The result is as follows:



There's a dotted rectangle around the part of the picture that is currently displayed in the MacPaint window. If you suddenly decide that you would like to erase the entire picture, all you have to do is to position the pointer just to the right of the image, and drag the picture away to the right.

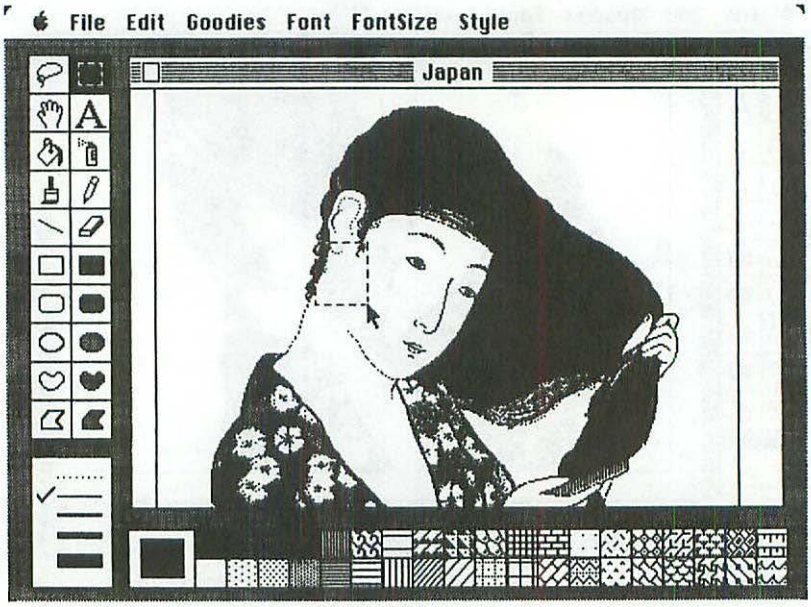
If, on the other hand, you wish to return to the normal MacPaint window, then click the **OK** button.

The hand-shaped drawing tool that we have just mentioned is in fact a *scroll* device. If we click it, we can then drag different portions of the entire painting into the window, as shown in the following illustration:

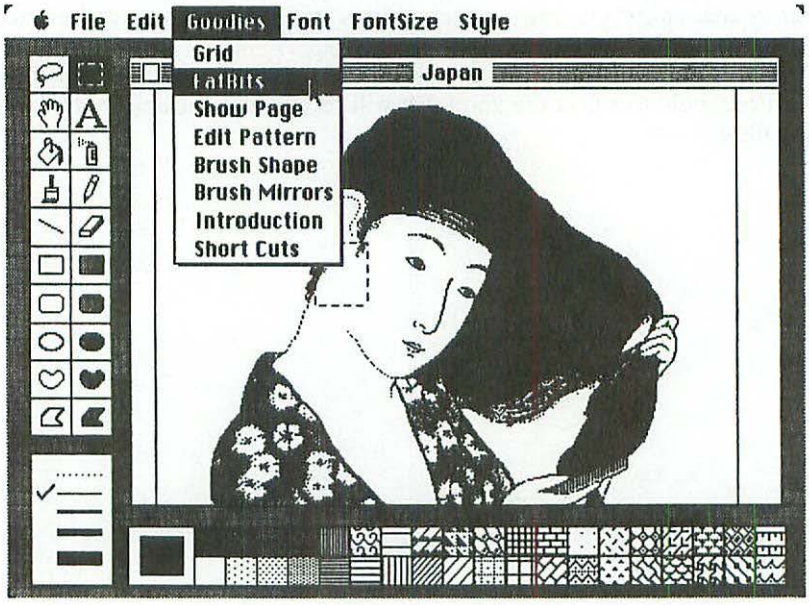


The final MacPaint feature that we are going to examine is referred to, in the Goodies menu, as *FatBits*. (Only someone from Apple Computer could have got away with such a curious term. The serious employees of a typical international corporation manufacturing business machines would have surely preferred an expression such as "enlargement facility" or "zoom".)

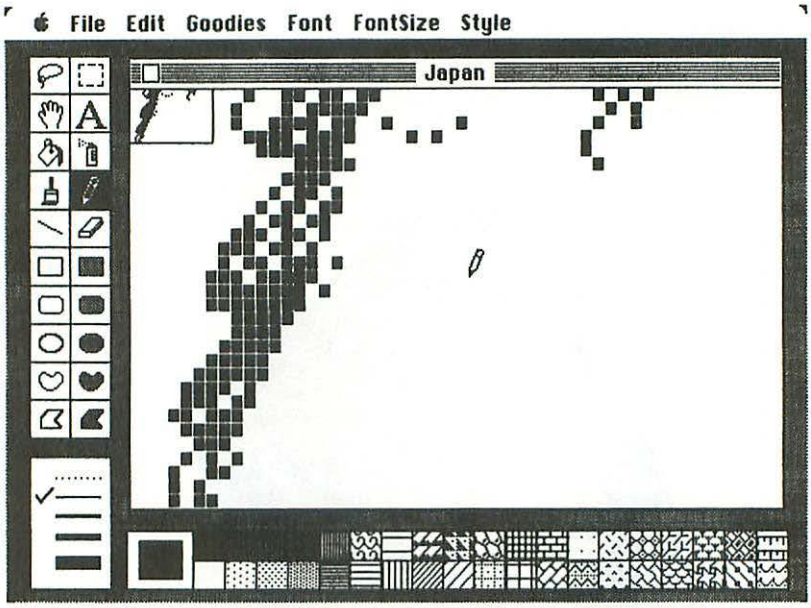
What we want to do is to put an earring on our lovely Oriental. Start by using the dotted rectangle to select the zone that will receive the earring, as shown in the following illustration:



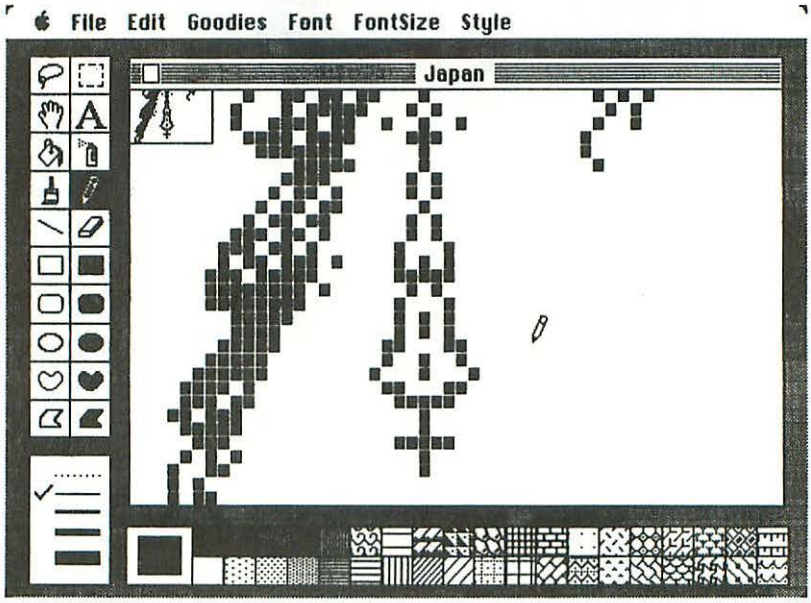
Now pull down the Goodies menu (see following illustration) and choose the FatBits zoom facility. The same result could have been obtained by double-clicking the pencil icon.



The following illustration shows a blown-up point-by-point version of the selected rectangle, with a normal-sized reproduction of the same zone in the upper left corner of the window:



Use the familiar pencil to click in the black and white points that represent an elegant earring, as shown in the following illustration:

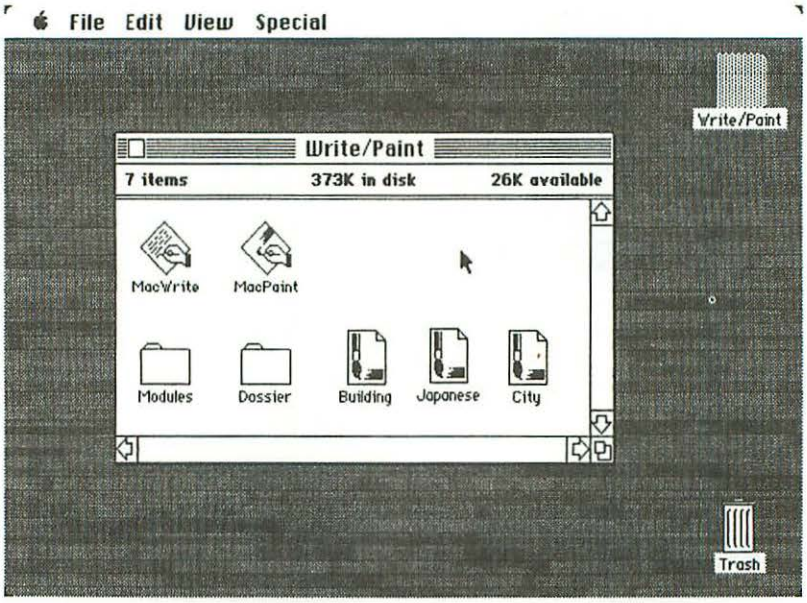


To get back to the normal window, either double-click the pencil icon, or simply click the small rectangle in the upper left corner of the screen that contains a normal-size representation of the selected area.

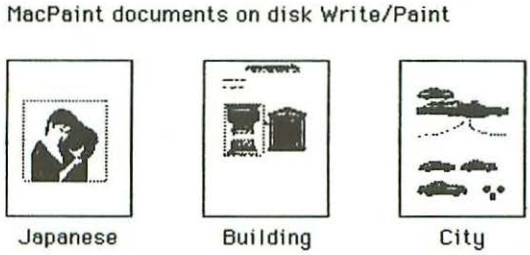
To obtain a hard copy of a MacPaint document, pull down the menu and choose the **Print** command. The result is shown in the following illustration:

It's a lovely earring, and the graphics designer is attracted by the temptation of creating an equally-esthetic MacPaint necklace to decorate the bare bosom of the young lady. But let's leave beauty aside, and get on with the subject. . . .

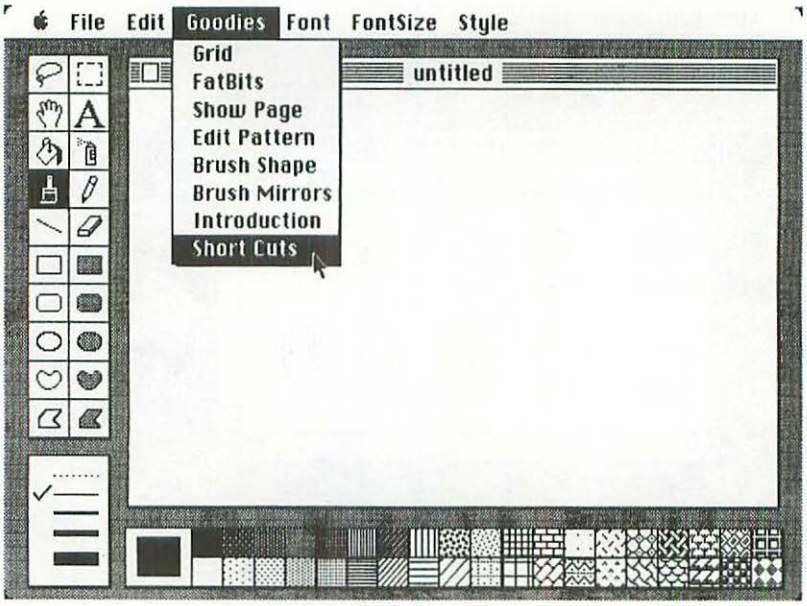
Let's imagine that you have a "Write/Paint" disk with the following directory:



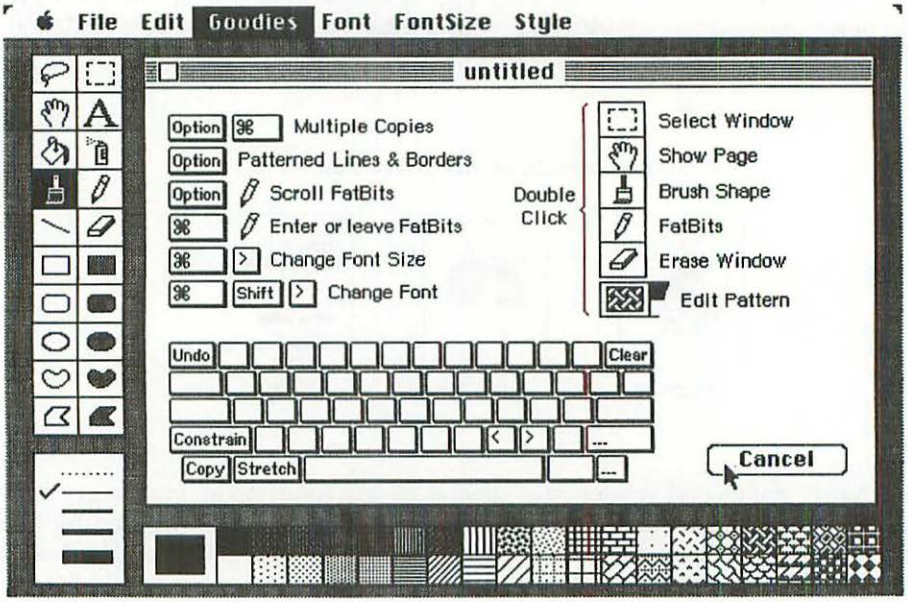
The three icons in the lower right corner of the directory window — labeled "Building", "Japanese" and "City" — represent documents created by MacPaint. Now, if we open MacPaint and then pull down the File menu, there's a **Print Catalog** command that enables us to obtain a hard copy of the catalog of MacPaint documents, in a reduced form, as shown in the following illustration:



We arrive at the end of our presentation of MacPaint. Apart from such minor subjects as the **Brush Mirrors** command in the Goodies menu (which enables you to produce symmetrical shapes), practically all that remains is to take a look at the **Short Cuts** command in this same menu, as indicated in the following illustration:



It tells you (see following illustration) how to carry out a dozen or so typical MacPaint operations in the most rapid fashion.

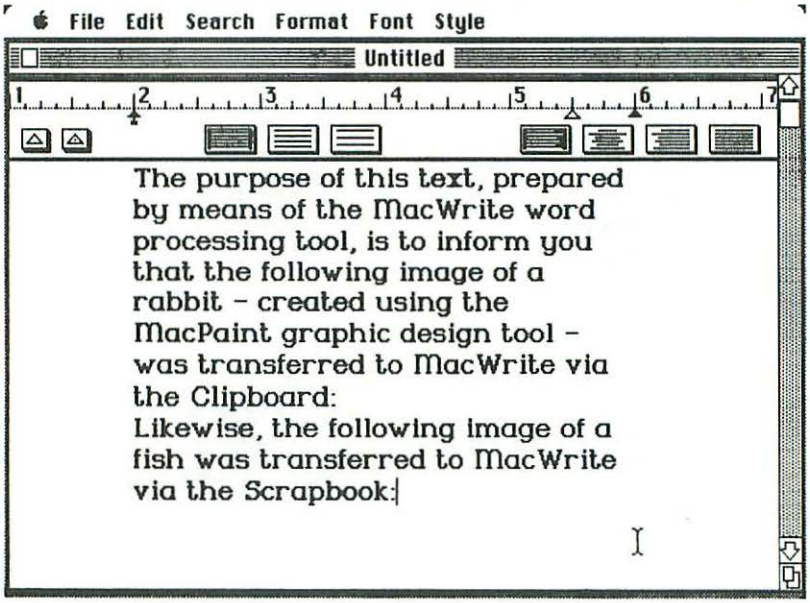


For example, the key in the top left corner of the Macintosh keyboard offers a shortcut that functions in the same manner as the **Undo** command in the Edit menu: it allows you to rub out your most recent painting operation. (If you use this undo function a second time, it puts back what was just rubbed out.)

Write/Paint relationships

One of the most remarkable features of Macintosh — as we have pointed out several times already — is that there is no conceptual barrier between text-based activities and image-based activities.

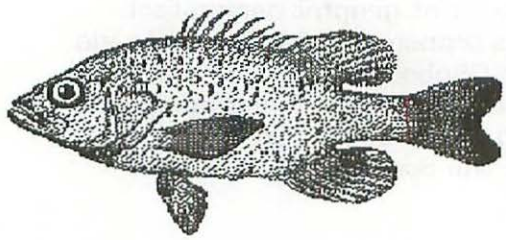
Let us return to MacWrite for an instant, to produce the following text:



When we actually call upon the Clipboard and Scrapbook to insert the appropriate images, here is the result, printed out by MacWrite:

The purpose of this text, prepared by means of the MacWrite word processing tool, is to inform you that the following image of a rabbit - created using the MacPaint graphic design tool - was transferred to MacWrite via the Clipboard:

Likewise, the following image of a fish was transferred to MacWrite via the Scrapbook:



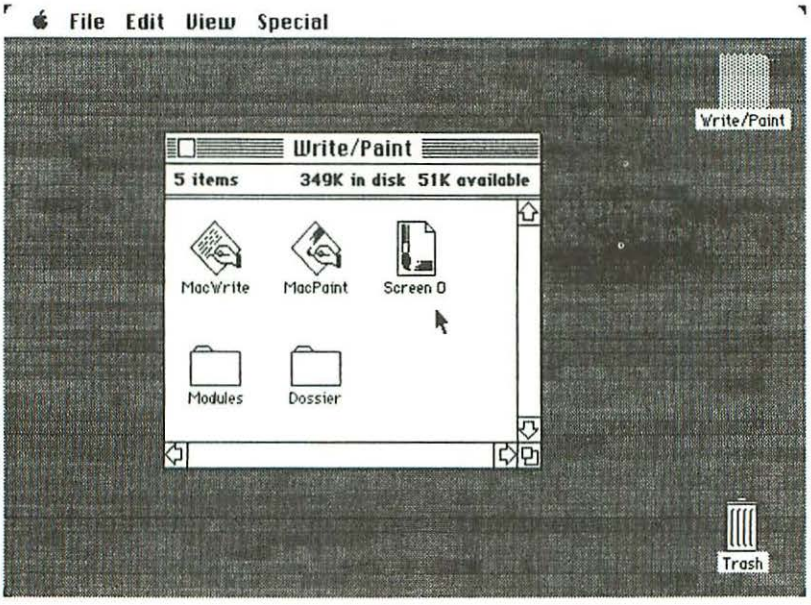
To put a MacPaint picture into the Scrapbook, you have to cut and paste it via the Clipboard. Likewise, to insert a Scrapbook image into a MacWrite document, you have to copy and paste it via this same Clipboard. As pointed out already, every cut & paste operation on Macintosh involves this famous Clipboard.

Screen dumps and snapshots

Since we've brought up the subject of images inserted into textual documents, maybe it's time to explain to inquisitive readers just how the various illustrations in this book were obtained.

Naturally, we got them from Macintosh itself, using the Imagewriter printer. But we used two quite different methods, and readers will have no trouble in distinguishing between the two results. Most of the illustrations are simple *screen dumps*, which are obtained by pressing, first, the <**Caps Lock**> key, and then, simultaneously, the following three keys: <**Shift**> , <⌘> and the number 4. If you happen to be holding down the mouse button to obtain the display that you wish to print out, then the screen dump will be printed as soon as you release the button. (One feels, at times, that Macintosh was designed ideally for people with at least three hands!)

But certain illustrations have been obtained using the quite remarkable technique of so-called *snapshots*. The general idea is that it is possible at most times to store away, on disk, an image of the current screen display. To do this, you merely press the following three keys: <**Shift**> , <⌘> and the number 3. The contents of the screen are then stored as if they were a MacPaint document. The first snapshot is labeled "Screen 0" (see following illustration, showing the "Write/Paint" directory), and you can store up to ten of them on the disk. Each of these snapshots can be "touched up" using classical MacPaint facilities, making it possible to add in textual explanations and all sorts of highlighting gadgets such as arrows and boxes.



Once you've finished processing a snapshot with the MacPaint tool, and obtained a nice high-resolution hard copy, you can simply throw the disk-based snapshot document into the trash can, making room for further snapshots.

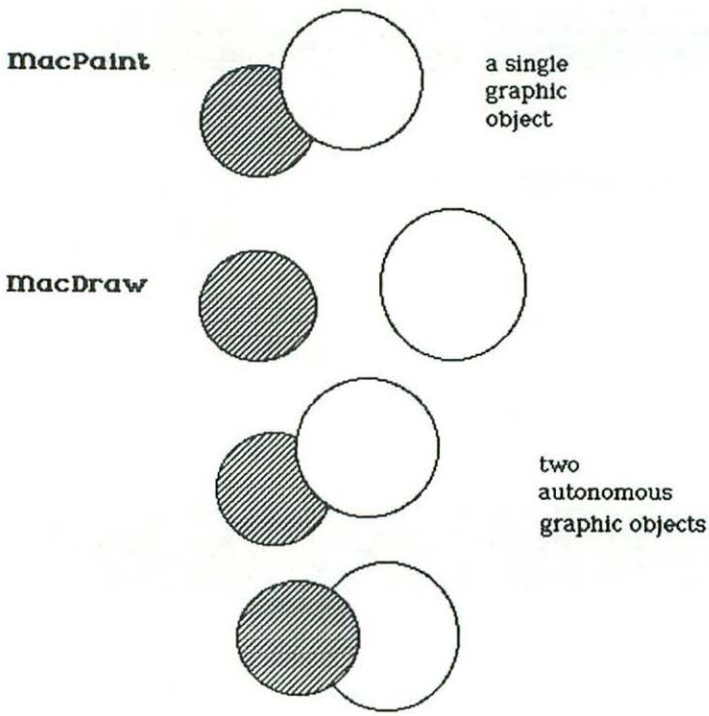
There is no doubt about it that Macintosh, thanks to this snapshot technique, is the first personal computer that is capable, narcissistically, of handling its own image.

chapter 7

"MacDraw" — Graph Processing

There are many superficial similarities between MacPaint and MacDraw, but the two products function quite differently, and they are designed to carry out different types of tasks.

Let us imagine an example. In MacPaint, suppose that you create a striped circle that is then partly hidden by an an overlapping white circle (see following illustration). The two circles are henceforth blended into a *single* image, and there is no way of "removing" the white circle in order to rediscover the hidden part of the striped circle.



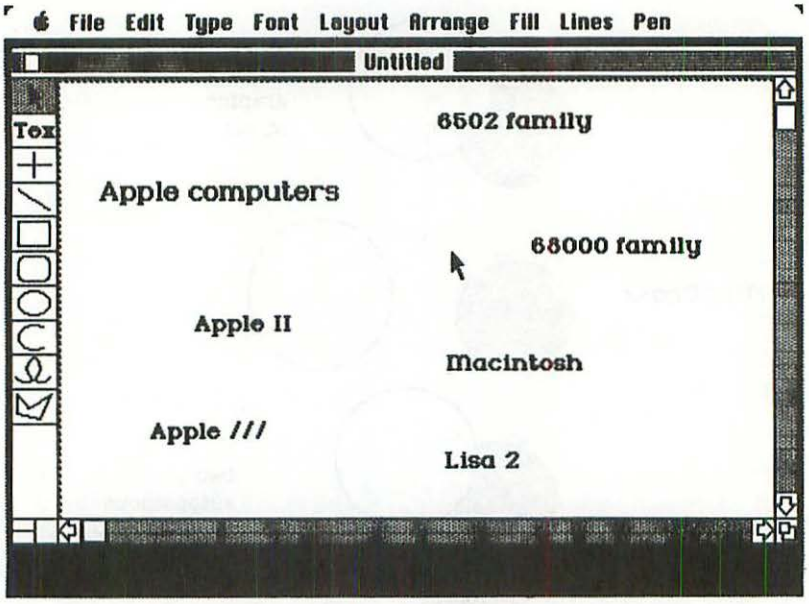
In MacDraw, on the contrary, there is a concept — *graphic objects* — that does not exist in MacPaint. This means that you can continue to consider the two circles as a pair of autonomous objects, even though they may be overlapping for the moment. You can request that the striped circle be placed on top of the white one, or you can simply move the two circles apart on the screen so that both of them are completely visible.

This idea of being able to refer specifically to autonomous elements of a drawing brings us closer to the manner in which most simple technical illustrations are created, and the expression "graph processing" — although it might not be entirely appropriate — has been retained to designate this kind of image production.

Sets of objects

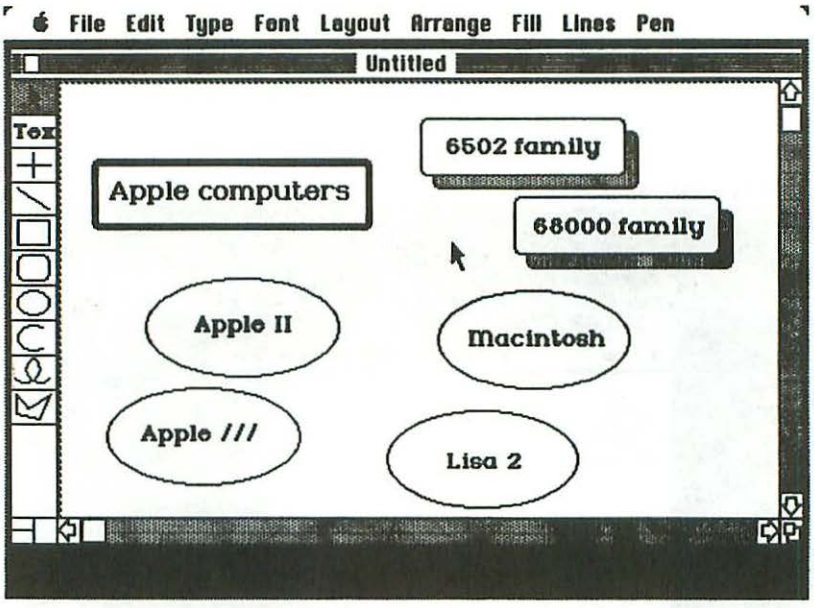
The notion of MacDraw objects can best be demonstrated by means of a simple example.

The following illustration shows us seven short textual elements on an untitled MacDraw window:

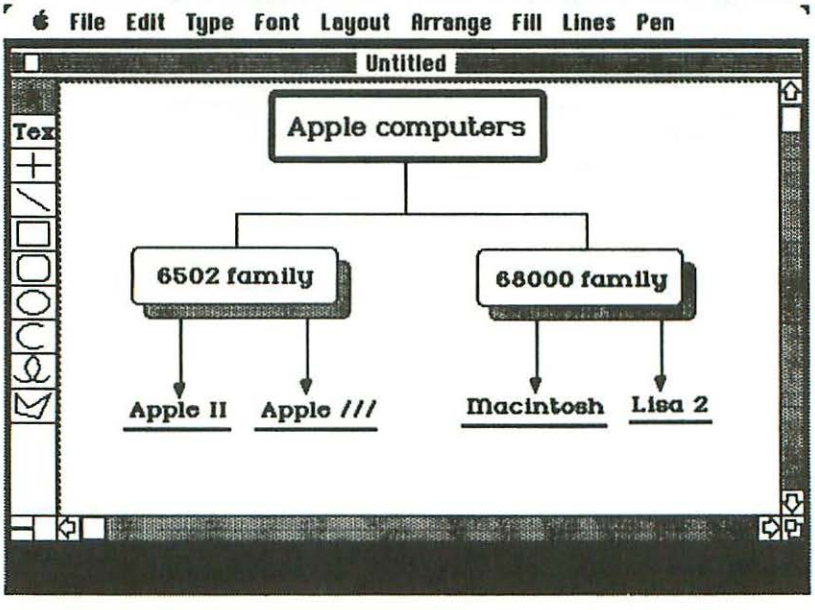


Since these textual elements can be manipulated separately, we can enclose them in all kinds of "boxes" (see following illustration), and we can move all these objects around on the MacDraw window as freely as we like.

Don't forget that, when we enclose a phrase in a box, *both* the phrase and the box can continue to be thought of as autonomous graphic objects. In other words, we can easily break them apart if we so decide.



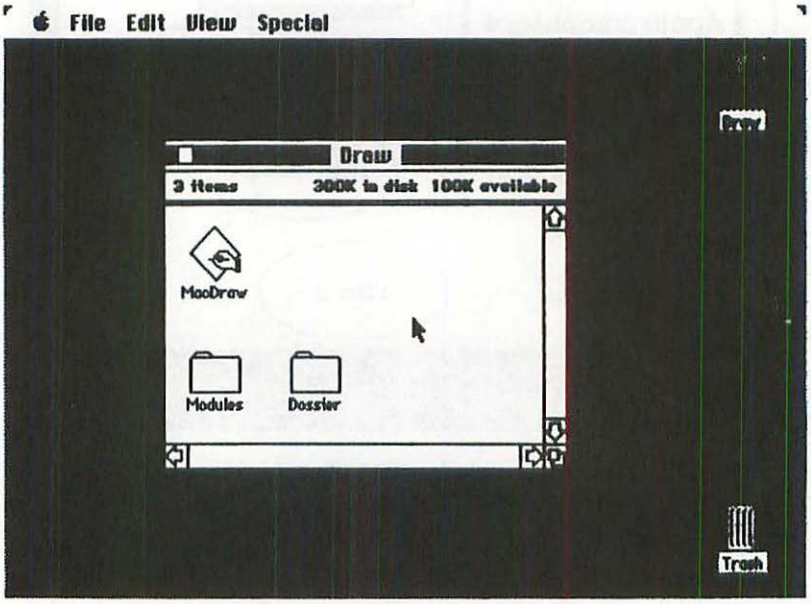
This set of objects can be reassembled quite differently in the MacDraw window, using conventional dragging techniques (see following illustration). Some objects — the ellipses — have been discarded, whereas others — all the lines and arrows — have been added.



So much for this brief example, which was merely intended to provide a rough idea of the sort of diagrams that might be executed by means of MacDraw. Let us now get around to actually using this software tool.

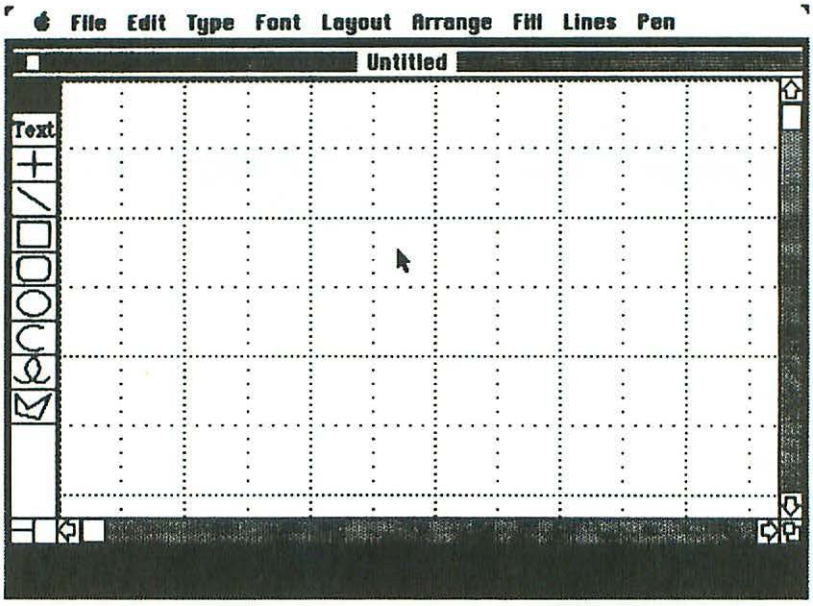
Demonstration

The directory of the "Draw" disk is shown in the following illustration:

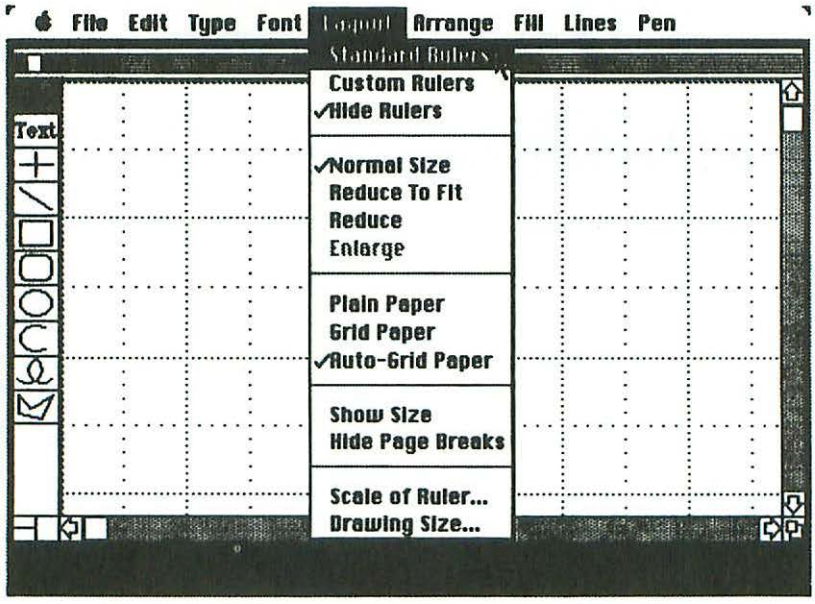


The icon named "MacDraw" is the program itself. "Modules", as usual, is a folder containing half a dozen elements of system software, and "Dossier" is an empty folder that we might use later on for filing away the drawings we create.

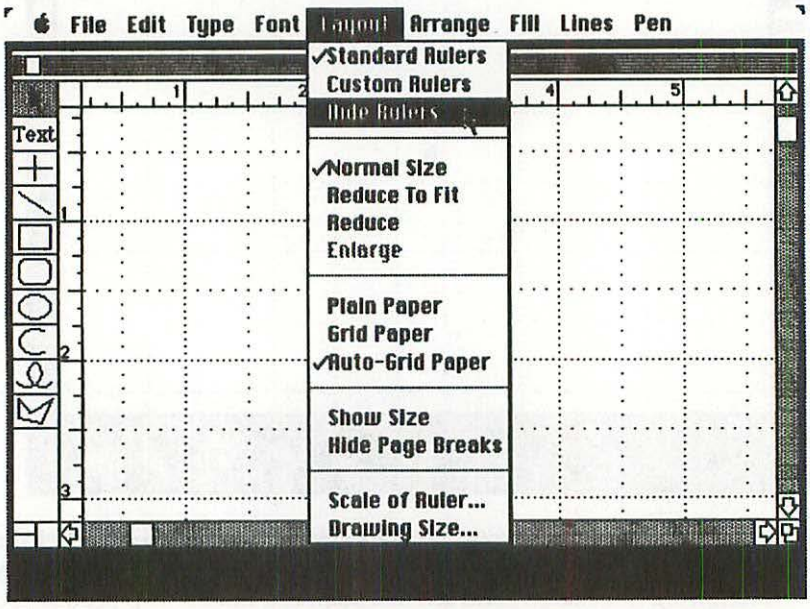
As soon as we double-click the "MacDraw" icon, the screen appears as follows:



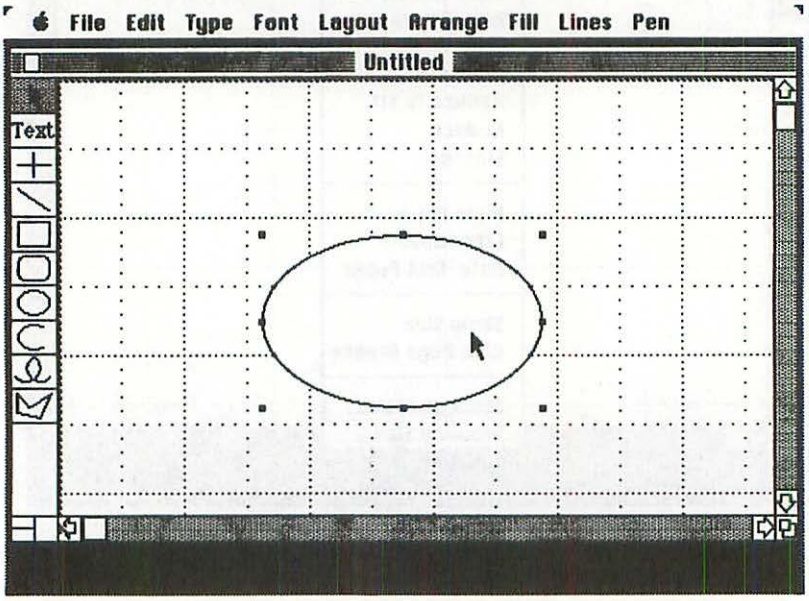
Notice that the main window is divided into a *grid* of rectangles, to help us in laying out graphic objects in an orderly fashion. You can pull down the **Layout** menu (see following illustration) and ask that **Standard Rulers** be placed along the top and left sides of the MacDraw window.



You can use the scroll bars to access a global surface that is eight inches wide and ten inches high. The same menu offers a **Hide Rulers** command (see following illustration) to restore the screen to its original state.

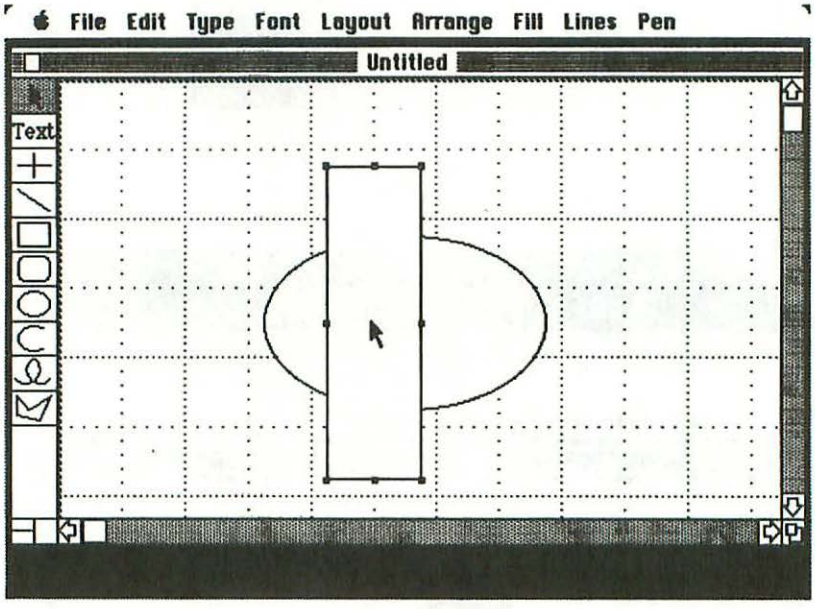


The ten icons to the left of the window designate functions similar to those we encountered in MacPaint. For example, an ellipse can be obtained by clicking the hollow ellipse icon and dragging the pointer over the window, as shown in the following illustration:

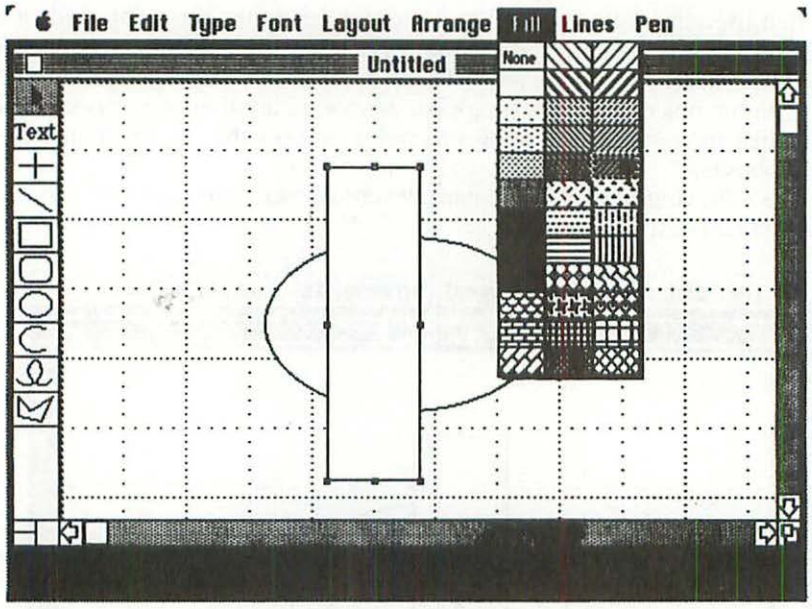


Notice that the ellipse is enclosed in a rectangular perimeter composed of eight black points. They are referred to as *handles*, because they enable us to manipulate the elliptical image in several ways. These handles are displayed on any newly-created graphic object, but they disappear as soon as another object is selected. At any particular moment, these handles reappear on only the most-recently selected object or objects.

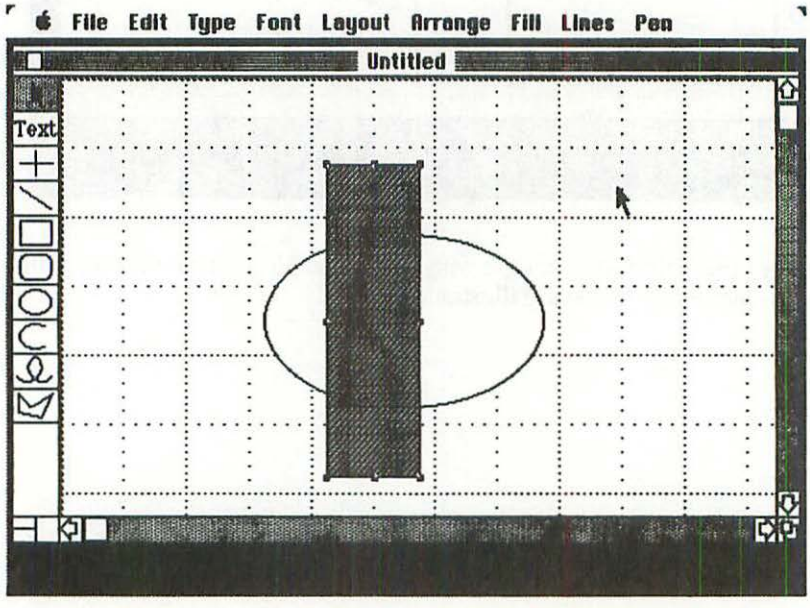
In the following illustration, a narrow vertical rectangle has been superposed over the left half of the ellipse:



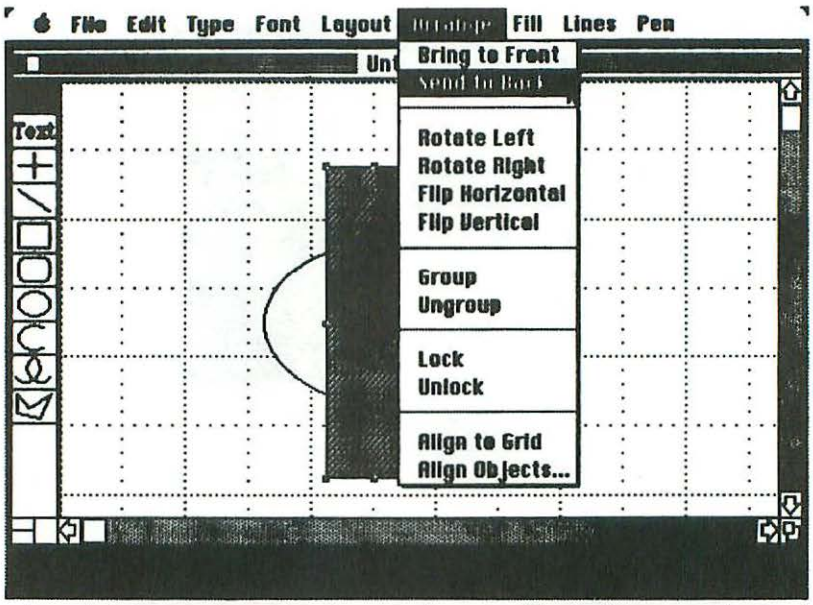
We can paint this selected rectangle with any of the 36 "colors" offered by the **Fill** menu, as shown in the following illustration:



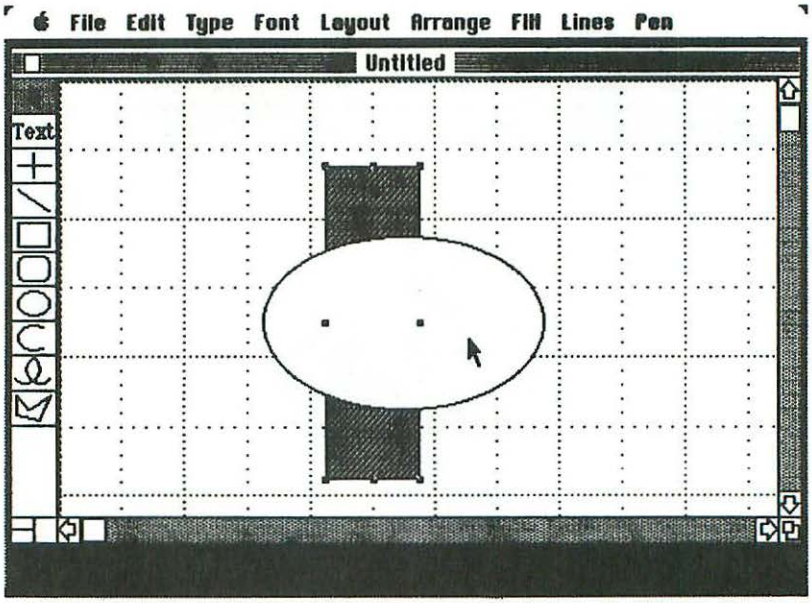
We clicked one of the striped "colors". Here's the result:



In the **Arrange** menu, there's a command called **Send to Back**, as shown in the following illustration:

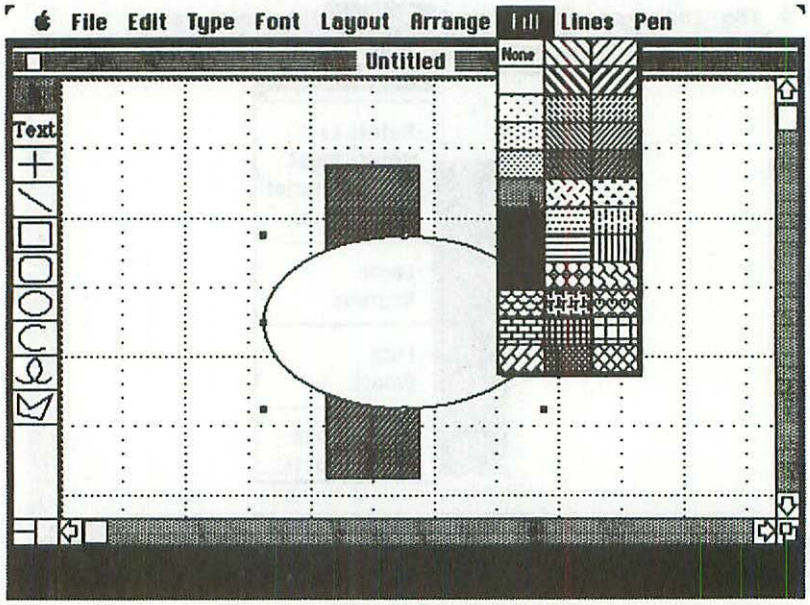


The selected object — the striped rectangle — is effectively sent to the back of the white ellipse, as shown in the following illustration:

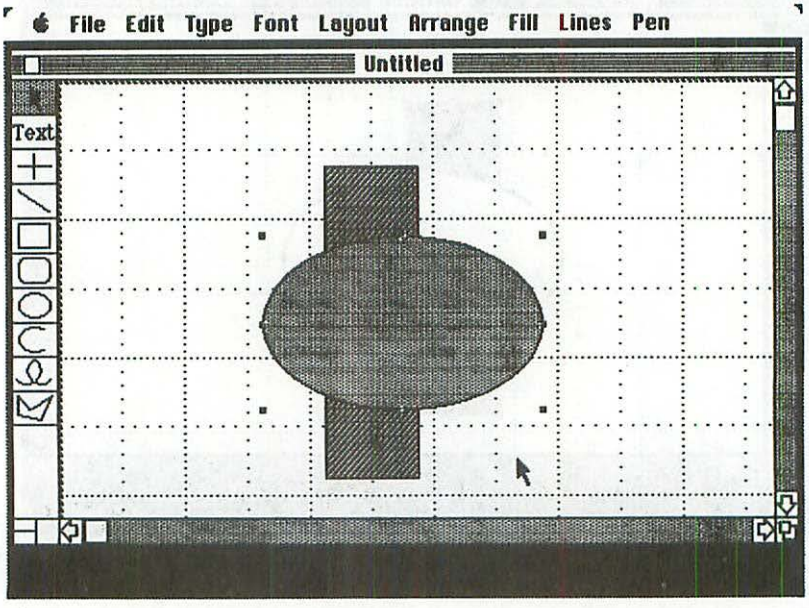


Notice, however, that the rectangle remains selected. Its middle pair of handles remain on top of the ellipse.

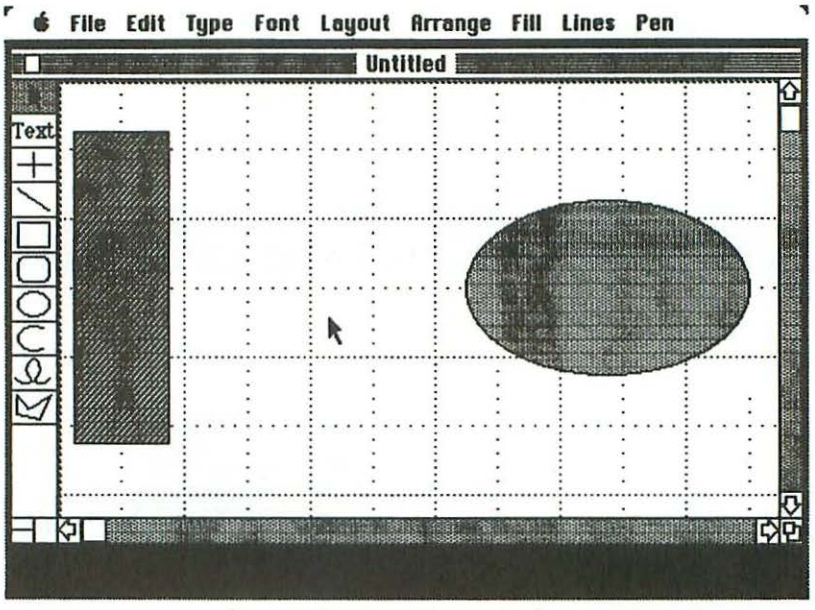
Now we'll click inside the ellipse, to select it, and choose another "color" as shown in the following illustration:



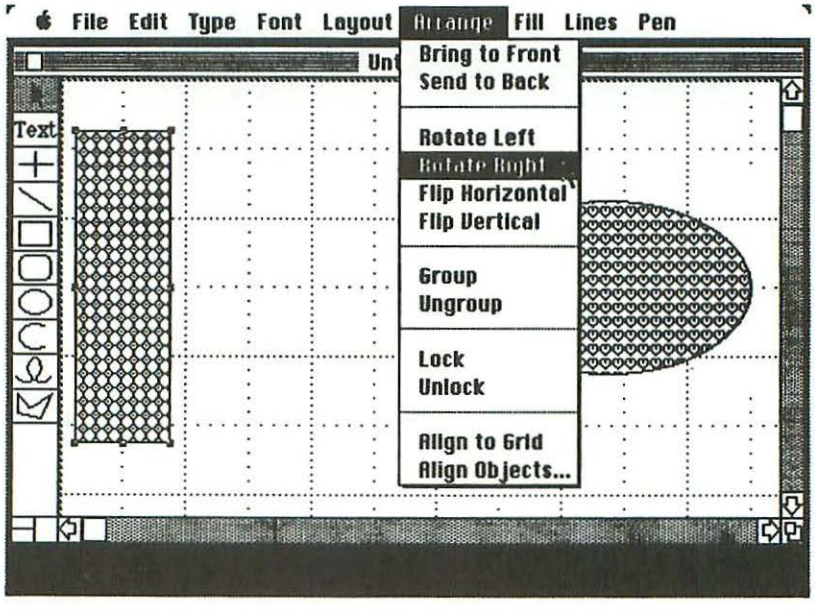
Here's the result:



We can drag the two objects apart if we like (see following illustration). To do this, you simply position the pointer on the object and drag it.

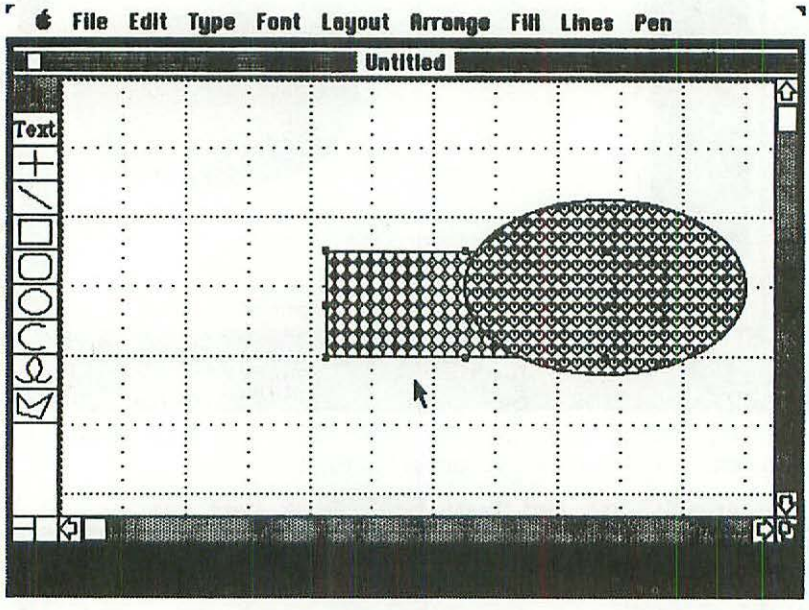


In the Arrange menu (see following illustration), you have **Rotate** and **Flip** commands similar to those that we saw in MacPaint.



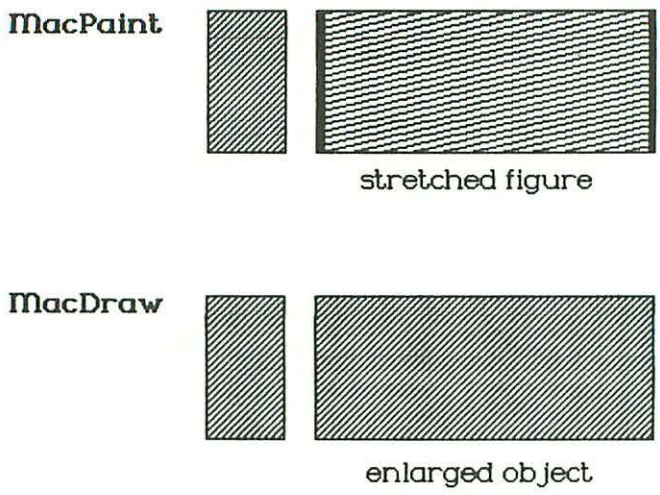
Notice that we have painted the objects with different patterns. This "color" change could *not* have been carried out quite so easily in MacPaint. In MacPaint, if you place the pot of paint on a striped circle and attempt to paint it black, the program will paint only the space between the pair of stripes that are touched by the icon, because they form a closed boundary and MacPaint cannot "understand" that you want to paint the entire circle.

The following illustration shows the result of rotating the rectangle and placing it behind the ellipse:



All our examples are composed of simple figures that could be printed out (using the **Print One** or **Print** commands from the File menu) on a single sheet of paper. But the **Drawing Size** command of the Layout menu makes it possible for a single MacDraw document to extend over dozens of adjacent sheets of paper.

What in fact are the handles used for? As you might have guessed, you can drag them to enlarge or reduce the dimensions of the object. Here again, this enlargement or reduction is quite different to MacPaint-style stretching and shrinking. For example, if you stretch a figure with narrow stripes in MacPaint, the stripes change their appearance. In MacDraw, on the other hand, if you enlarge an object that is filled with narrow stripes, you end up with a bigger object that is filled with exactly the same narrow stripes. These differences are shown in the following illustration:



Since there is no eraser device in MacDraw, unwanted elements of a drawing must first be selected and then removed by means of the **Clear** command in the Edit menu.

It's almost a pity — whatever that means! — that MacDraw will probably always be compared with its sophisticated younger cousin named MacPaint. (MacDraw is a direct descendant of LisaDraw, whereas a forerunner to MacPaint did not exist in the good old Lisa days of several months ago.) The truth of the matter is that both products are excellent tools for making pictures . . . even though it would be dishonest not to acknowledge the fact that MacPaint is a far more user-friendly tool than MacDraw.

Rather than trying to choose between MacPaint and MacDraw, the more intelligent approach is to exploit *both* tools, together, along with MacWrite. In that vein of thinking, let's conclude this all-too-rapid overview of MacDraw with an interesting demonstration of the procedures for carrying out a Scrapbook swap between this product and MacWrite.

Scrapbook swapping

It has already been pointed out that disk-to-disk operations with Macintosh are frankly weird, because the machine takes complete control of the situation and starts ordering you to swap disks like a robot, without bothering to inform you what it's actually all about. It's disconcerting in a comical way, but not really annoying. . . .

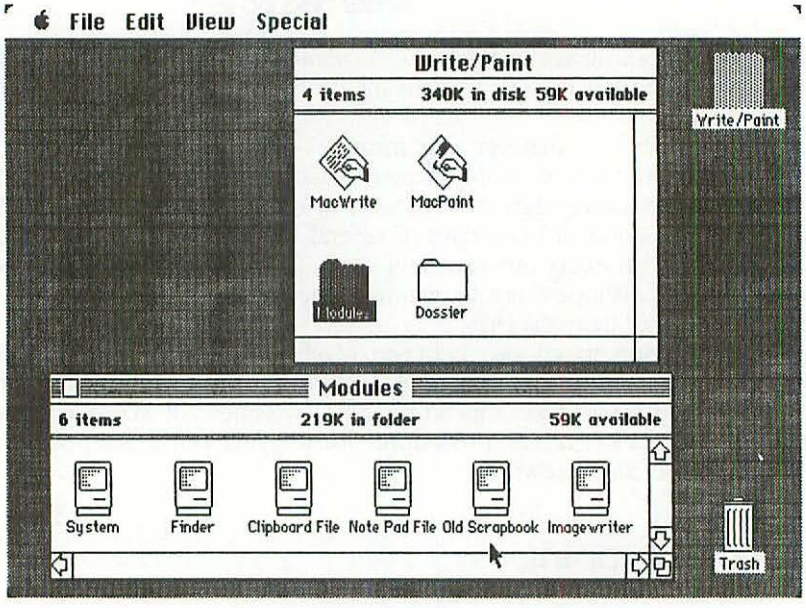
As long as you carry out your cut & paste operations via the Clipboard, there's no problem whatsoever in changing from one disk to another, because the Clipboard is a global desktop device that is not attached to any specific disk.

Things are more complicated if you want to use the Scrapbook created on one disk as a source of data for a tool on a second disk. What we intend to do now, by way of an example, is to make a MacDraw figure and store it in the Scrapbook on

the disk named "Draw", then transfer this Scrapbook to the disk named "Write/Paint", so that we can finally use MacWrite to produce a text that incorporates the image created by MacDraw.

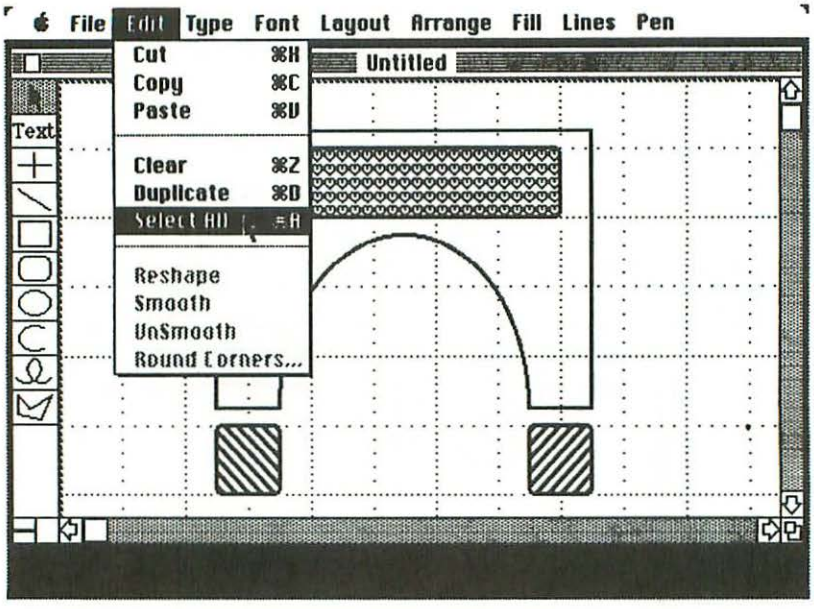
We have to start out by a bit of forward thinking at the level of the target disk, "Write/Paint". In its "Modules" folder, this disk already contains an object called "Scrapbook File". Now, if we attempt later on to transfer a *second* "Scrapbook File" onto this disk, there'll be a names conflict . . . so we'd better do something right away to avoid this problem.

The "something" that we can do depends on whether or not there's anything worth keeping in the current Scrapbook on the "Write/Paint" disk. If not, then no sweat: you simply drag "Scrapbook File" into the trash can, and you can be sure that this radical measure will eliminate any potential conflicts about names. On the other hand, if you have sentimental reasons for wishing to retain the "Write/Paint" Scrapbook, then you should at least change its name. "Old Scrapbook" is as good a choice as any, as indicated in the following illustration:

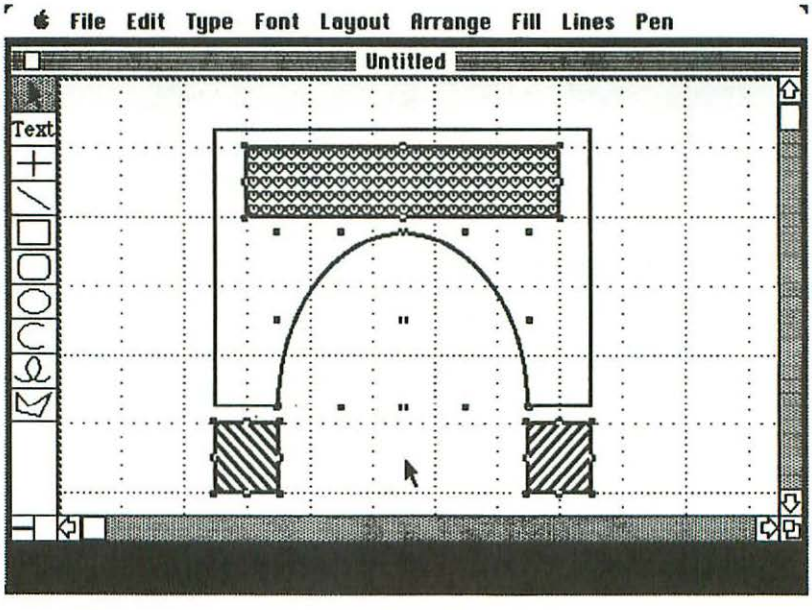


Let's get back now to MacDraw. Napoléon himself would have found it a splendid tool for drawing triumphal arches, because it's easy to respect symmetry.

Once our arch is completed (see following illustration), we use the **Select All** command from the Edit menu to select the entire set of components in the drawing.

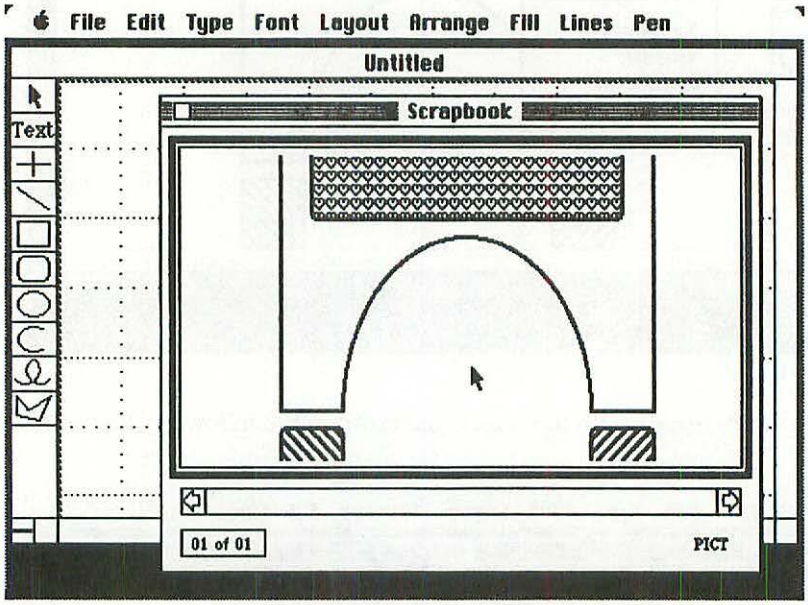


All the handles reappear on the screen, as shown in the following illustration:

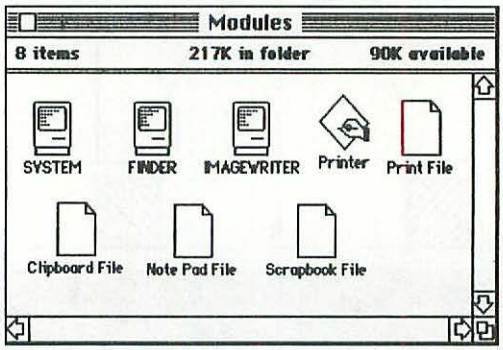


Next, we pull down the Edit menu and use the Copy command to put the arch onto the Clipboard, then we open the Scrapbook and use the Paste command to put a copy of the arch onto the first and only page.

Don't be worried if you find that the triumphal arch appears to have been amputated a bit, both at the top and bottom (see following illustration). That's merely what you might call an "optical illusion", due to the fact that the Scrapbook window is relatively small; rest assured that the entire arch is present on page 1 of the Scrapbook.

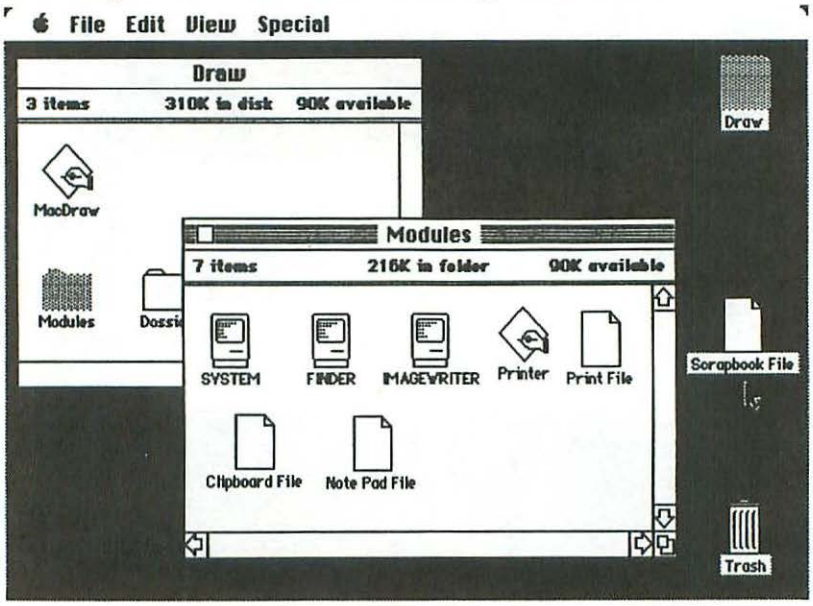


Now, the information that goes into the Scrapbook is in fact retained in a disk file in the "Modules" folder called "Scrapbook File", as shown in the following illustration:

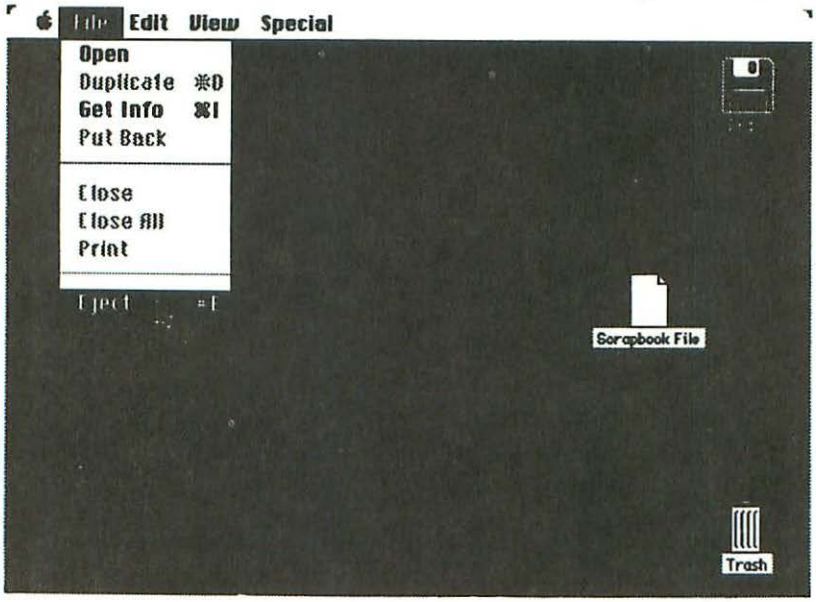


(Notice that the person who developed this product has preferred not to use the Macintosh-shaped icon for the Clipboard, Note Pad and Scrapbook files. This decision is regrettable in the sense that the icon representing a sheet of paper is already used for documents created by MacWrite.)

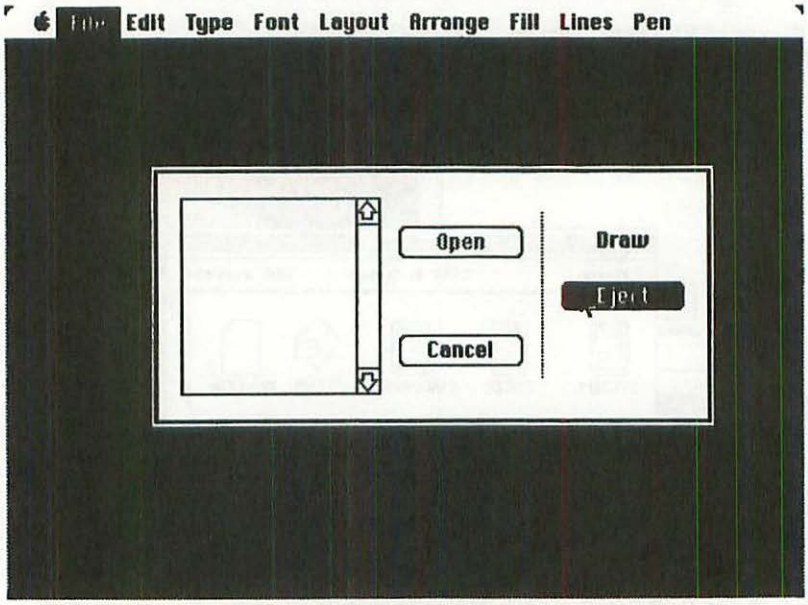
We want to transfer "Scrapbook File" to the "Write/Paint" disk. So, the first thing to do is to drag it over to one side of the desktop, as shown below:



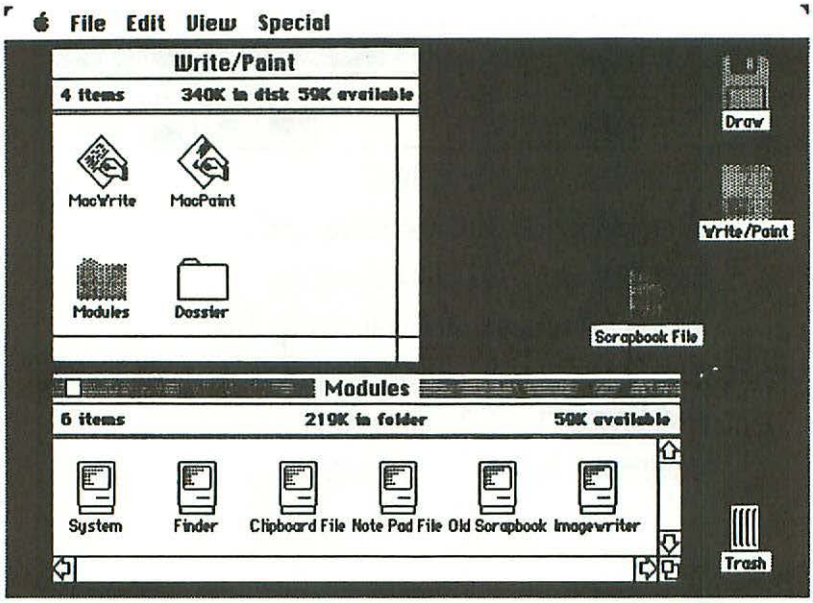
Now we can close the "Modules" and "Draw" windows by clicking their close boxes, and eject the "Draw" disk as indicated in the following illustration:



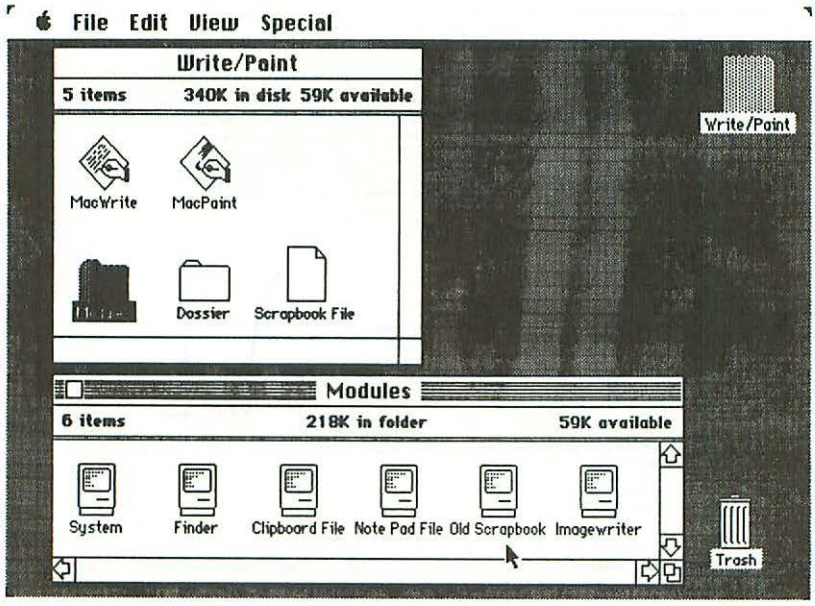
It may happen — depending upon the exact manner in which you terminate your use of the "Draw" disk — that you fall upon a dialog box (see following illustration) with an *Eject* button. Whichever way you terminate, the important thing is to get the "Draw" disk out of the machine, to make way for the "Write/Paint" disk.



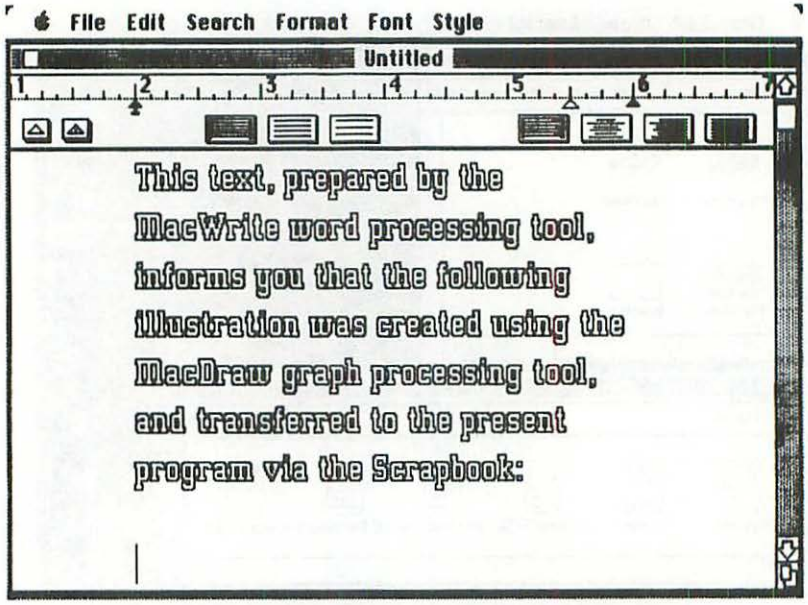
Once the "Draw" disk is ejected, put the "Write/Paint" disk in the drive. If you open the "Modules" folder (see following illustration), you can see that the file called "Old Scrapbook" is still there, but what we want to do now is to get MacWrite to use the Scrapbook borrowed from the other disk.



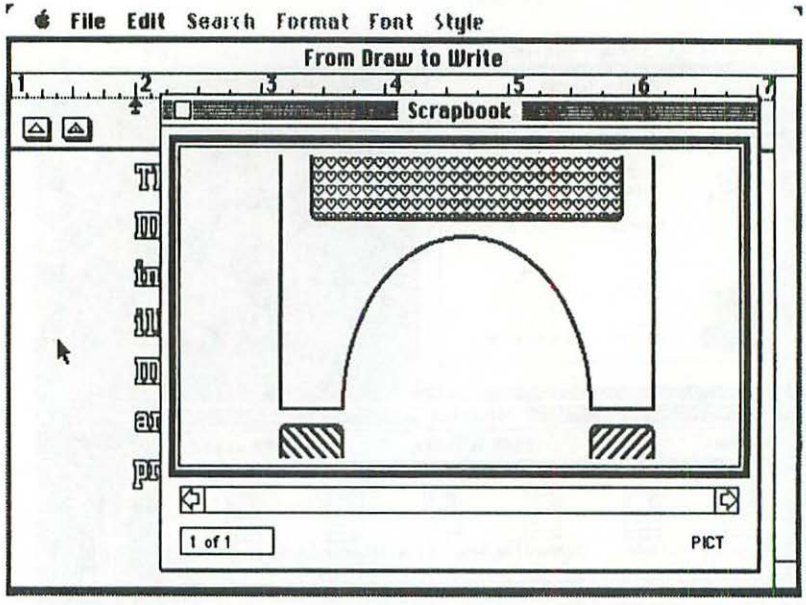
Drag the "Scrapbook File" icon onto the "Write/Paint" directory window (see following illustration). This triggers off a series of swapping operations involving the "Draw" and "Write/Paint" disks, and the final result is that the latter disk houses, at last, the "Draw" Scrapbook.



Use MacWrite to create a document such as we see in the following illustration:

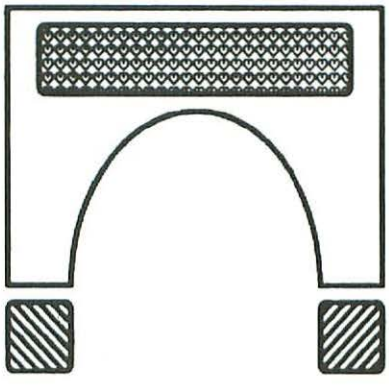


Next, open the Scrapbook in the usual manner, by pulling down the menu associated with the apple icon, as shown in the following illustration:



Once you use the Copy command, the arch is transferred onto the Clipboard. Then you can use the Paste command to insert the picture into the MacWrite document we just prepared. The result, when printed, is as follows:

This text, prepared by the
MacWrite word processing tool,
informs you that the following
illustration was created using the
MacDraw graph processing tool,
and transferred to the present
program via the Scrapbook:



Maybe that's as good a point as any to leave the subject of these graphics-oriented Macintosh products, and to get back to the sort of thing that we normally associate with computers: namely, *calculating*. . . .

chapter 8
"Multiplan" — Electronic Worksheet

The *Microsoft Multiplan* tool enables you to develop worksheets on the Macintosh.

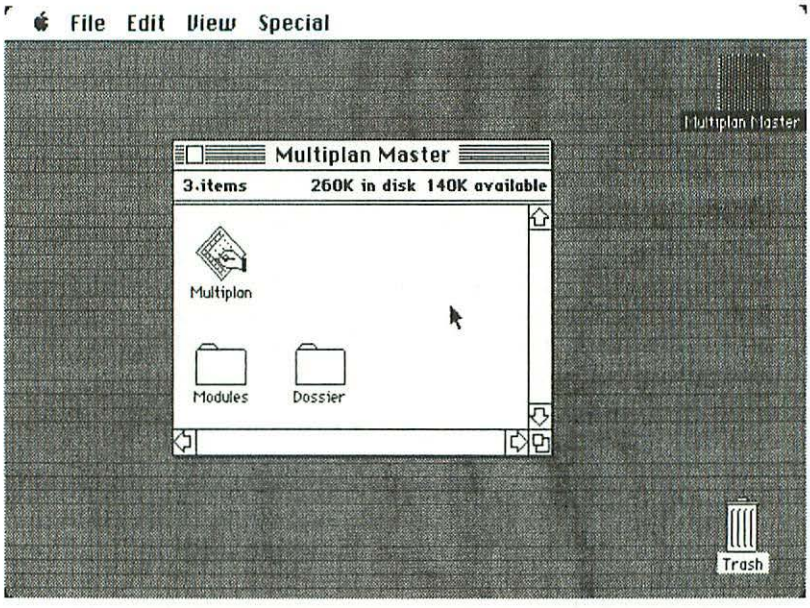
What exactly is a worksheet? Basically it's a rectangular grid of rows and columns that form *cells*. Inside the various cells you can record elements of information. Quite often, the element of information in a particular cell is in fact a *formula* explaining how the value of that cell must be calculated using the values in other cells.

Every time you change the data inside a single cell, Multiplan recalculates the value of every other cell on the worksheet.

Using Multiplan, you can carry out such tasks as budgeting, investment planning, sales forecasting, etc. In fact you can create a Multiplan model of any real-life situation that can be represented quantitatively by a rectangular grid of rows and columns. The horizontal axis of the grid frequently represents *time*, and the vertical axis then indicates the magnitude of some particular event (e.g., sales volumes) at specific moments in time. But Multiplan models could be created for countless different types of happenings. There's even a famous jazz musician who uses the electronic worksheet technique to schedule his future concerts.

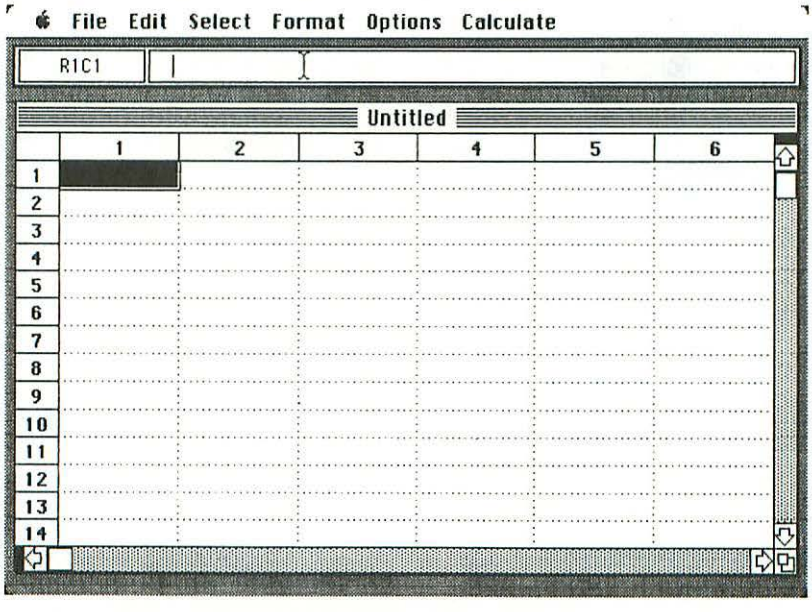
Manipulating the worksheet

The directory of the "Multiplan Master" disk is shown in the following illustration:



The icon named "Multiplan" is the program itself. "Modules", as usual, is a folder holding six elements of system software, and "Dossier" is an empty folder that we might use later on for filing away the worksheet that we are about to create.

As soon as we open "Multiplan", the screen appears as follows:



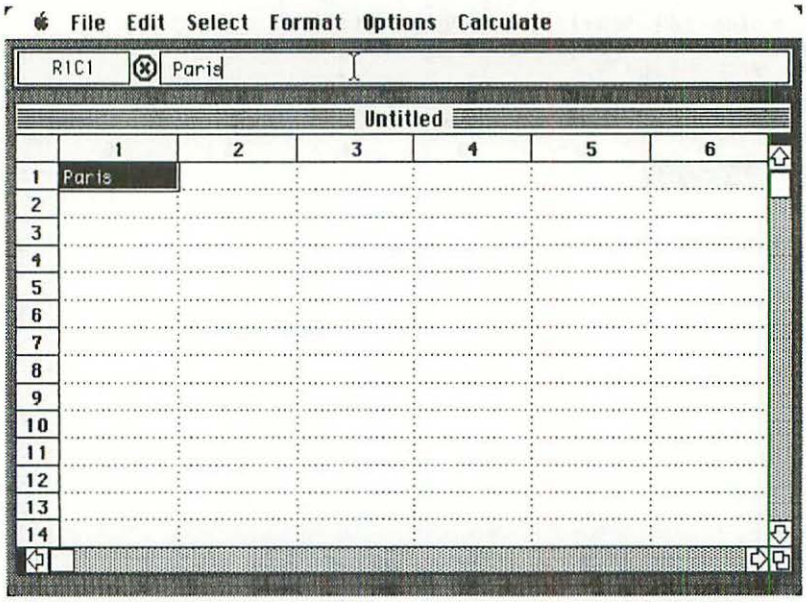
Notice, first of all, that there are six names in the menu bar. We'll look at the menus themselves later on.

Most of the screen is taken up by the *worksheet window*, labeled "Untitled". In fact, the six columns and fourteen rows that you see at present are merely the upper left corner of the worksheet, which can contain up to 255 rows and 63 columns. The scroll bars are used in a conventional manner to move to any part of the entire worksheet.

The black cell with the narrow white border is the *current cell*. At any particular moment, when creating or running a Multiplan model, the current cell is the one in which the user is currently working. In fact, this notion of a current cell is quite similar to the Macintosh concept of *selection* that we have encountered many times already. As you might have expected, selection of a new current cell is carried out simply by moving the pointer to the cell in question and clicking the mouse button.

Between the menu bar and the title bar of the main window, there's a so-called *formula bar*. To the left of this formula bar, the *cell register* displays at all times the coordinates of the current cell: here R1C1, for row 1 and column 1.

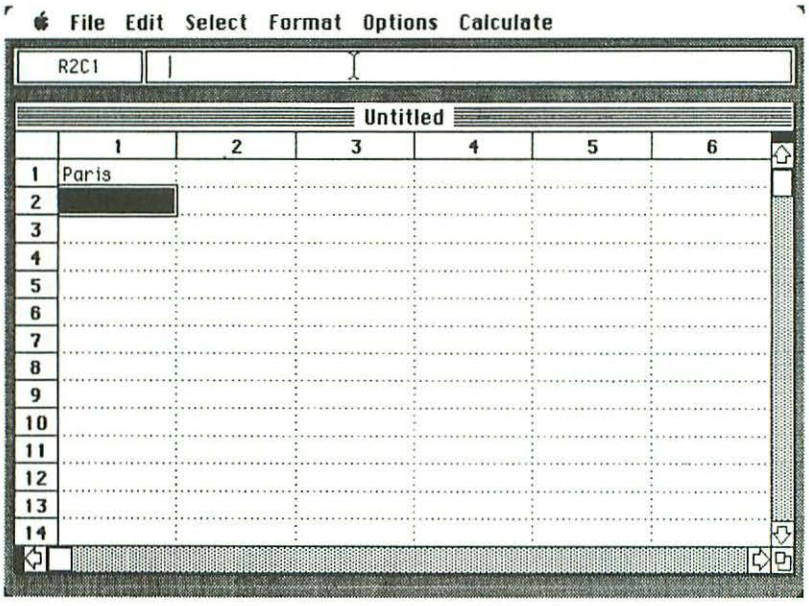
The main zone of the formula bar — with a blinking vertical bar to indicate the insertion point — is where you actually describe the manner in which the value of the current cell is to be derived. To enter information into this formula bar, you first have to click it, and then you can use the keyboard to specify the formula or the value for the current cell. For example, let's suppose that we enter the word "Paris" into the first cell on the worksheet. The following illustration indicates the appearance of the screen just before the <**Return**> key is hit:



Between the cell register and the word we have just typed, a so-called *cancel icon* has appeared, indicating that the contents of the current cell are to be canceled and replaced by what we are typing.

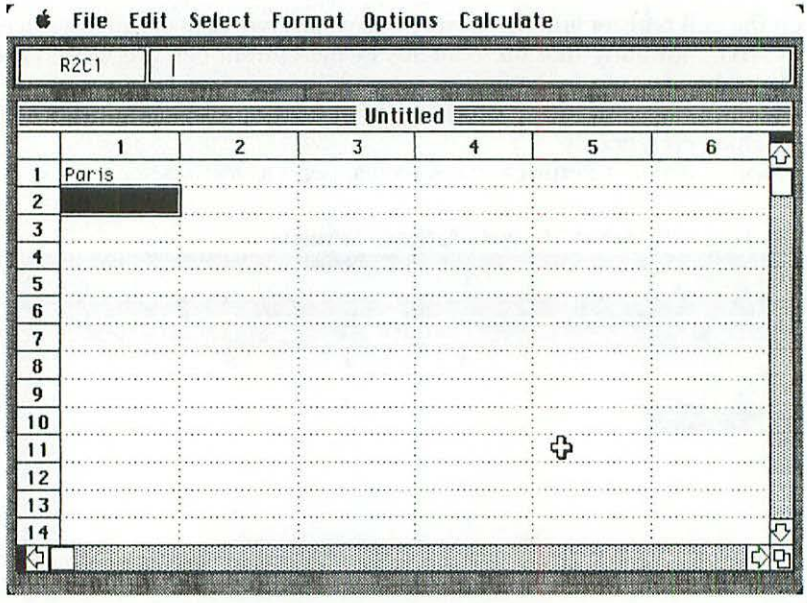
Notice that the word being typed in the formula bar appears simultaneously in the highlighted current cell.

As soon as we hit <**Return**>, the screen appears as follows:



Cell R1C1 has now received the value "Paris", and the next cell has been selected.

If we want to select a different cell, say R11C5, we move the pointer there (see following illustration) and click it. Notice that the pointer, as soon as you move it onto the worksheet window, takes the form of a cross.

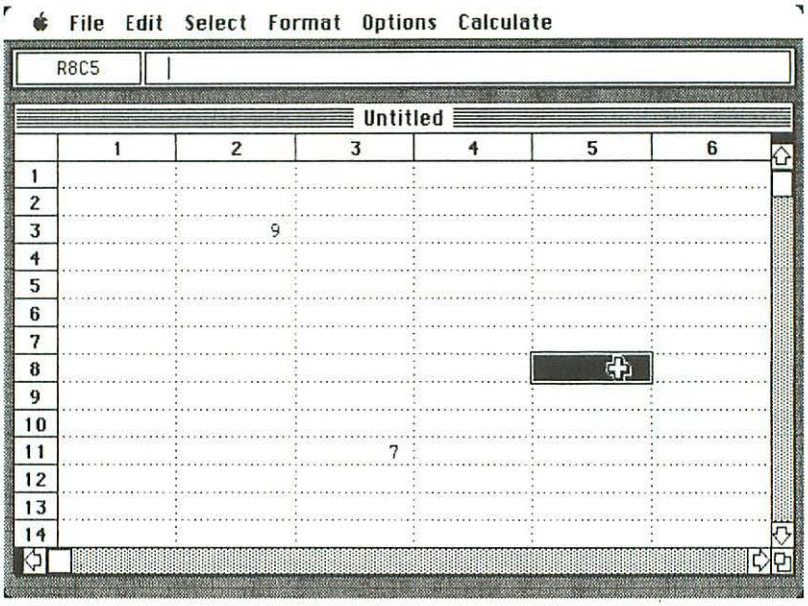


As far as pointers are concerned, Multiplan is a star performer, with no less than *five* different models, depending on the part of the screen where the pointer is located. In the menu bar and menus it's a conventional arrow; in the formula bar, an I-beam; in the worksheet window, a cross; in the scroll bars, a little hand with a pointing finger; and there's even a Celtic cross for the title bars of windows.

Calculations

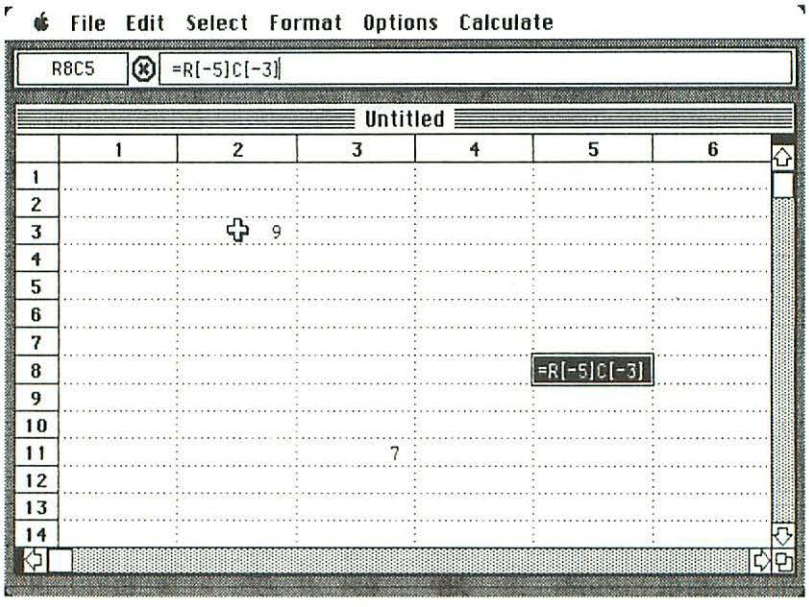
Let us examine the simplest calculation imaginable: the addition of two numbers.

In the following illustration, cells R3C2 and R11C3 have received respectively the values 9 and 7, and cell R8C5 is selected:



What we would like to do is to write a formula instructing Multiplan to put the result 9 + 7 in the current cell.

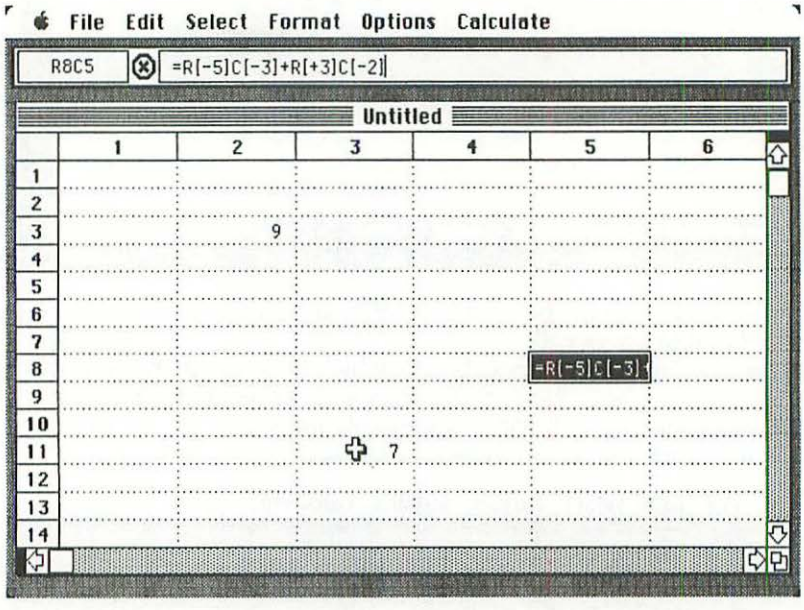
We start out by typing an equals sign, which informs the machine that we are about to enter a formula. Now, to let Macintosh know that the first term to be summed is the value in cell R3C2, we actually move the pointer to this cell and click it. At that point, the screen appears as follows:



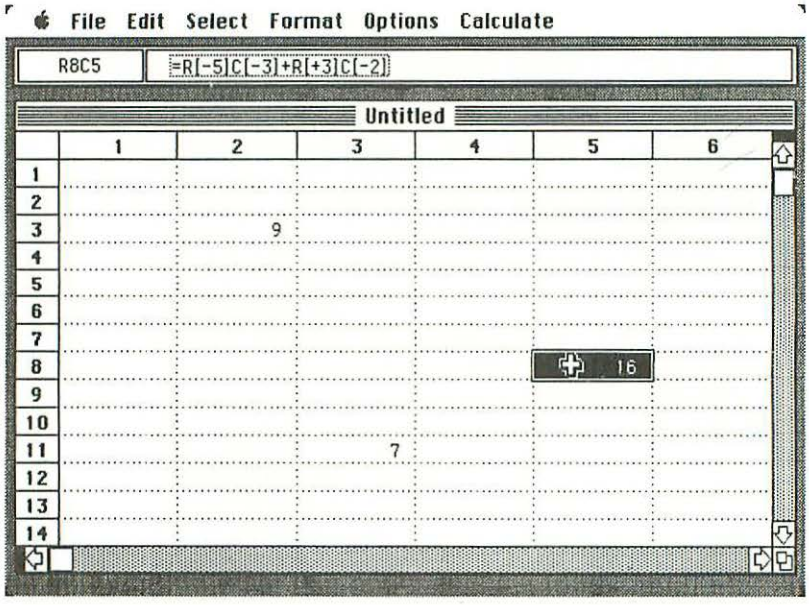
Instead of calling this cell R3C2, Multiplan refers to it as R[-5]C[-2]. This is the *relative address* of cell R3C2 with respect to the current cell, R8C5. The numbers in square brackets indicate how many rows and columns there are between the current cell and the cell we just clicked. The pair of negative numbers indicate that the clicked cell is 5 rows up from the current cell, and 3 columns to the left of it.

Next, we type a + sign on the keyboard, and we select the second term of the addition, cell R11C3, by moving the pointer there and clicking it. The relative address of this second cell is R[+3]C[-2]. That's to say, it's 3 rows down from the current cell, and 2 columns to the left of it.

The following illustration shows the final formula:

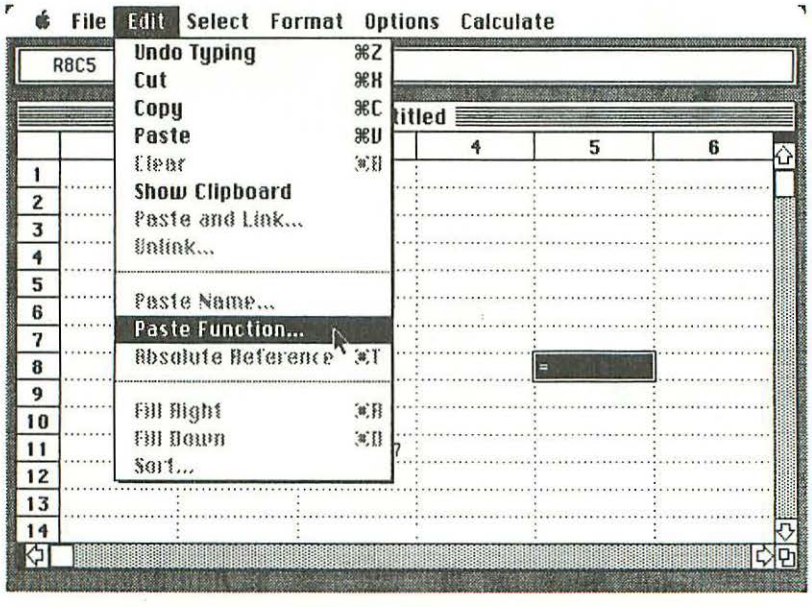


As soon as we hit <**Return**>, to indicate that the formula is complete, cell R8C5 receives the result 16, as shown in the following illustration:

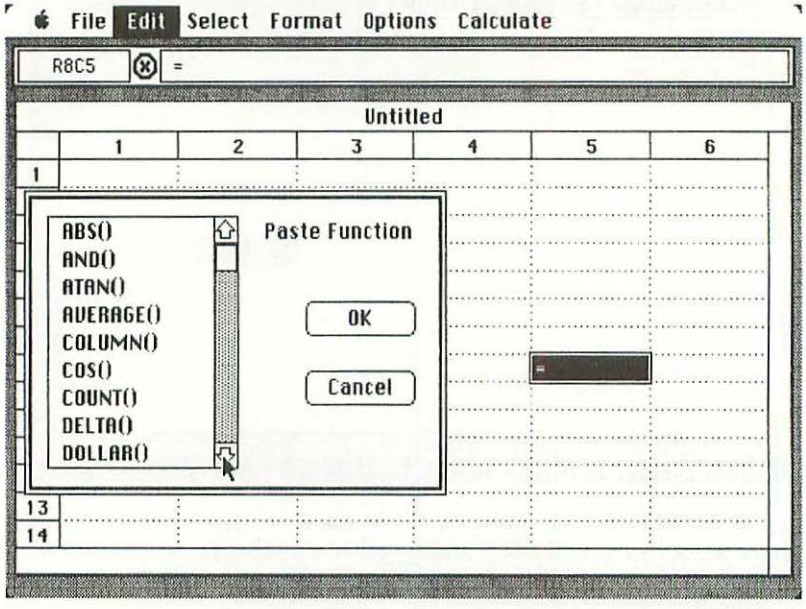


We could have obtained this same result in another manner. Let's return to the point where we selected cell R8C5 and typed an equals sign to indicate that we're about to enter a formula. This time we'll create the formula itself in a slightly different fashion.

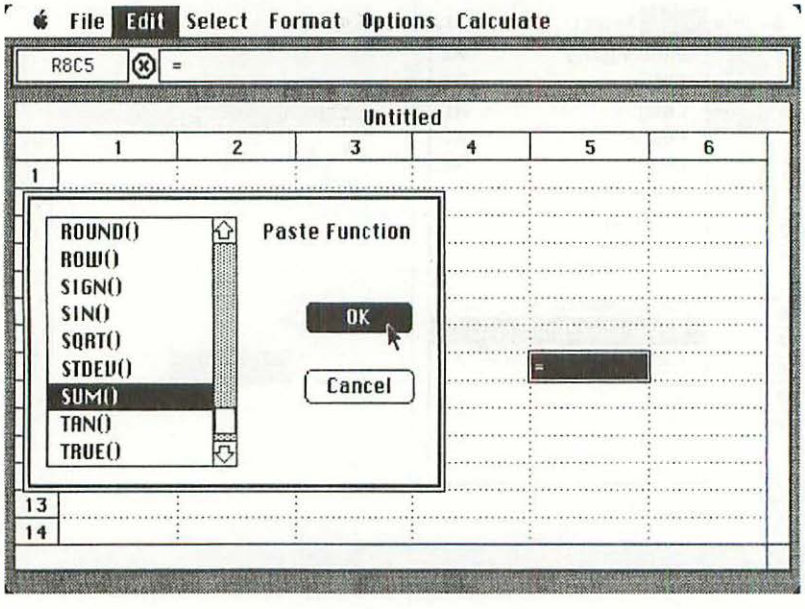
If you pull down the Edit menu (see following illustration), there's an interesting command called **Paste Function** that allows you to browse through a list of over forty Multiplan *functions*.



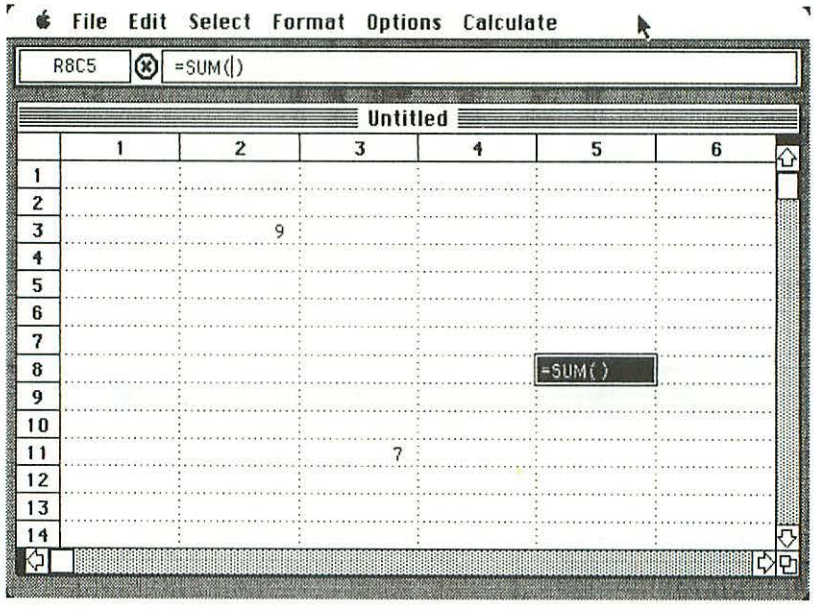
A dialog box (see following illustration) includes a mini-window — called a *list box* — with the names of nine functions in the list. You can use the vertical scroll bar to move through the entire list.



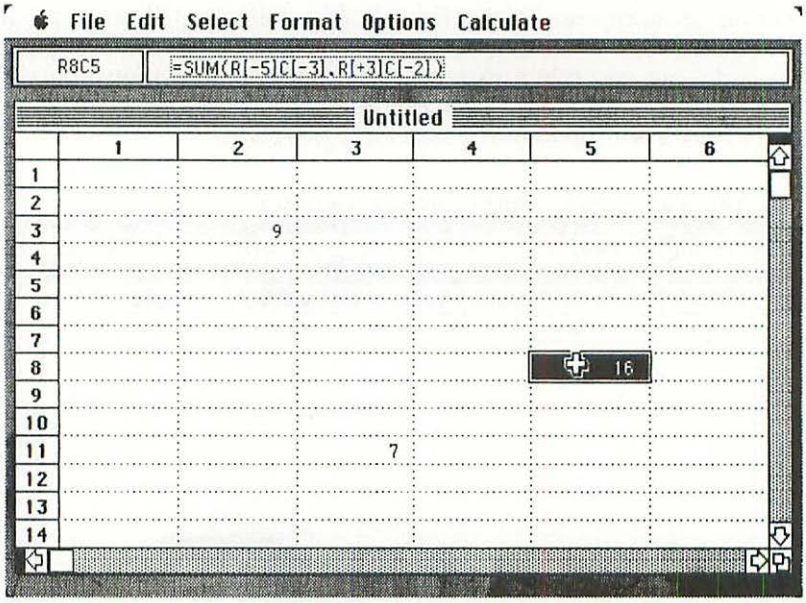
Let's scroll through to the SUM function (see following illustration). Having selected this function by clicking it, we only have to click the **OK** button in order to get the function pasted into our future formula.



The function, as it appears in the formula (see following illustration), has no argument. SUM is a function whose argument is a *list* of elements separated by commas, and — as you might expect — the purpose of this function is to add together all the elements in the list. In our example, we have only two elements to be summed, so it's now up to us to select these two cells.



We do this in the same way as before. First we move the pointer to cell R3C2 and click it. Its relative address appears in the formula bar as R[-5]C[-3]. We then type in a comma, to separate the two elements in the list. Next we move the pointer to cell R11C3 and click it. Its relative address is inserted in the formula as R[+3]C[- 2]. Then, to indicate that the formula is complete, we hit **<Return>**. The final result is shown in the following illustration:



In this very elementary example, the formula using the SUM function is almost identical to the previous formula. But Multiplan functions can be most handy in more complicated situations.

Here is a list of most of the functions available:

Mathematic functions

ABS(number)	Absolute value of a number
AVERAGE(list)	Average of the values in the list
COUNT(list)	Counts the number of items in the list
EXP(number)	Exponentiation
INT(number)	Integer portion of a number
LN(number)	Logarithm, natural
LOG10(number)	Logarithm, base 10
MAX(list)	Maximum value in list
MIN(list)	Minimum value in list
MOD(number to divide, number to divide by)	Remainder after division
NPV(rate, list)	Net present value of list at rate

ROUND(number, number of digits)	Rounds a number to number of digits
SIGN(number)	Sign of a number (1 positive, —1 negative)
SQRT(number)	Square root
STDEV(list)	Standard deviation
SUM(list)	Sum of numbers in list

Trigonometric functions

ATAN(number)	Arctangent
COS(number)	Cosine
SIN(number)	Sine
TAN(number)	Tangent

Logical functions

IF(logical-expression, value-if-true, value-if-false)	Chooses which value to give depending on the logical values of an expression
AND(list)	True if all values in list are true
NOT(logical-expression)	True if false, False if true
OR(list)	True if any value in list is true
TRUE()	True (logical value)
FALSE()	False (logical value)

Text functions

DOLLAR(number)	Dollar representation of a number
FIXED(number, decimals)	Fixed format of a number
LEN(text)	Length of text
MID(text, start-position, number-of-characters)	Extracts characters from a text value
REPT(text) number-of-times)	Repeats text number of times
VALUE(text)	Changes text value to number

Other functions

COLUMN()	Column number
ROW()	Row number
INDEX(area, subscripts)	Gets a value from area located by subscripts
LOOKUP(number, table)	Looks up a value in a table

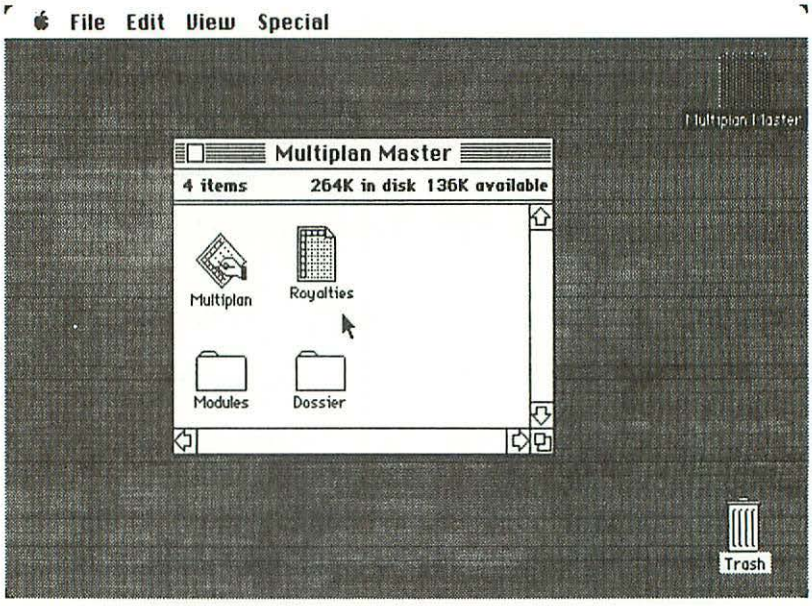
To use one of these functions in a formula, either you type the function name on the keyboard, or you use the Paste Function facility in the Edit menu.

Demonstration

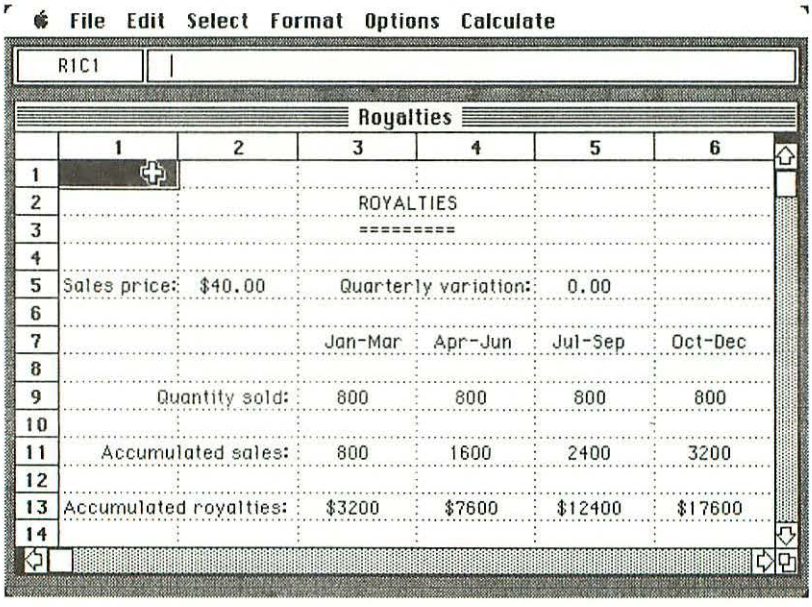
Let us now work through a complete example in Macintosh Multiplan, for that's the best way of seeing how this tool functions.

Imagine the case of a freelance programmer who has just created an entertainment product that is going to be marketed by a software company, and suppose that the programmer has decided to use a Multiplan model to figure out how much money she's likely to earn in the way of royalties. The company calculates royalties in the following manner: 10% of the sales price for the first thousand copies sold, 15% for the following two thousand copies, and 20% on all sales beyond these first three thousand copies. The sales price has not been fixed yet, and so the Multiplan model should allow us to propose various different values, so as to see the incidence of the sales price upon the royalties. The model — which deals with sales over a twelve-month period — is designed in such a way that we can propose the volume of sales for the first quarter, and then suppose that sales vary each quarter by such-and-such a fixed percentage (which may be either positive or negative).

Let's suppose that the heroine of our case study has devoted the necessary time and energy (about twenty minutes' work) to the construction of a Multiplan model called "Royalties", that is now stored away safely on the disk, as shown in the following illustration:



Double-click the "Royalties" icon to open the document. The main part of the worksheet then appears on the screen, as shown in the following illustration:

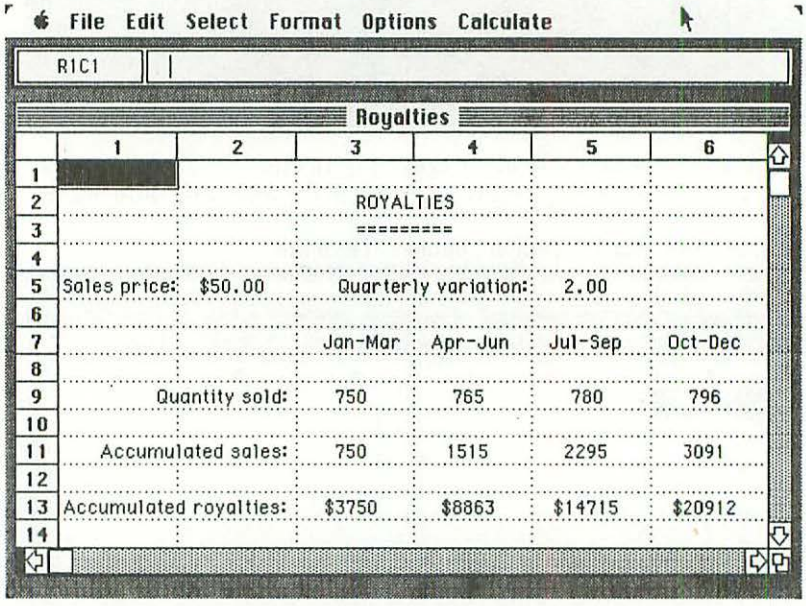


	1	2	3	4	5	6
1						
2			ROYALTIES			
3			=========			
4						
5	Sales price:	$40.00	Quarterly variation:		0.00	
6						
7			Jan-Mar	Apr-Jun	Jul-Sep	Oct-Dec
8						
9	Quantity sold:		800	800	800	800
10						
11	Accumulated sales:		800	1600	2400	3200
12						
13	Accumulated royalties:		$3200	$7600	$12400	$17600
14						

This section of the worksheet — which might be thought of as the visible part of the iceberg — is quite easy to understand. The programmer has imagined the case in which her product would be selling for $40, at a regular rate of 800 copies per trimester. (A quarterly variation of zero means that sales figures remain stationary all through the year.) During the first quarter, she sells less than a thousand copies, and so her royalties remain in the 10% bracket. In the second quarter, she breaks into the 15% bracket for 600 copies, and she remains at this rate of royalties for all sales in the third quarter. Finally, for 200 of the copies sold in the fourth quarter, she moves into the 20% bracket.

Worksheets are useful, above all, for answering questions of the "What if . . .?" variety. For example, our programmer might like to ask the following question: What would the royalties amount to if the product were to be sold for $50 instead of $40, if sales for the first trimester were 750, and if there was a 2% increase in sales every trimester?

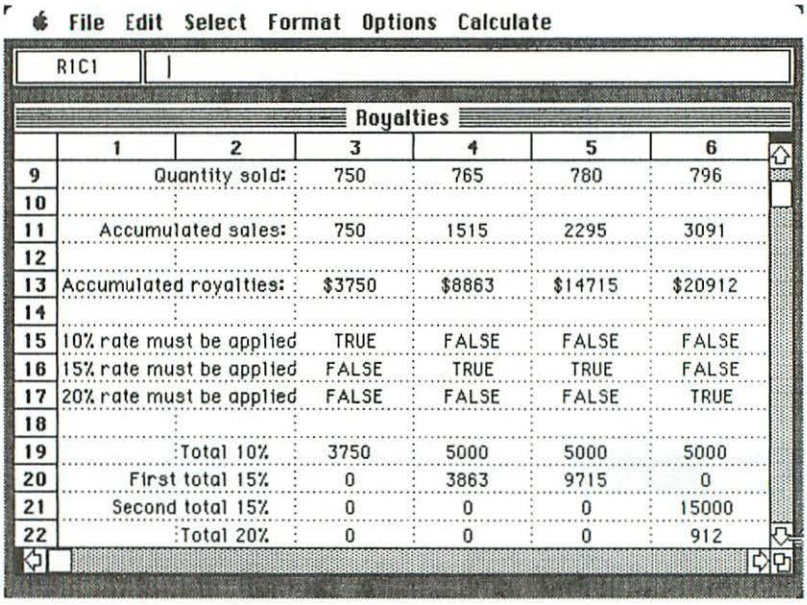
The answer is provided in the following illustration:



	1	2	3	4	5	6
1						
2			ROYALTIES			
3			=========			
4						
5	Sales price:	$50.00	Quarterly variation:		2.00	
6						
7			Jan-Mar	Apr-Jun	Jul-Sep	Oct-Dec
8						
9	Quantity sold:		750	765	780	796
10						
11	Accumulated sales:		750	1515	2295	3091
12						
13	Accumulated royalties:		$3750	$8863	$14715	$20912
14						

Now, Multiplan appears to be a very simple tool to use when you look at it from a superficial level, as an observer. But we must examine the actual "mechanisms" behind these results, for they are not necessarily child's play.

To discover this hidden part of the iceberg, let's scroll down the worksheet some eight rows, as shown in the following illustration:



	1	2	3	4	5	6
9		Quantity sold:	750	765	780	796
10						
11		Accumulated sales:	750	1515	2295	3091
12						
13		Accumulated royalties:	$3750	$8863	$14715	$20912
14						
15		10% rate must be applied	TRUE	FALSE	FALSE	FALSE
16		15% rate must be applied	FALSE	TRUE	TRUE	FALSE
17		20% rate must be applied	FALSE	FALSE	FALSE	TRUE
18						
19		Total 10%	3750	5000	5000	5000
20		First total 15%	0	3863	9715	0
21		Second total 15%	0	0	0	15000
22		Total 20%	0	0	0	912

The seven lines in the lower half of the screen reveal what is happening, behind the scenes, each time you run this Multiplan model.

For each trimester, there are three *logical flags* that indicate — by means of the values TRUE and FALSE — which of the rates must be applied in the calculation of royalties. One of these flags must always be TRUE, to indicate the end-of-trimester royalties rate, and the other two flags must be FALSE. We shall shortly see how these flags actually function.

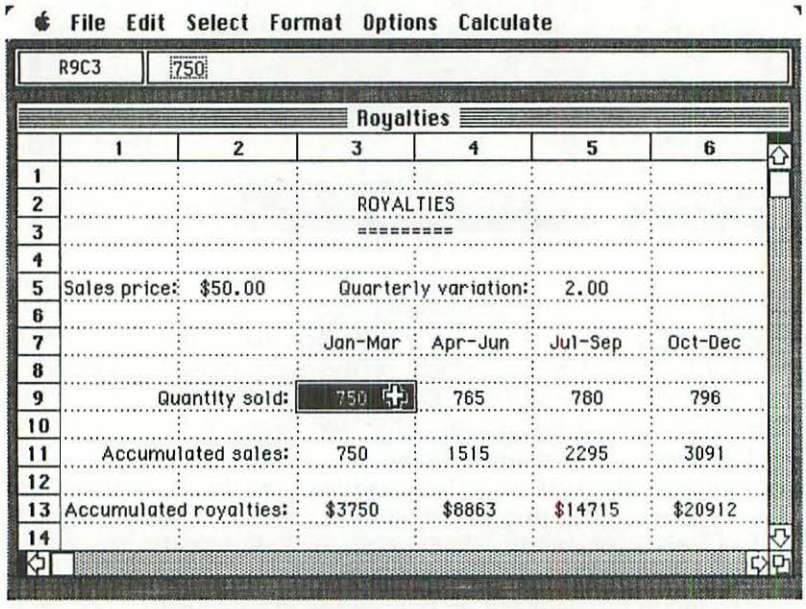
The four lines at the bottom of the screen indicate the accumulated royalties in each rate bracket. Notice that the royalties at 15% are broken down into two parts: the first total is updated so long as the end-of-trimester rate is still 15%, whereas the second total is used as soon as the 20% bracket has been attained. (This minor technical problem will become clearer in a moment or two, when you see the actual formulas for these various cells.)

Let's start at the beginning. There are three or four different kinds of cells on this worksheet. First of all, it's interesting to realize that this "Royalties" model uses only 59 of the 16065 cells available on the worksheet: that is, less than *half a percent* of the global capacity of the worksheet. In other words, most of the cells remain empty.

If we look at the 59 cells that do in fact contain something, this "something" is of three kinds: *legends*, such as "Sales price:" in cell R5C1; *numbers*, such as the sales price of fifty dollars in cell R5C2; or *formulas*.

As far as legends are concerned, we should point out that they only occupy the cell in which they start, even though they give the impression of flowing beyond the cell boundary onto the neighboring cell. For example, the legend "Accumulated royalties:" occupies only the cell R13C1, and the neighboring cell R13C2 remains unused and therefore completely empty. In other words, the cells themselves must be thought of as mini-windows that can hold more than meets the eye. We'll see, later on, that there's a command that allows you to actually widen any column on the worksheet.

Let us turn now to the three cells on the "Royalties" worksheet that contain simple numbers. In the following illustration, the pointer has been positioned at cell R9C3:



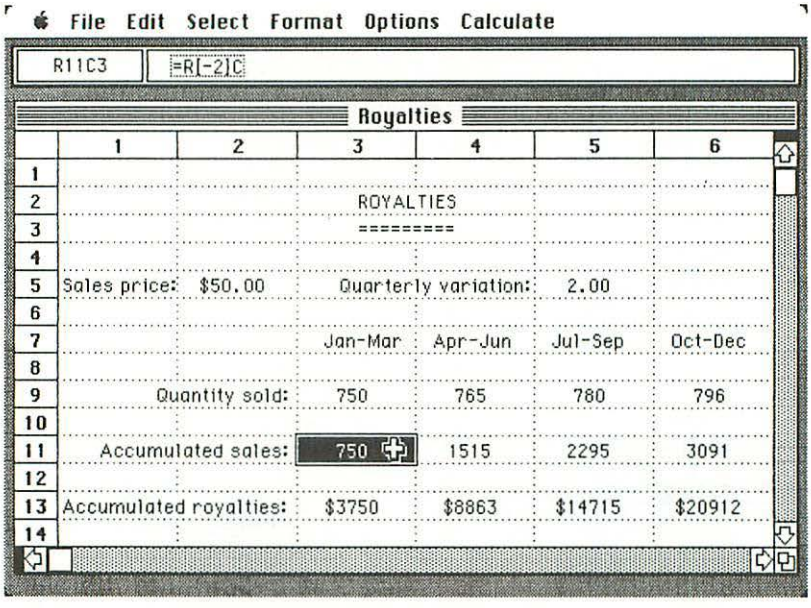
	1	2	3	4	5	6
1						
2			ROYALTIES			
3			=========			
4						
5	Sales price:	$50.00	Quarterly variation:		2.00	
6						
7			Jan-Mar	Apr-Jun	Jul-Sep	Oct-Dec
8						
9		Quantity sold:	750	765	780	796
10						
11		Accumulated sales:	750	1515	2295	3091
12						
13	Accumulated royalties:		$3750	$8863	$14715	$20912
14						

In the formula bar, we discover the value 750, which is simply the current contents of the cell. If you want to change this value, all you have to do is to type the new value on the keyboard and hit <**Return**>, then Multiplan would recalculate the entire model to take account of this modification.

There are two other cells on the worksheet that can be changed in the same way: R5C2, which contains the sales price in dollars, and R5C5, which contains the quarterly variation expressed as a percentage.

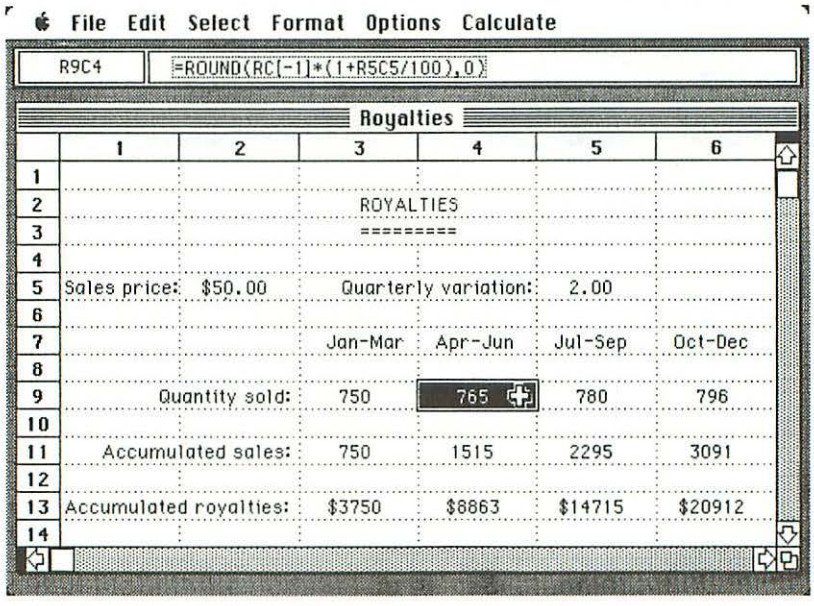
Besides the cells containing legends, and these three cells containing straightforward numbers, all the other non-empty cells on the worksheet contain formulas.

Let us start with the simplest formula of all, as shown in the following illustration:



Up in the formula bar, the relative address R[-2]C indicates that the value of cell R11C3 (the accumulated sales at the end of the first trimester) is identical to the value located two rows further up in the same column. Obviously, for this *first* trimester, the "accumulated" quantity is identical to the quantity sold during the trimester.

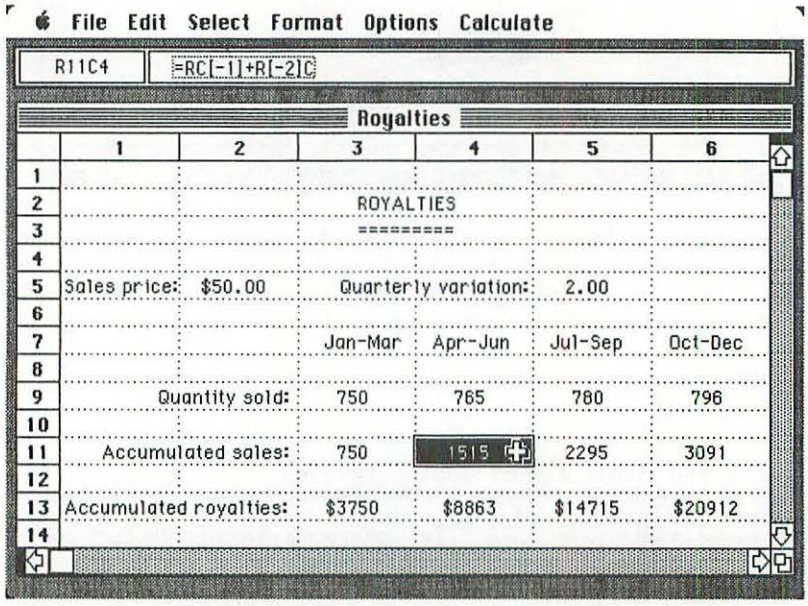
Cell R9C4 provides us with a more interesting formula, as shown in the following illustration:



The ROUND function — which was no doubt pasted into this formula just like SUM, in our earlier examples — gets rid of unwanted fractions. The formula states that sales for the second trimester must be calculated by multiplying (asterisk symbol) the number in the previous column by the quarterly variation, cell R5C5, viewed as a percentage.

Why is cell R5C5 referred to in the formula, exceptionally, by its *absolute address*, instead of a relative address? The reason is that this formula for R9C4 can now be copied, as such, for the two remaining cells in the row: R9C5 and R9C6 (sales for the third and fourth trimesters, respectively). If the formula for cell R9C4 had used the relative address of cell R5C5, then it would not have been possible to simply copy the same formula for cells R9C5 and R9C6.

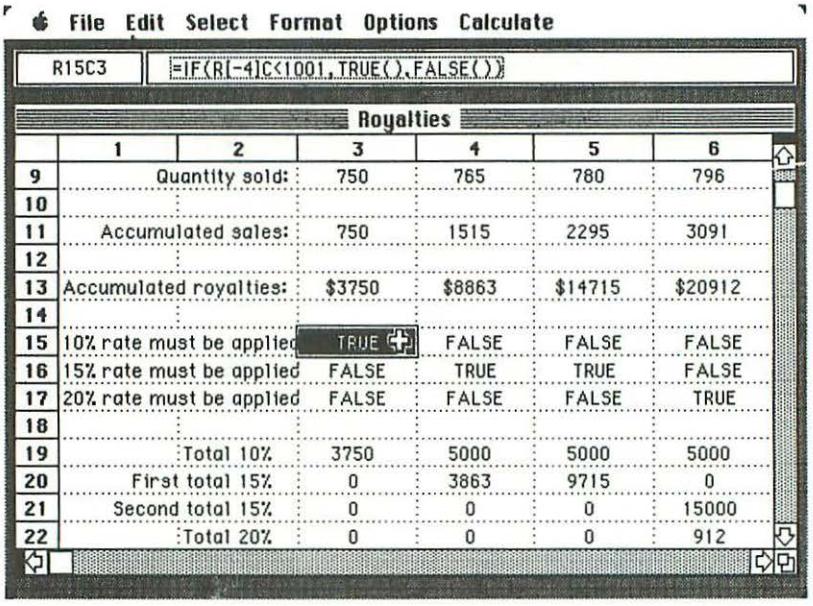
Now let's look at the formula for cell R11C4, as shown in the following illustration:



	1	2	3	4	5	6
1						
2			ROYALTIES			
3			=========			
4						
5	Sales price:	$50.00	Quarterly variation:		2.00	
6						
7			Jan-Mar	Apr-Jun	Jul-Sep	Oct-Dec
8						
9		Quantity sold:	750	765	780	796
10						
11		Accumulated sales:	750	1515	2295	3091
12						
13	Accumulated royalties:		$3750	$8863	$14715	$20912
14						

To calculate the accumulated sales for the second trimester, Multiplan is instructed to add the quantity sold during the current trimester (the number two rows up in the same column) to the accumulated sales at the end of the previous trimester (the number one column back in the same row). This same formula can be copied, identically, for the cells R11C5 and R11C6.

Let's move down to the lower half of the model in order to examine several quite meaty formulas that use the so-called *logical functions* such as IF, TRUE, FALSE, AND and NOT. Here's the formula for cell R15C3:



	1	2	3	4	5	6
9		Quantity sold:	750	765	780	796
10						
11		Accumulated sales:	750	1515	2295	3091
12						
13		Accumulated royalties:	$3750	$8863	$14715	$20912
14						
15		10% rate must be applied	TRUE	FALSE	FALSE	FALSE
16		15% rate must be applied	FALSE	TRUE	TRUE	FALSE
17		20% rate must be applied	FALSE	FALSE	FALSE	TRUE
18						
19		Total 10%	3750	5000	5000	5000
20		First total 15%	0	3863	9715	0
21		Second total 15%	0	0	0	15000
22		Total 20%	0	0	0	912

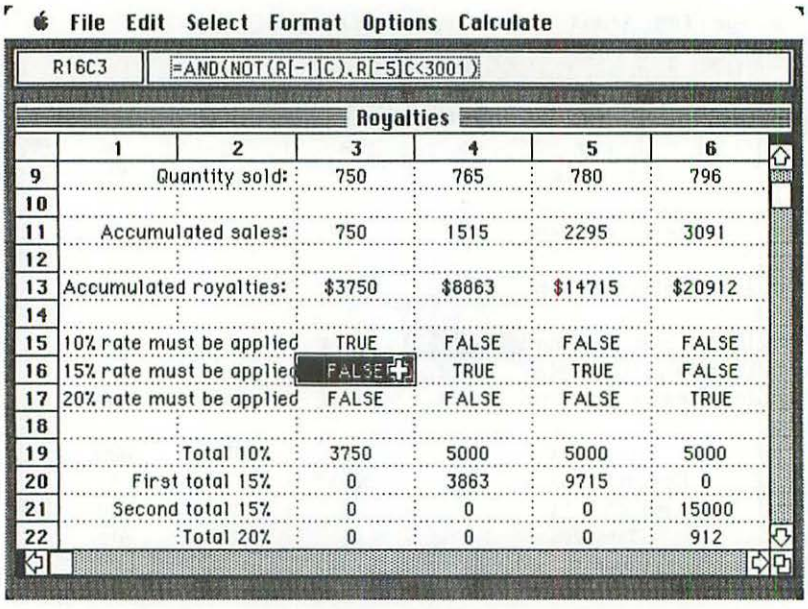
The argument of the IF function is a list of three elements, separated by commas. If the first element — a *logical expression* — turns out to be true, then the second element supplies the *value* of the function. But, if the logical expression turns out to be false, then the value of the function is supplied by the third element. As you have no doubt gathered, a so-called "logical expression" is something that must be either true or false.

Here, the logical expression is a comparison between the value that is four rows up in the same column (the accumulated sales at the end of the current trimester), and the number 1001. More precisely, Multiplan is told that, if it's true that the accumulated sales are less than 1001, then the value TRUE (second element in the list) should be inscribed in cell R15C3. On the other hand, if it's *not* true that the accumulated sales are less than 1001, then the value FALSE (third element in the list) should be inscribed in cell R15C3. Incidentally, don't worry about those curious pairs of empty round brackets after the function names TRUE and FALSE; that's merely a syntactic idiosyncrasy of no importance.

In other words, this cell must read TRUE if accumulated sales are below the thousand mark, or FALSE otherwise. That's what we mean when we speak — as we did, earlier on — of a "logical flag".

Obviously, this same formula can be copied, identically, in the three other cells in the row: R15C4, R15C5 and R15C6.

Let's move down to cell R16C3, as shown in the following illustration:



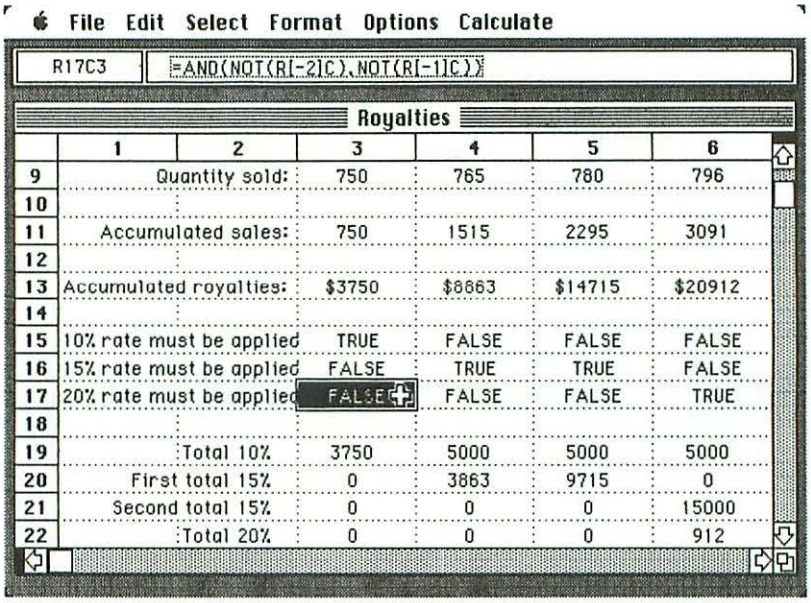
	1	2	3	4	5	6
9		Quantity sold:	750	765	780	796
10						
11		Accumulated sales:	750	1515	2295	3091
12						
13	Accumulated royalties:		$3750	$8863	$14715	$20912
14						
15	10% rate must be applied		TRUE	FALSE	FALSE	FALSE
16	15% rate must be applie		FALSE	TRUE	TRUE	FALSE
17	20% rate must be applied		FALSE	FALSE	FALSE	TRUE
18						
19		Total 10%	3750	5000	5000	5000
20		First total 15%	0	3863	9715	0
21		Second total 15%	0	0	0	15000
22		Total 20%	0	0	0	912

The argument of the AND function is a list of logical expressions. If *all* of them turn out to be true, then the function returns the value TRUE, otherwise it returns the value FALSE.

If cell R16C3 is to be TRUE, then two things are necessary: the cell one row up in the same column must NOT be true, AND the value of the cell five rows up in the same column (accumulated sales) must be less than 3001. In other words, this logical flag is set to TRUE as long as more than a thousand, but not more than three thousand, copies have been sold; otherwise it's set to FALSE.

This same formula can be copied for the other cells in the row: R16C4, R16C5 and R16C6.

Let's move down to the next cell, as shown in the following illustration:

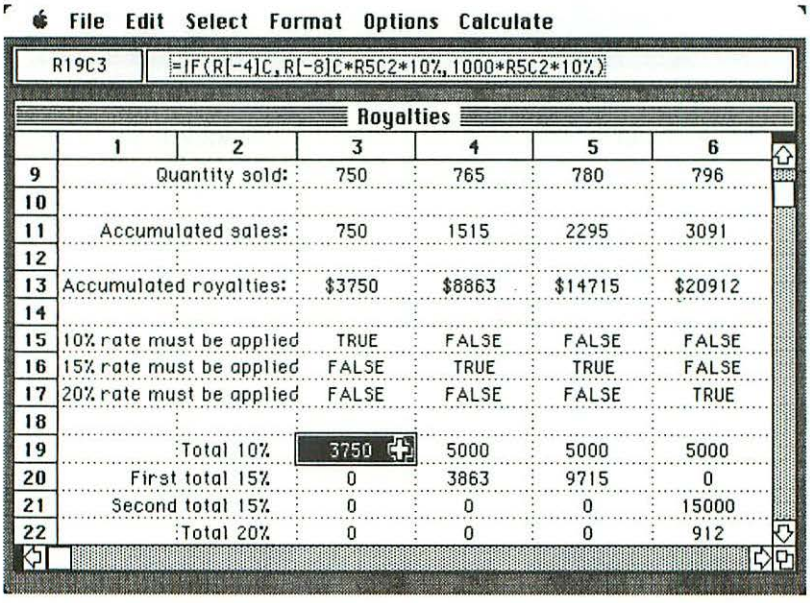


	1	2	3	4	5	6
9		Quantity sold:	750	765	780	796
10						
11		Accumulated sales:	750	1515	2295	3091
12						
13		Accumulated royalties:	$3750	$8863	$14715	$20912
14						
15		10% rate must be applied	TRUE	FALSE	FALSE	FALSE
16		15% rate must be applied	FALSE	TRUE	TRUE	FALSE
17		20% rate must be applied	FALSE	FALSE	FALSE	TRUE
18						
19		Total 10%	3750	5000	5000	5000
20		First total 15%	0	3863	9715	0
21		Second total 15%	0	0	0	15000
22		Total 20%	0	0	0	912

No problems at this level: cell R17C3 will be TRUE if both cells above it are FALSE.

The same formula can be copied into the adjacent cells: R17C4, R17C5 and R17C6.

Now, let's turn to the formulas, down the bottom of the model, that actually calculate the royalties in each bracket. The following illustration deals with the 10% bracket:

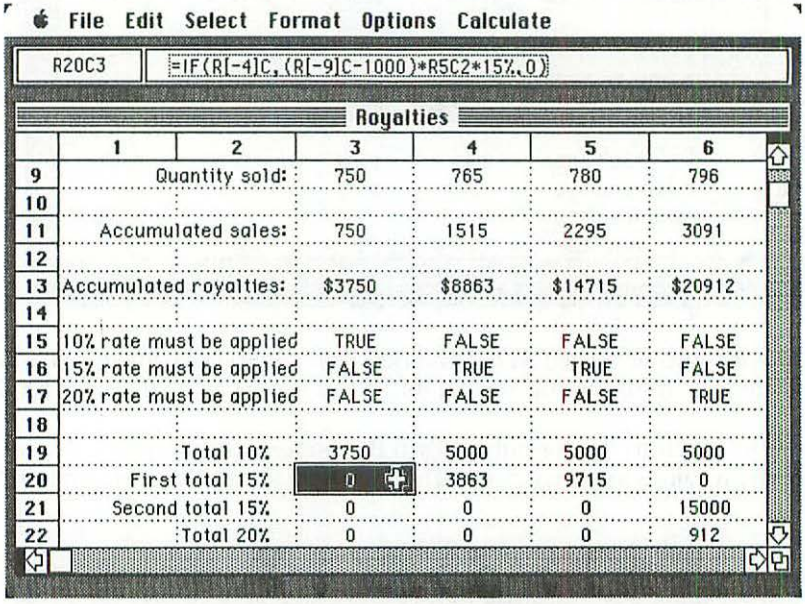


	1	2	3	4	5	6
9		Quantity sold:	750	765	780	796
10						
11		Accumulated sales:	750	1515	2295	3091
12						
13		Accumulated royalties:	$3750	$8863	$14715	$20912
14						
15		10% rate must be applied	TRUE	FALSE	FALSE	FALSE
16		15% rate must be applied	FALSE	TRUE	TRUE	FALSE
17		20% rate must be applied	FALSE	FALSE	FALSE	TRUE
18						
19		Total 10%	3750	5000	5000	5000
20		First total 15%	0	3863	9715	0
21		Second total 15%	0	0	0	15000
22		Total 20%	0	0	0	912

The outcome of this rather complicated formula for cell R19C3 depends upon the 10% logical flag just four rows up, in cell R15C3. If the flag is TRUE, then the function returns a value that is 10% of the product of the sales price in R5C2 and the accumulated sales, eight rows up in the same column. But, if the flag is FALSE, then the function returns a fixed amount that is 10% of the revenue obtained from the first thousand sales.

The same formula applies to the other three cells in the row: R19C4, R19C5 and R19C6.

Let's carry on moving down the column:

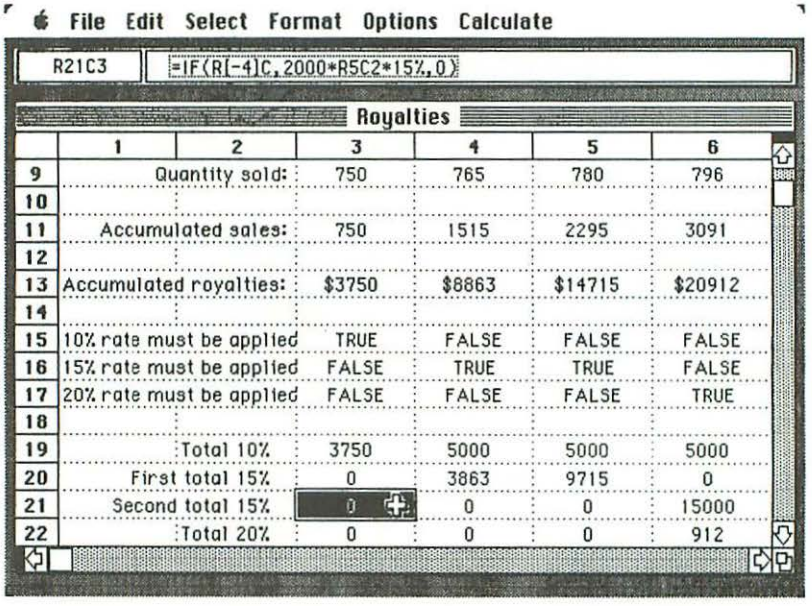


	1	2	3	4	5	6
9		Quantity sold:	750	765	780	796
10						
11		Accumulated sales:	750	1515	2295	3091
12						
13		Accumulated royalties:	$3750	$8863	$14715	$20912
14						
15		10% rate must be applied	TRUE	FALSE	FALSE	FALSE
16		15% rate must be applied	FALSE	TRUE	TRUE	FALSE
17		20% rate must be applied	FALSE	FALSE	FALSE	TRUE
18						
19		Total 10%	3750	5000	5000	5000
20		First total 15%	0	3863	9715	0
21		Second total 15%	0	0	0	15000
22		Total 20%	0	0	0	912

Cell R20C3 will contain a non-zero value *only if* the 15% logical flag, four rows up in the same column, is set to TRUE. In that case, cell R20C3 will contain an amount that is 15% of the revenue obtained from all sales beyond the first thousand.

The same formula can be copied into the neighboring cells: R20C4, R20C5 and R20C6.

Let's move on to the second-last cell in the column:

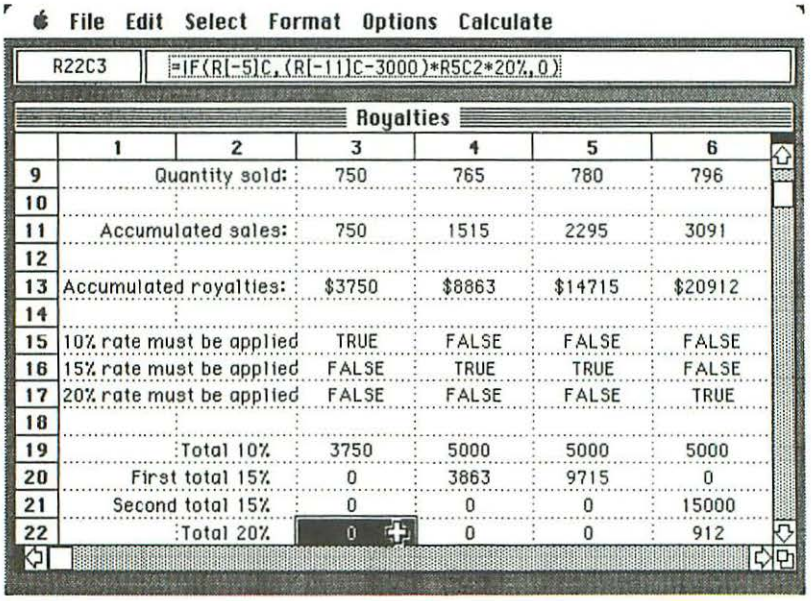


	1	2	3	4	5	6
9	Quantity sold:		750	765	780	796
10						
11	Accumulated sales:		750	1515	2295	3091
12						
13	Accumulated royalties:		$3750	$8863	$14715	$20912
14						
15	10% rate must be applied		TRUE	FALSE	FALSE	FALSE
16	15% rate must be applied		FALSE	TRUE	TRUE	FALSE
17	20% rate must be applied		FALSE	FALSE	FALSE	TRUE
18						
19		Total 10%	3750	5000	5000	5000
20		First total 15%	0	3863	9715	0
21		Second total 15%	0	0	0	15000
22		Total 20%	0	0	0	912

Cell R21C3 will contain a non-zero value *only if* the 20% logical flag, four rows up in the same column, is set to TRUE. In that case, cell R21C3 will contain a fixed amount that is 15% of the revenue obtained from the sale of two thousand copies.

The same formula can be copied into cells R21C4, R21C5 and R21C6.

We finally reach the last cell in the column, as shown in the following illustration:

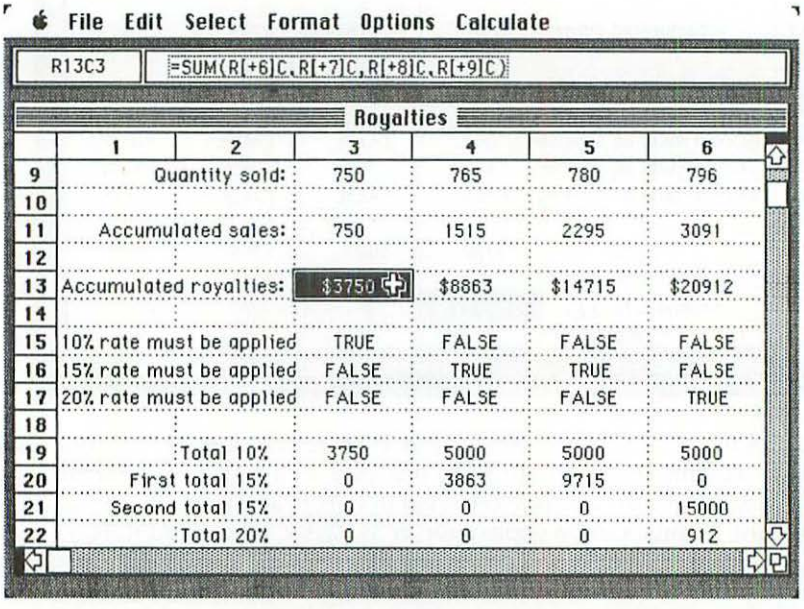


	1	2	3	4	5	6
9	Quantity sold:		750	765	780	796
10						
11	Accumulated sales:		750	1515	2295	3091
12						
13	Accumulated royalties:		$3750	$8863	$14715	$20912
14						
15	10% rate must be applied		TRUE	FALSE	FALSE	FALSE
16	15% rate must be applied		FALSE	TRUE	TRUE	FALSE
17	20% rate must be applied		FALSE	FALSE	FALSE	TRUE
18						
19		Total 10%	3750	5000	5000	5000
20		First total 15%	0	3863	9715	0
21		Second total 15%	0	0	0	15000
22		Total 20%	0	0	0	912

The value of cell R22C3 depends upon the 20% logical flag, five rows up in the same column. If the flag is true, then cell R22C3 receives an amount that represents 20% of revenue obtained from all sales beyond the first three thousand.

The same formula applies to cells R22C4, R22C5 and R22C6.

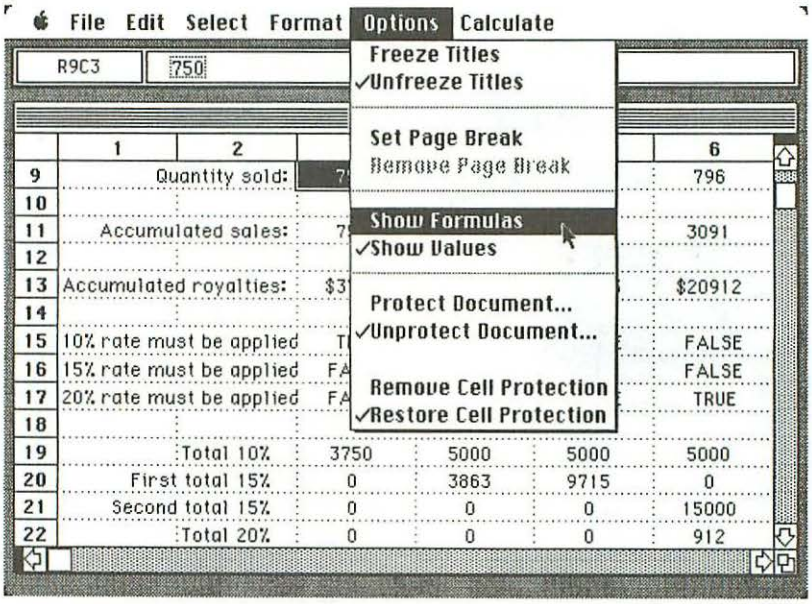
Let's move back up to the row that displays the accumulated royalties, as shown in the following illustration:



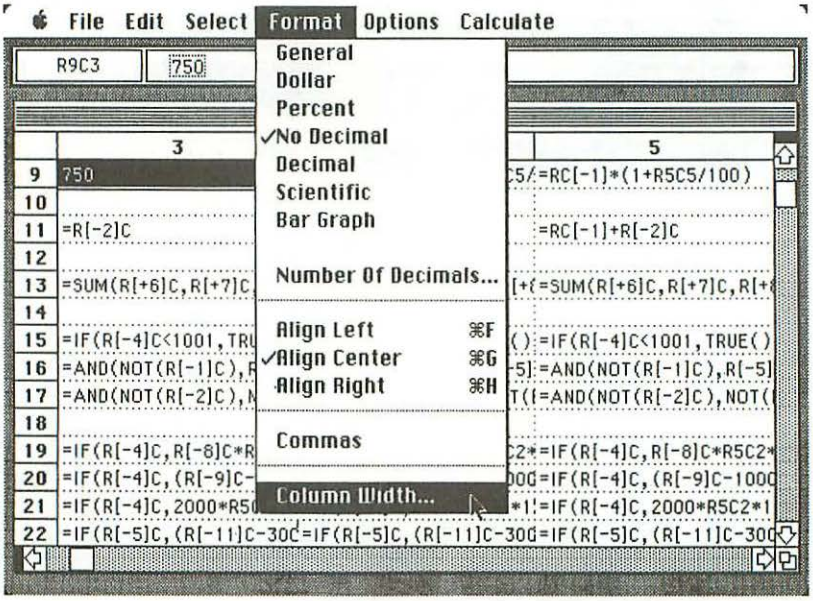
	1	2	3	4	5	6
9		Quantity sold:	750	765	780	796
10						
11		Accumulated sales:	750	1515	2295	3091
12						
13		Accumulated royalties:	$3750	$8863	$14715	$20912
14						
15	10% rate must be applied		TRUE	FALSE	FALSE	FALSE
16	15% rate must be applied		FALSE	TRUE	TRUE	FALSE
17	20% rate must be applied		FALSE	FALSE	FALSE	TRUE
18						
19		Total 10%	3750	5000	5000	5000
20		First total 15%	0	3863	9715	0
21		Second total 15%	0	0	0	15000
22		Total 20%	0	0	0	912

The value of cell R13C3 is simply the sum of the four values at the bottom of the model. This same formula is used for each of the remaining trimesters: cells R13C4, R13C5 and R13C6.

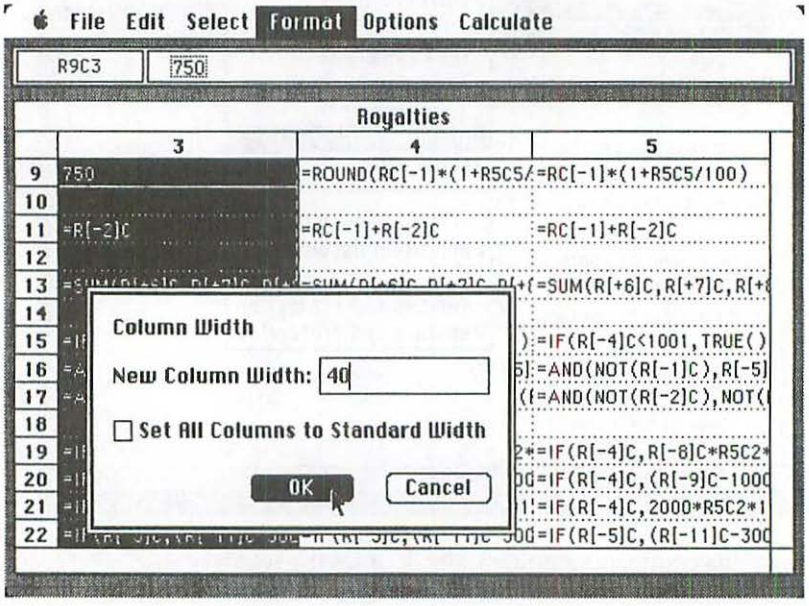
That's all there is in this Multiplan model. If you want to actually see all these formulas simultaneously, then you can pull down the **Options** menu (see following illustration) and choose the **Show Formulas** command.



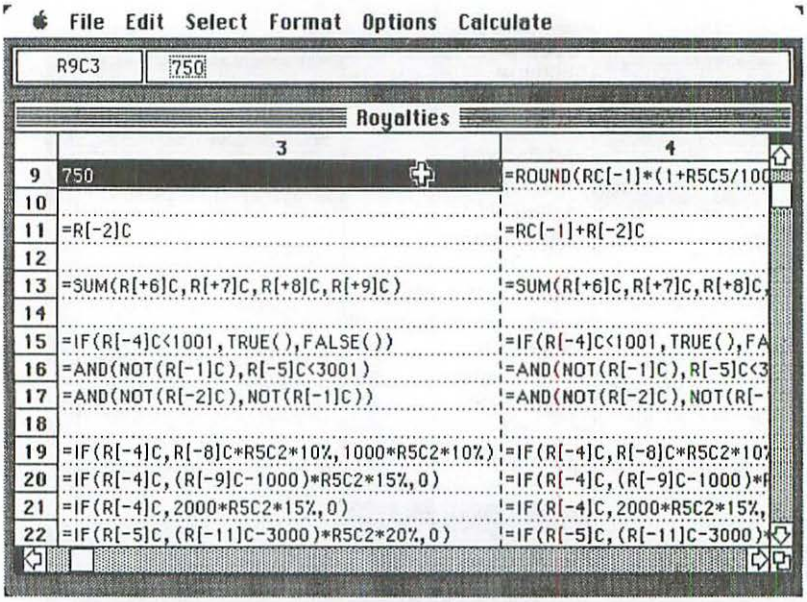
Even though this command automatically doubles the width of displayed columns, many of the formulas are still too long to fit into the allotted space. So it's best that you pull down the **Format** menu and ask for the **Column Width** command, as shown in the following illustration:



A dialog box asks you for a new setting for the column width (see following illustration). Reply by 40, and click the **OK** button to carry on.



Here's the result for column 3:



That brings us to the end of this rapid overview of Microsoft Multiplan. It's an elegant program that provides a superb example of what might be termed the "Macintosh style" of computing. Above all, it's a powerful productivity tool for people who are concerned by "What if . . .?" questions.

And what if we took a look now at a sister product of Microsoft Multiplan?

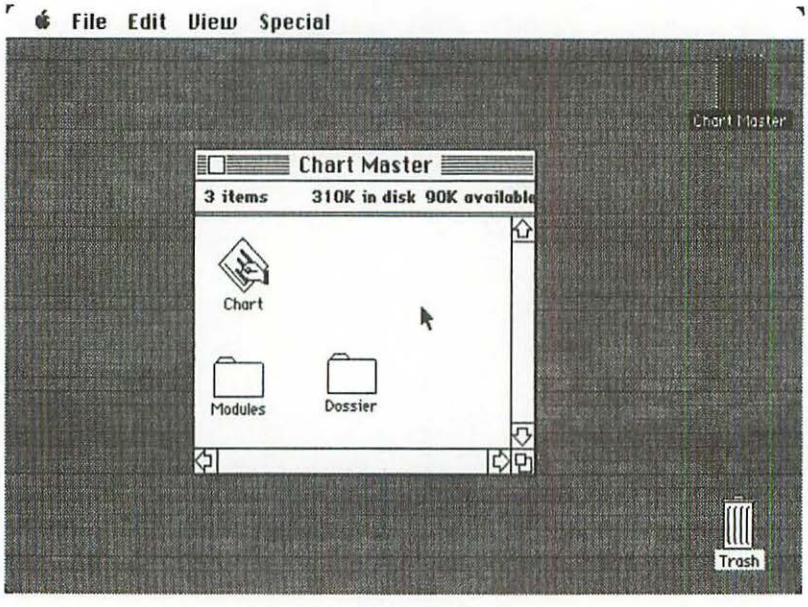
chapter 9
"Chart" — Business Graphics

The *Microsoft Chart* tool enables you to produce a great variety of figures of the kind that are usually referred to as *business graphics*: that is, charts that reflect the relationship between a pair of variables, one of which is often time.

The most familiar representation of such a chart is simply a rectangular graph. In Microsoft Chart terminology, the variable represented by the horizontal x-axis designates the so-called *categories* of the graph, whereas the y-axis is said to provide the *values*.

Chart offers you the choice between over 40 different formats, including various *histograms*, *line graphs* and *pie charts* . . . and nothing prevents you from changing rapidly from one format to another, with a view to selecting the most attractive presentation.

The directory of the "Chart Master" disk is shown in the following illustration:

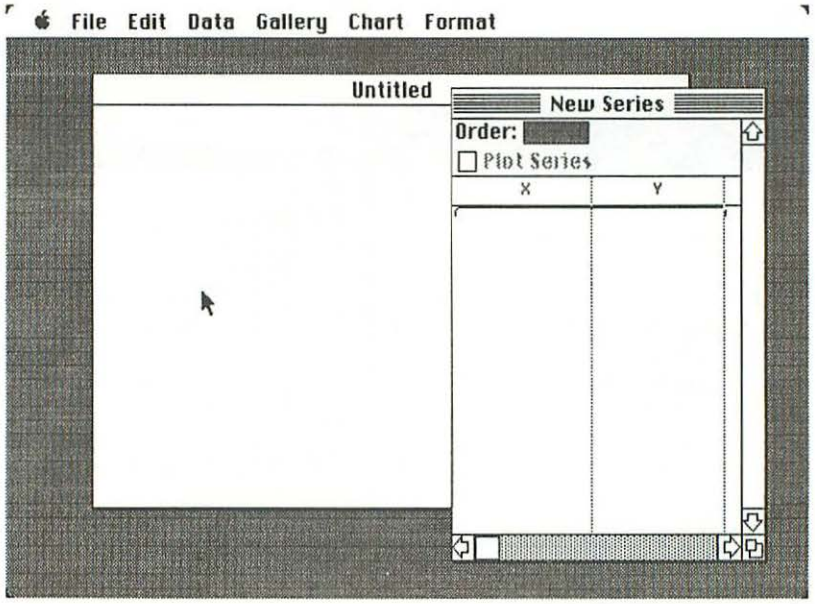


The icon named "Chart" is the program itself. "Modules", as usual, is a folder holding half a dozen elements of system software, and "Dossier" is an empty folder that might be used later on for storing some of the graphic documents that we are about to create.

Demonstration

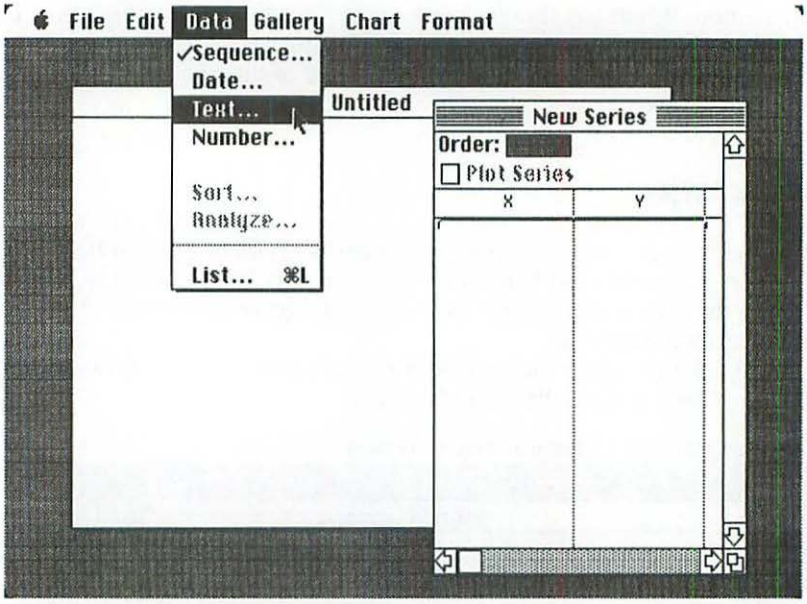
Let us imagine an elementary example that consists of plotting two series of figures for a business: *forecast* monthly sales for the year, and *actual* monthly sales. What we intend to do is to present these two series of figures as a single chart, so that they can be easily compared.

Double-click the "Chart" icon to start the program. Two windows appear on the screen, as shown in the following illustration:



In the background, the untitled *chart window* will not be used until later on, to display actual charts.

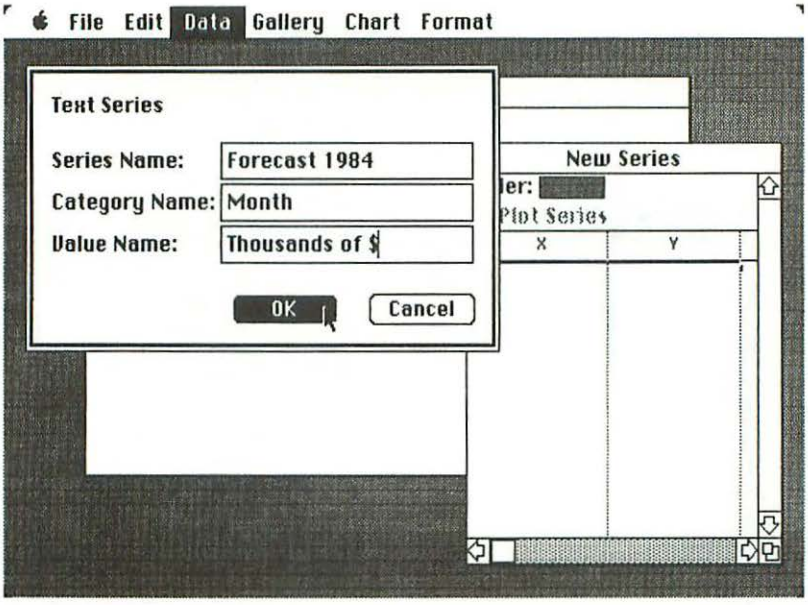
For the moment, it's the window labeled "New Series" that concerns us. The term "series" merely designates a set of pairs of corresponding x-y coordinates. Notice that the window has a column for the categories on the left (x-axis), and a column for the values on the right (y-axis). But, before being able to insert any information concerning the new series we intend to create, we must tell the machine what sort of data we are going to propose. This is done by pulling down the **Data** menu and choosing the **Text** command, as shown in the following illustration:



This informs Chart that we intend to refer to the categories on the x-axis, not by numbers (as might have been expected), but by textual labels: in fact, by the abbreviated names of the twelve months of the year. Notice that, besides Text, there are three other commands at the top of this menu: **Sequence** puts a simple arithmetic series of numbers in the left column, such as 1, 2, 3, etc. **Date** inserts an actual series of precise dates into the left column; and **Numbers** allows you to propose any numeric terms whatsoever for the x-axis.

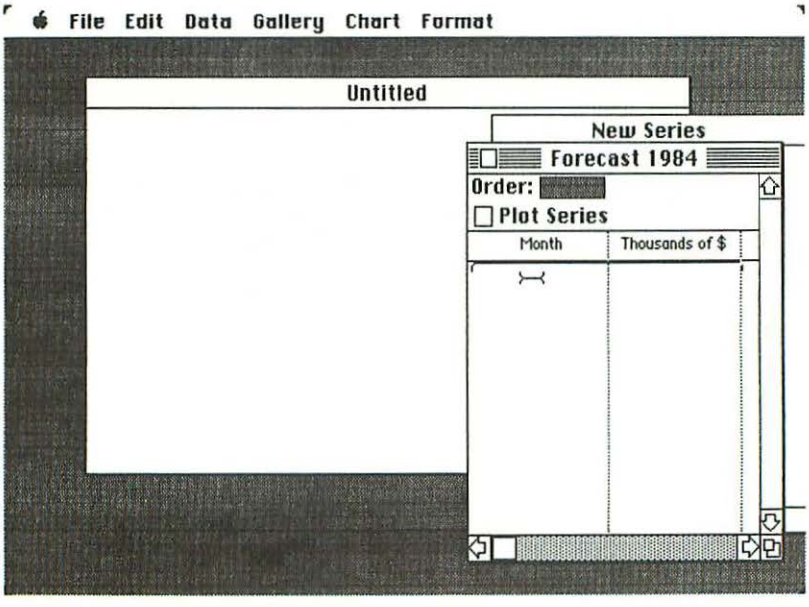
As far as the values along the y-axis are concerned, only numbers are allowed.

The Text command offers us a dialog box (see following illustration) that allows us to name the future series itself as well as the two axes.



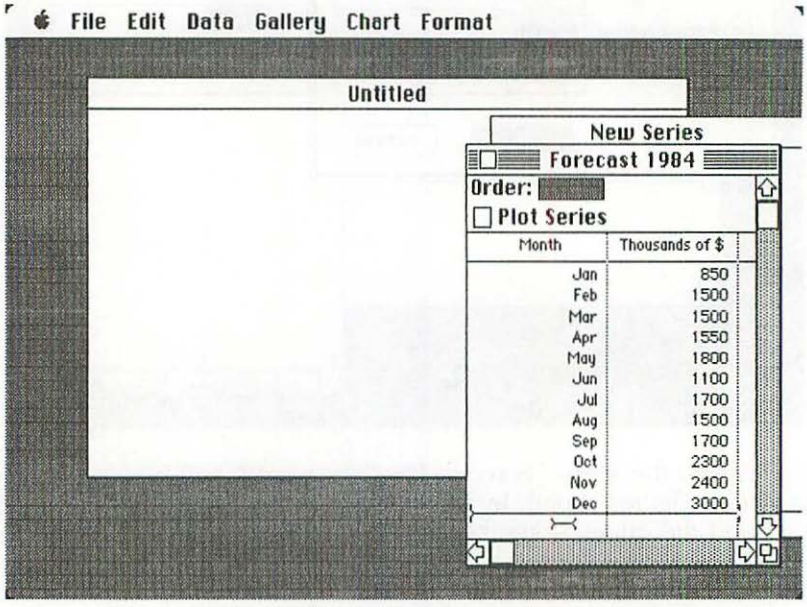
We have named the series "Forecast 1984", because it will be composed of the projected sales figures, month by month, for that year. The x-axis is to be called "Month", and the values along the y-axis are to be represented as thousands of dollars.

Once the **OK** button is clicked, Chart will present us with another window, named "Forecast 1984", as shown in the following illustration:



Notice that the window labeled "New Series" is tucked away just behind this latest window. The letters x and y have been replaced by the axis names that we just indicated, and there is a curious little pointer that marks the spot where we are about to start entering the names of the twelve months.

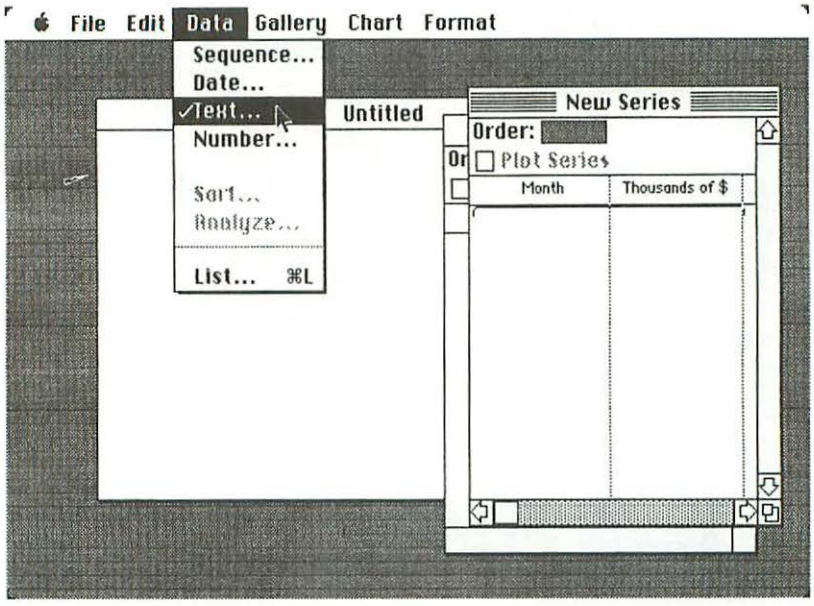
The two columns might be filled in as shown in the following illustration:



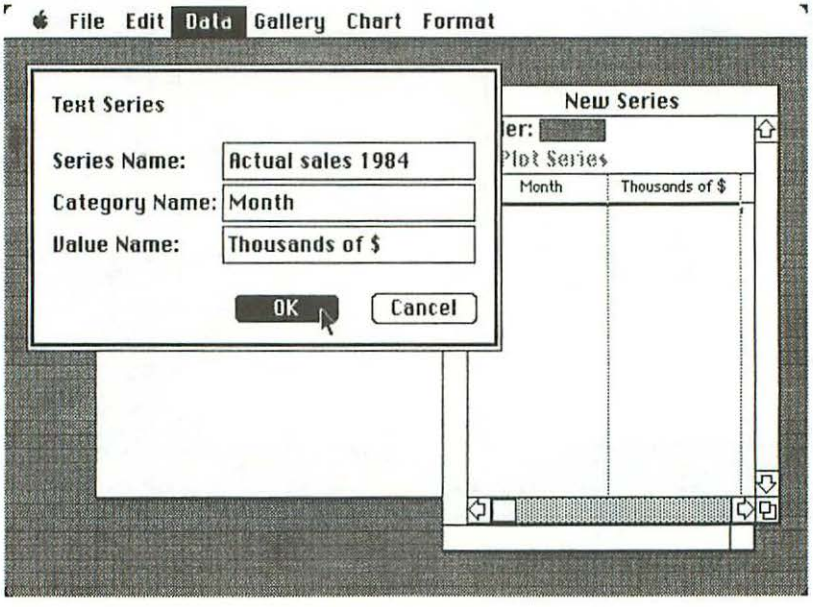
In creating this series, you use the mouse to move the pointer from one column to the other, or from one line to the next, and then you have to click it. If you do not place an explicit numeric value in the righthand column, Chart automatically inserts a zero.

The next step in our demonstration consists of creating a *second* series containing the actual sales figures for the year.

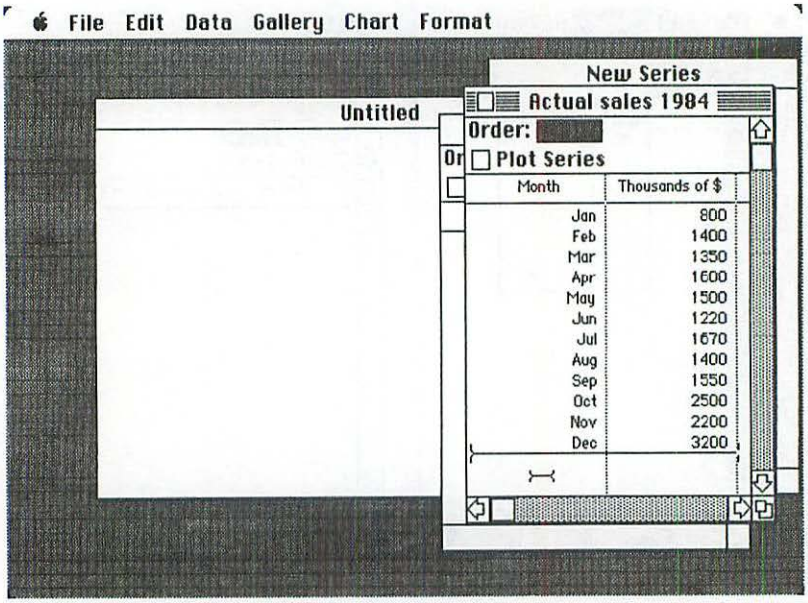
There is a little trick that you must not forget in order to carry out this transition successfully (that is, without accidentally destroying the series you just created). It consists of clicking the title bar of the hidden "New Series" window, to make it active. When this happens, it's the "Forecast 1984" window that gets tucked away behind the "New Series" window, as shown in the following illustration:



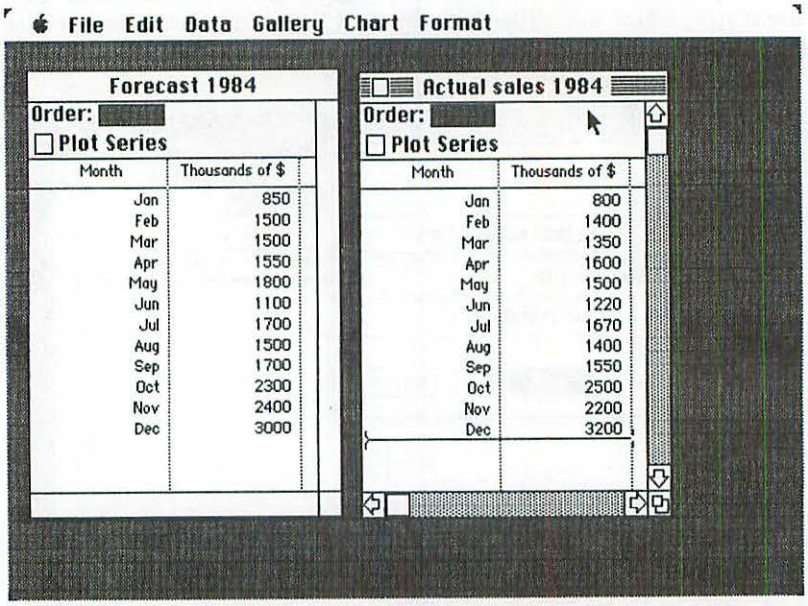
Now you can pull down the Data menu once again and choose the Text command. Fill in the name "Actual sales 1984" for this new series (see following illustration), and click the **OK** button.



This second series might be completed as indicated in the following illustration:



Now you can put both series side-by-side, as shown in the following illustration:

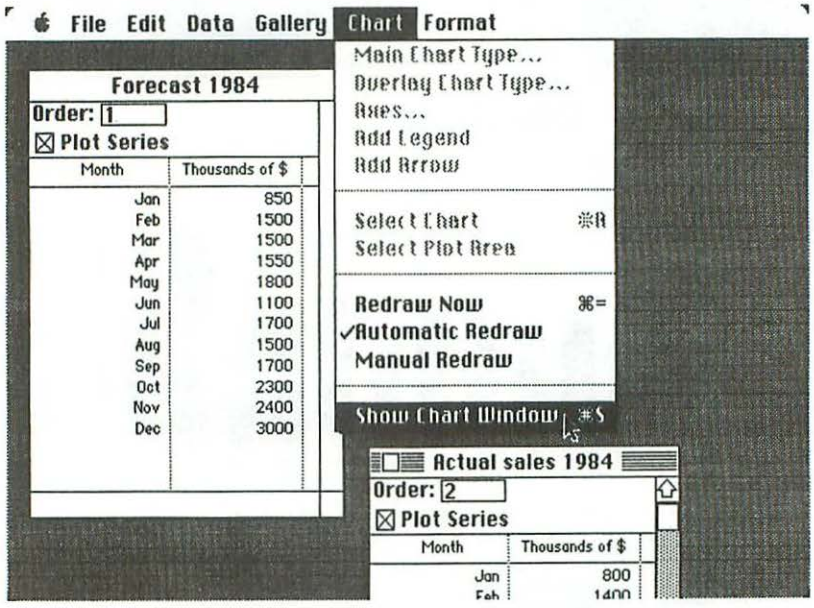


(In case you're wondering what has happened to the windows named "New Series" and "Untitled", they have simply been reduced to small rectangles and tucked away out of sight behind the two windows we have just created . . . so as to minimize the clutter on the desktop.)

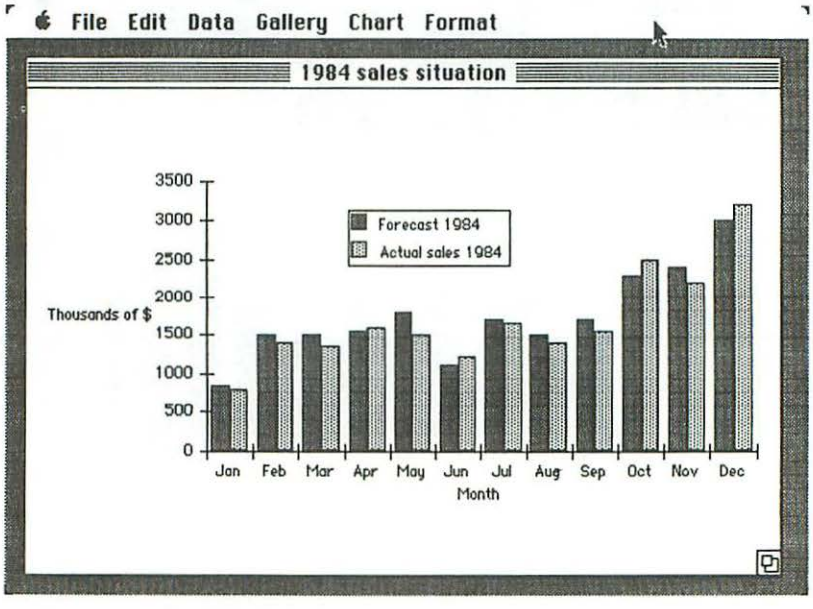
The next step is to tell the machine which series is to come first when the chart is drawn. This is done by clicking the little square boxes just to the left of the expression "Plot Series". The logical solution would be to put the forecast values

before the actual values, so first activate the window labeled "Forecast 1984" and then click its "Plot Series" box. Next, activate the window labeled "Actual sales 1984" and click its "Plot Series" box.

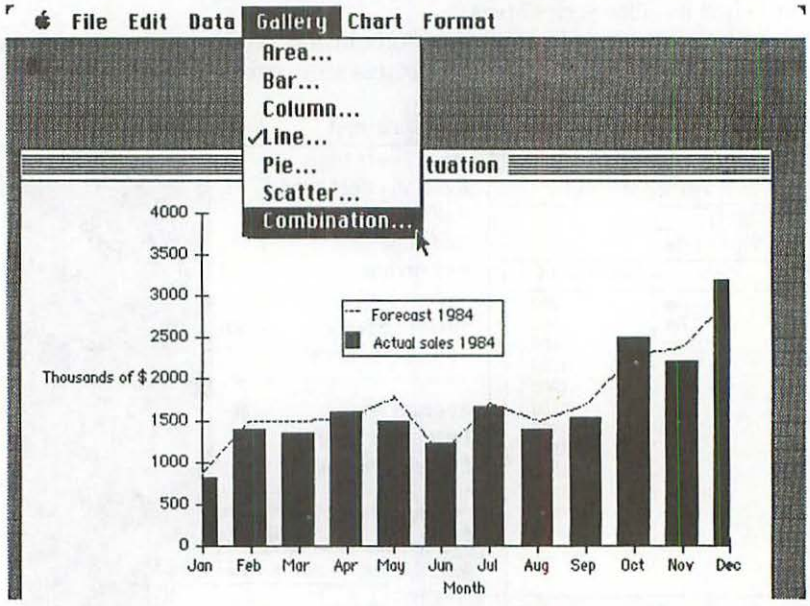
Chart carries out its plotting operations in the order you requested, and the "Order" boxes in both windows will reflect this sequence, as shown below:



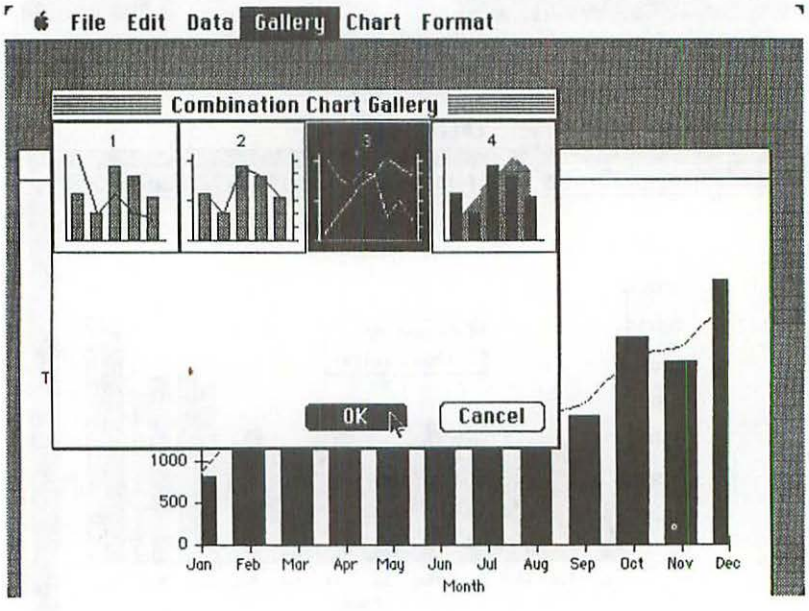
Now pull down the **Chart** menu and choose the **Show Chart Window** command in order to see the results, which are shown in the following illustration:



This is a chart of the so-called *Column* variety, but other models can be chosen by means of the **Gallery** menu, as shown in the following illustration:

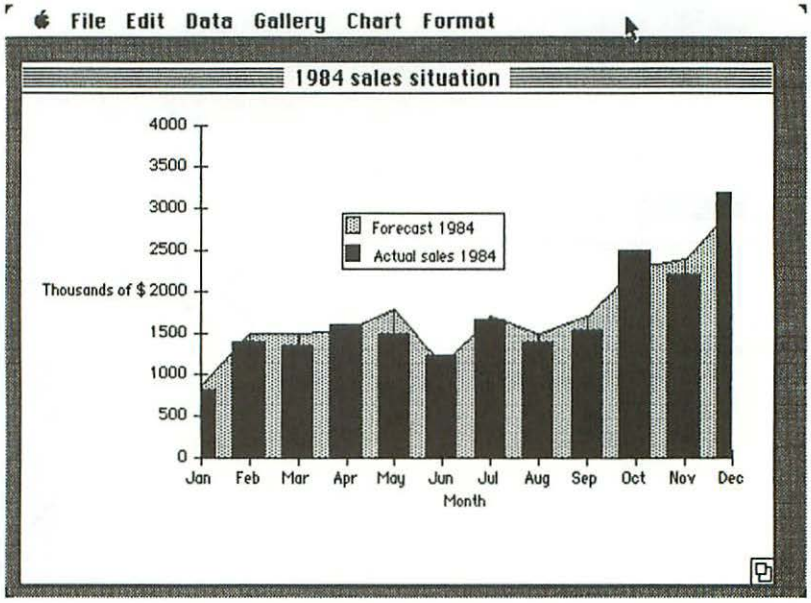


If you ask for the *Combination* variety, for example, the following set of four models is displayed in a window on the screen:

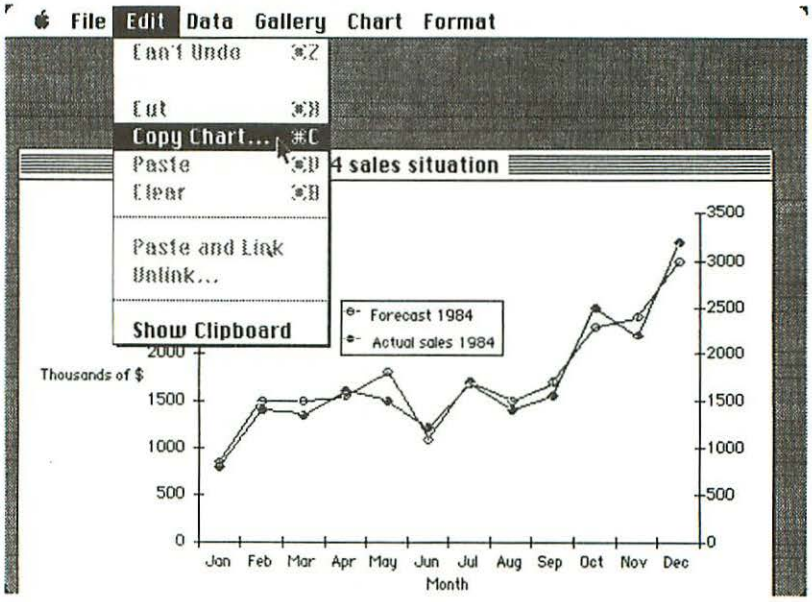


You click the model you want, then you click the **OK** button to obtain the rapid transformation of your chart into the requested style.

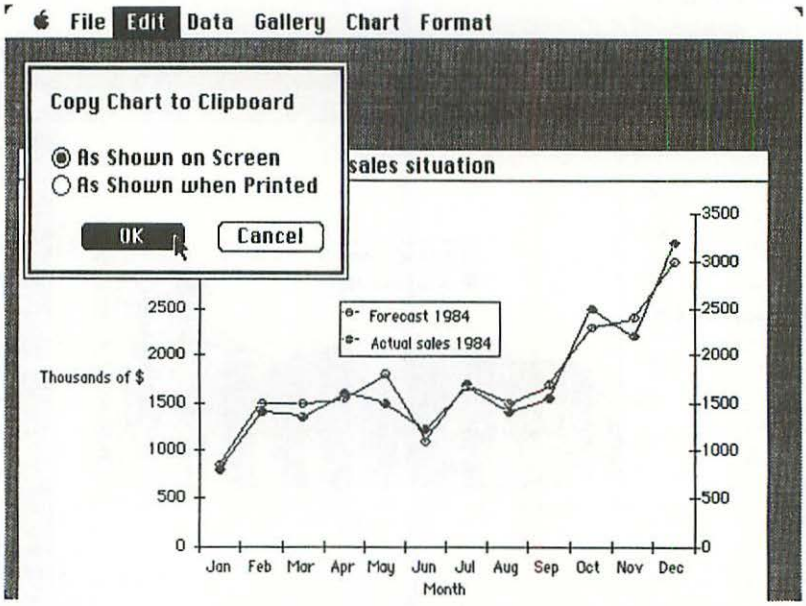
One of the most attractive models, in the case of our example, is shown in the following illustration:



Charts can be copied onto the Clipboard (see following illustration), enabling you to use them as graphic input to MacWrite or MacDraw.



Since there's a slight difference in the way they look on the screen and the way they're printed out on the Imagewriter, Chart asks you which solution you prefer, as shown in the following illustration:



There are many other interesting possibilities opened up by the existence of Microsoft Chart. For example, in the Data menu there is a **Sort** command that enables you to change the order of both categories and values. And there are half-a-dozen arithmetic and statistical calculations that can be called upon for any series.

Finally, it should be pointed out that there are links between Multiplan and Chart that enable you to actually visualize worksheet results, most effectively, by means of charts.

chapter 10
"MacProject" — Project Planning

Like Multiplan, the *MacProject* tool — created by Apple Computer — takes us into the domain of computer modeling and simulation. MacProject is a close cousin of *PERT* (Program Evaluation and Review Technique), which is a system of scheduling interdependent events in such a way that they combine together as efficiently as possible in order to complete a global project.

The events that might be incorporated into a MacProject model can be quite heterogeneous, and of any nature whatsoever. Maybe the only thing they have in common is that each event necessitates a certain amount of time. Apart from that, we are faced with a situation in which certain events can only start to take place once other specific events are completely terminated. This planning tool is designed to handle ideally the sort of project in which a certain number of team members — working concurrently, almost in parallel, on quite different tasks — must harmonize their efforts so that no particular individual is systematically slowing down all the other members of the team through inadequate planning. In other words, MacProject might be thought of as a tool for *operations research.*

Instead of theorizing any further about this excellent product, let us look at a simple case study using MacProject.

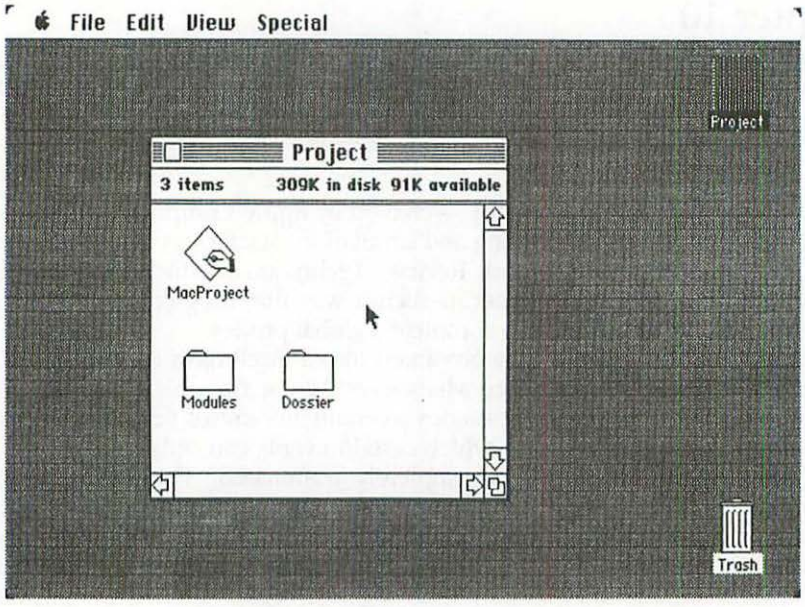
Demonstration

Our example, this time, concerns an author who — on January 26, 1984 — started to write the manuscript of a book about an interesting personal computer that had recently come into existence. He figured that the actual job of producing the manuscript would take no longer than six weeks, because he intended to use a word processing product to facilitate the task. The author had several friends with a remarkably thorough knowledge of the new computing machine (because they worked, in fact, for the company that manufactured it), and these friends had promised him to take no longer than a week, once the manuscript was given to them, to review it and to pull out all the bugs they could find. At the same time, the author planned to send his editor a copy of the draft manuscript, so as to receive further suggestions for improvements. This latter operation was almost certainly going to take two weeks, because the author and his editor actually lived

in two different countries, and the mail service was far from perfect. The author reckoned that he could handle all corrections to the manuscript in the course of a week. And since the publisher and his printer intended to use avant-garde computerized techniques for the actual production of the book, they expected to have it in the book stores no more than three weeks later.

So, let's see how this project was planned with the help of MacProject.

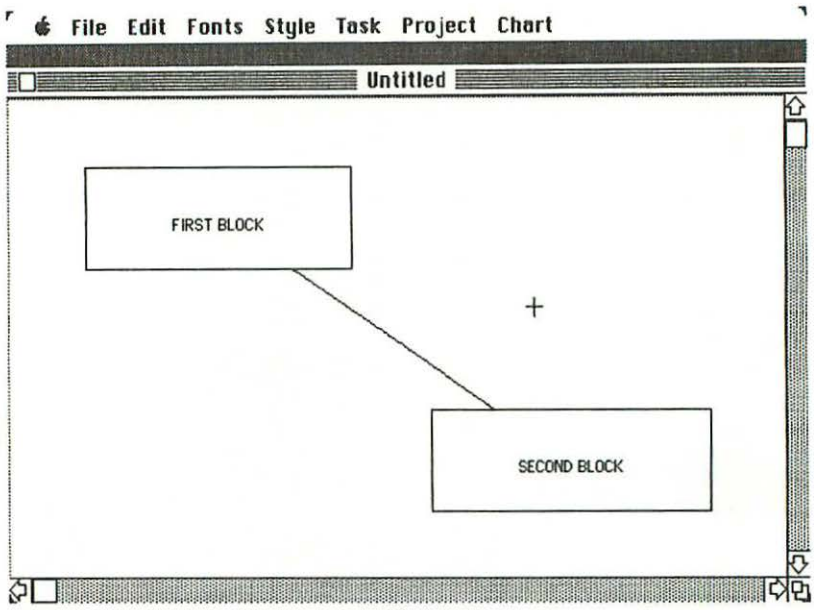
The directory of the "Project" disk is shown in the following illustration:



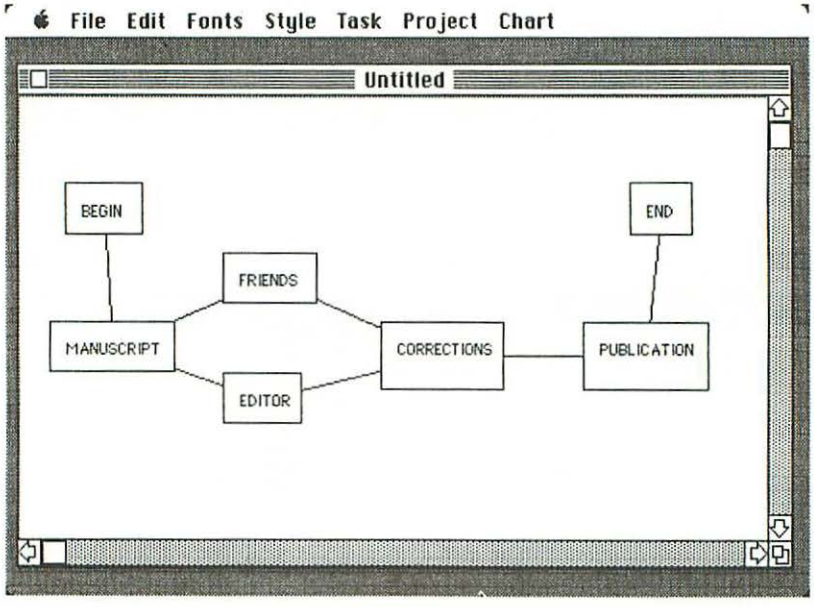
The starting point, with MacProject, is to draw a set of connected blocks that constitute the *flow diagram* of the project in question.

(As in the case of MacDraw, our example will be limited to a flow diagram that could be printed out on a single sheet of paper, but there is a **Drawing Size** command that makes it possible for a single MacProject document to extend over a few dozen sheets of paper.)

Now, it's easy to use this tool to draw connected rectangles. You position the pointer (a cross), then hold down the mouse button and drag the pointer in a "rubber banding" manner. The line that connects rectangles (see following illustration) is obtained automatically and rapidly by dragging the pointer from one rectangle to another. Text is entered by clicking the pointer inside a rectangle and then using the keyboard.



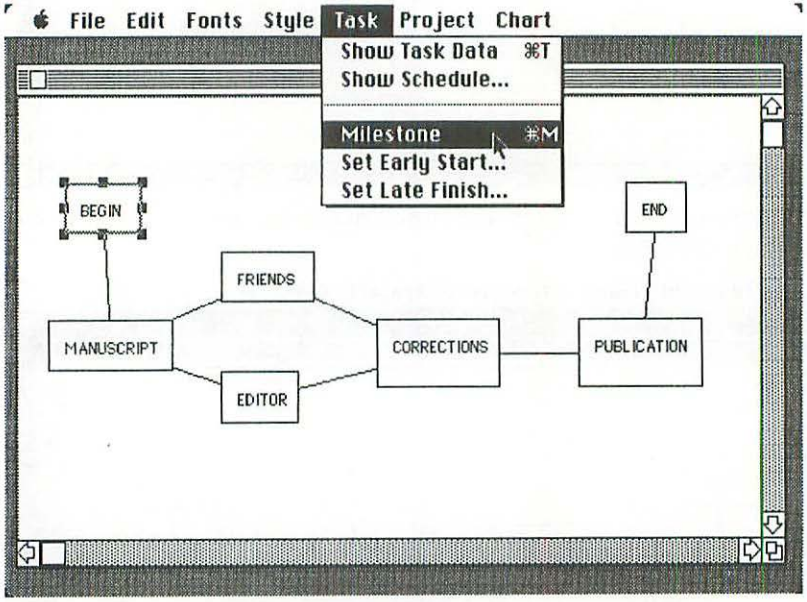
The basic structure of the MacProject flow diagram for our case study is shown in the following illustration:



The two blocks labeled *BEGIN* and *END* can be thought of as simple delimiters . . . which means that there are really only five major blocks in the structure.

MANUSCRIPT, as the name suggests, designates the six-week task of creating the initial manuscript, carried out by the author. *FRIENDS* is the name of one of the following operations, planned to last a week, during which the author's friends review the draft manuscript. *EDITOR*, at the bottom, is the name of the two-week phase during which the author's publisher carries out an analogous reading of the draft manuscript. The author intends to wait for both copies of the draft manuscript — so that he can compare the critiques — before getting involved in the task labeled *CORRECTIONS*, planned to last for a week. And then the final task, labeled *PUBLICATION*, will involve the publishing house and the printer for three weeks.

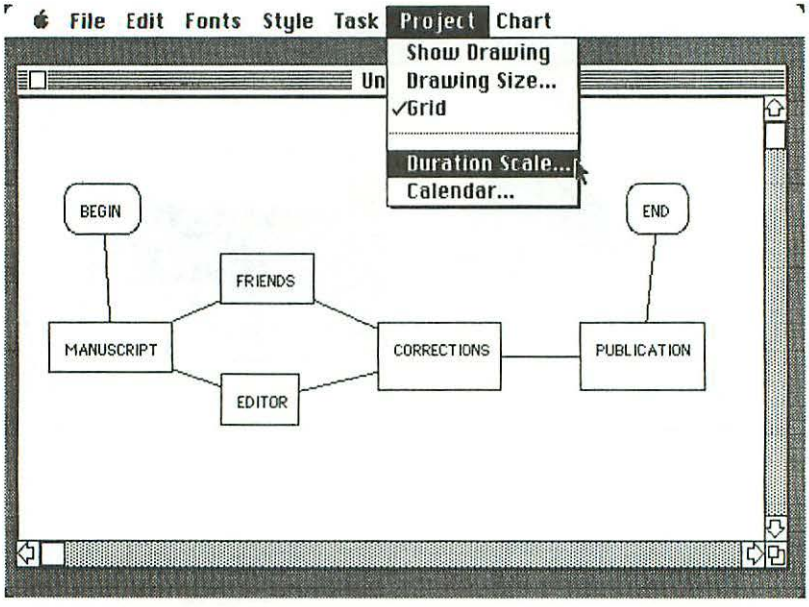
The *BEGIN* and *END* delimiters are referred to, in MacProject jargon, as "milestones". Each block can be selected by bringing the pointer to rest on the upper edge of the rectangle (see following illustration). The selection process results in the appearance of handles, just like in MacDraw.



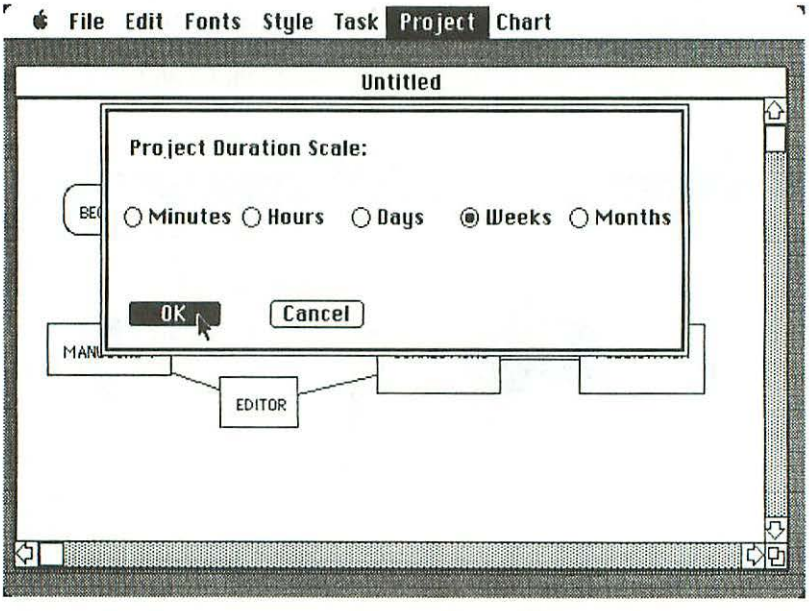
The **Task** menu is then pulled down, and a **Milestone** command makes it possible to give the two delimiters a distinctive round-cornered appearance, so that they won't be confused with authentic blocks.

Time is a fundamental notion in the MacProject context, and so the first thing we must do is to indicate to the machine what sort of time scale we're dealing with in the present example. Is it a matter of minutes, or rather months, or something in between?

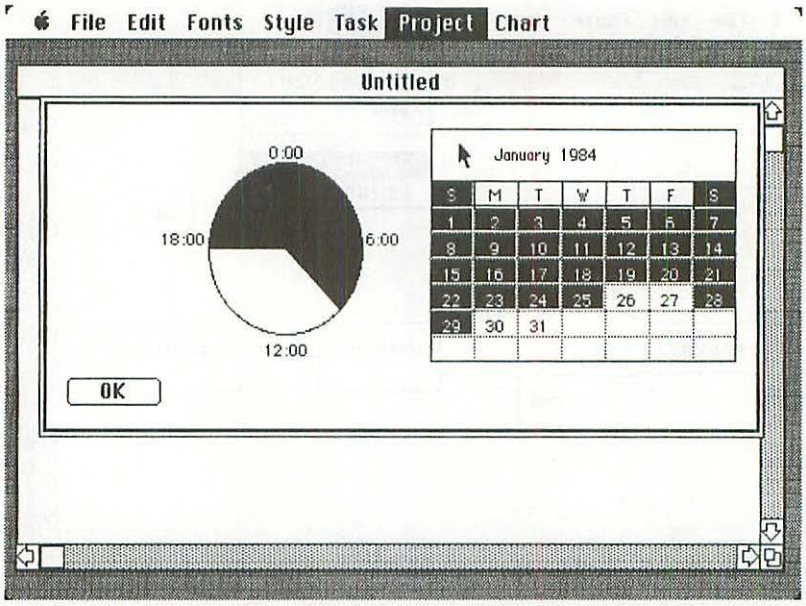
To give this information to MacProject, we pull down the **Project** menu and choose the **Duration Scale** command, as shown in the following illustration:



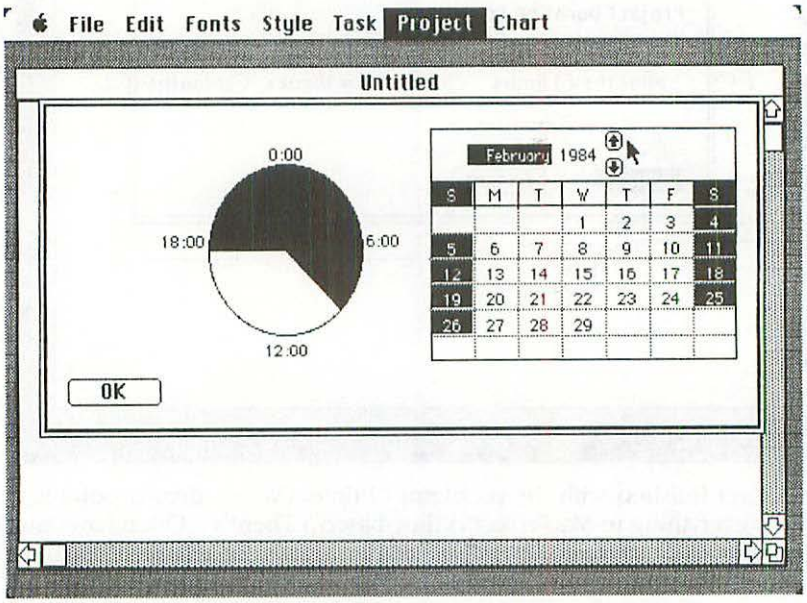
We select one of the five alternatives, by clicking its radio button (see following illustration), and then we click the **OK** button to get rid of the dialog box.



We're not yet finished with the problems of time. (We've already pointed out that practically everything in MacProject is time-based.) There's a **Calendar** command in the Project menu, and the following illustration shows the result of choosing it:



If we click around on this dialog box for long enough, we end up informing the machine — as indicated in the preceding illustration — that the author intends to start work on January 26, and that he intends to spend nine hours a day in front of the keyboard of his word processor. (Think of that black-and-white shape on the left as a 24-hour clock.) Instead of clicking the **OK** button, we need to click the word "January", at the top of the calendar, in order to move on to the next month, as shown in the following illustration:

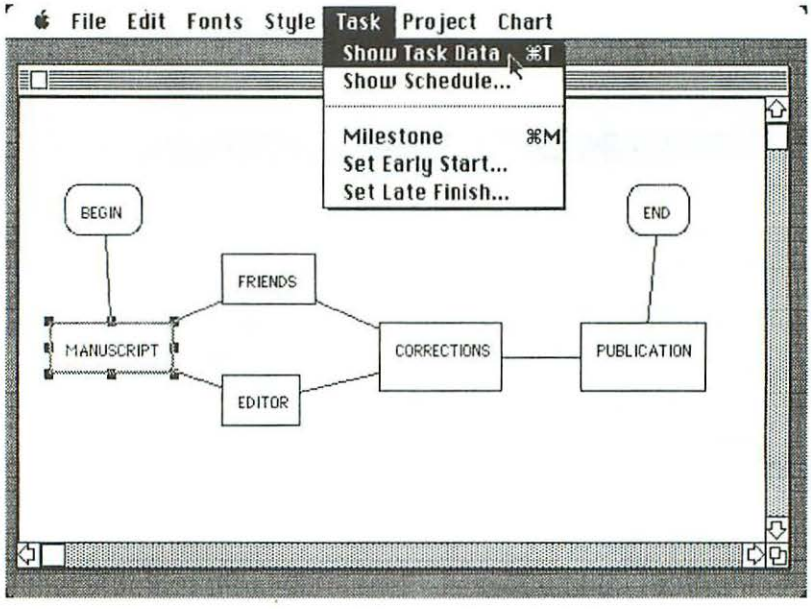


Here the author has indicated his intention to work for 21 days during the month of February. (Notice that Macintosh is smart enough to know that there are 29 days in February 1984.)

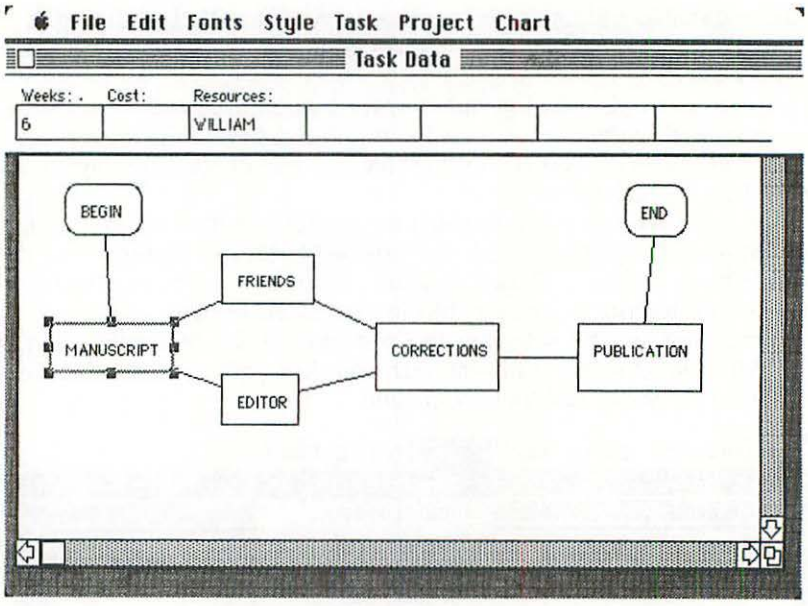
Click the upwards-pointing little arrow to reach the next month, and so on. (Unless you have particular reasons for supposing that things never turn out as planned, don't bother going any further than the end of April. Simply click the **OK** to escape from the calendar.)

We at last reach the point at which we can tell the machine something about the five phases of our project. Let's use precise MacProject terminology. The thing that we've been calling a "block diagram" is referred to more precisely as the *Schedule Chart*, and our five major "blocks" are called *tasks*.

We can start out by selecting the first task in the Schedule Chart, named *MANUSCRIPT* (see following illustration), and then pulling down the Task menu and choosing the **Show Task Data** command.



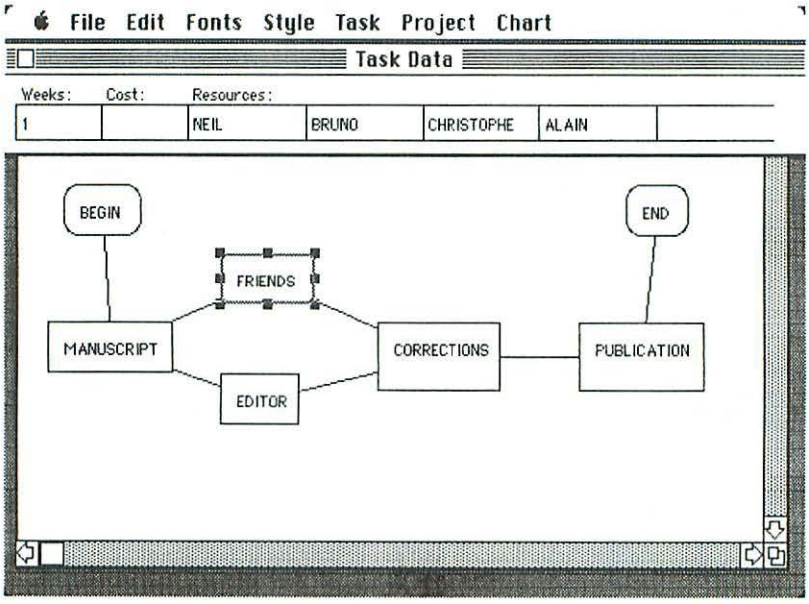
In fact, MacProject does not really intend to *show* us data concerning the *MANUSCRIPT* task . . . because nobody has told the machine anything yet about any one of the five tasks. All that MacProject can "show" us is a long line of narrow rectangles, towards the top of the screen (see following illustration), which are completely empty when they are first displayed. On the contrary, it's *we* who have to "show" MacProject what specific values we wish to find in these rectangles.



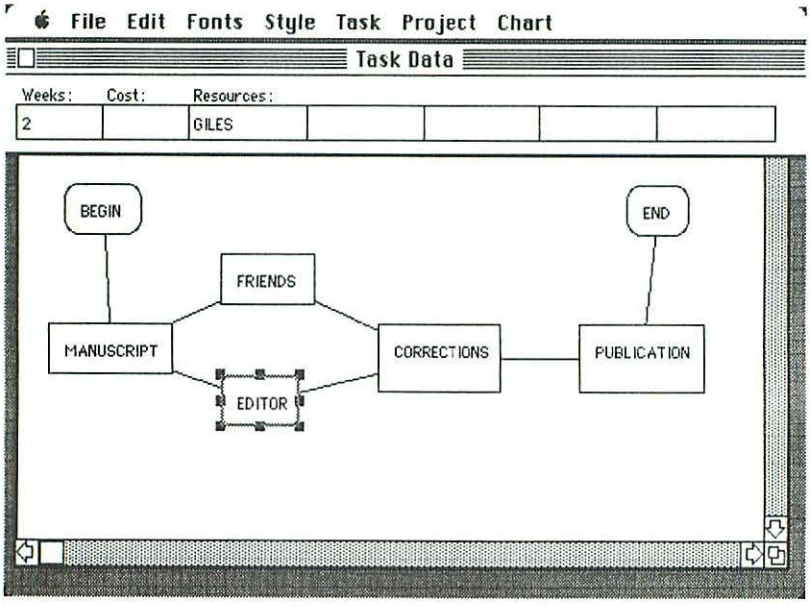
The first element of data to be entered is the number of weeks that the *MANUSCRIPT* task is planned to last. We type the number 6 and then hit <Tab> to move to the next rectangle. In this particular example, we don't intend to analyse the actual costs of the various tasks (which would be quite feasible in a bigger and more realistic case study), so we'll simply jump over this rectangle by means of another <Tab>. Finally we reach the rectangles for listing the resources required for this task. To simplify things, let's imagine that there's a single manpower resource, named *William*, devoted to this initial task.

We now move on to the task called *FRIENDS* (see following illustration). First it must be selected, as before, and then the empty rectangles must be retrieved using a Show Task Data command.

The time, in this case, is one week, and the names of four manpower resources — *Neil*, *Bruno*, *Christophe* and *Alain* — are mentioned.

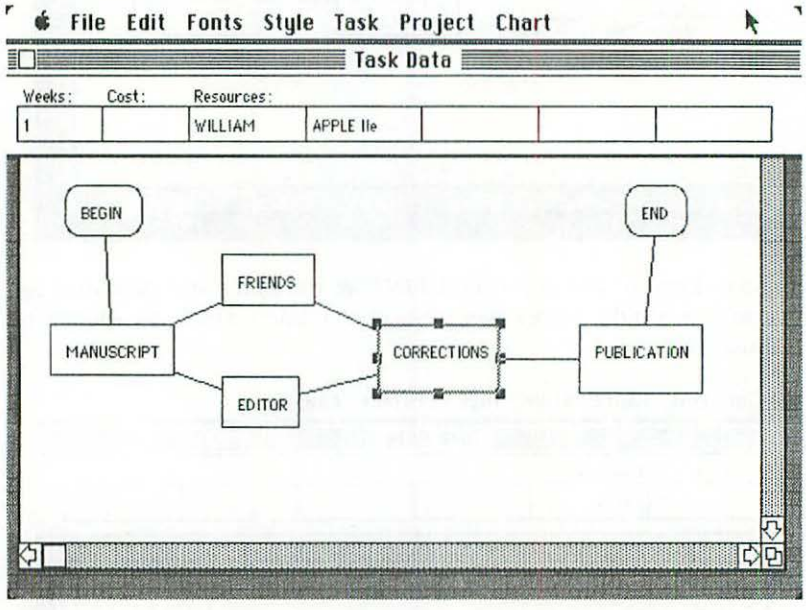


Next, we move down to the task called *EDITOR*, for which we indicate a time of two weeks and a single manpower resource, named *Giles*, as shown in the following illustration:

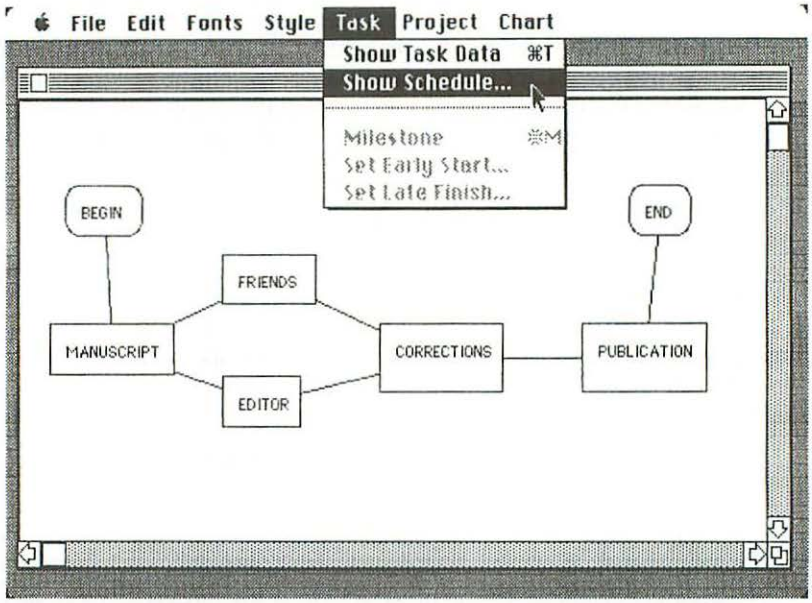


It is important to realize that the two tasks — *FRIENDS* and *EDITOR* — are thought of as starting simultaneously, whereas *EDITOR* is planned to continue for one week longer than *FRIENDS*.

As we have already pointed out, it is planned that the task named *CORRECTIONS* (see following illustration) should not start before both the preceding tasks are terminated. This means that the task called *FRIENDS* represents a special case in this Schedule Chart, because it alone could be executed up to a week late, and that still wouldn't provoke any overall delay in the planned global schedule. In other words, the path that leads through the task *FRIENDS* is not as *critical* (to use the jargon of this branch of computer modeling) as the path leading through the task *EDITOR*.

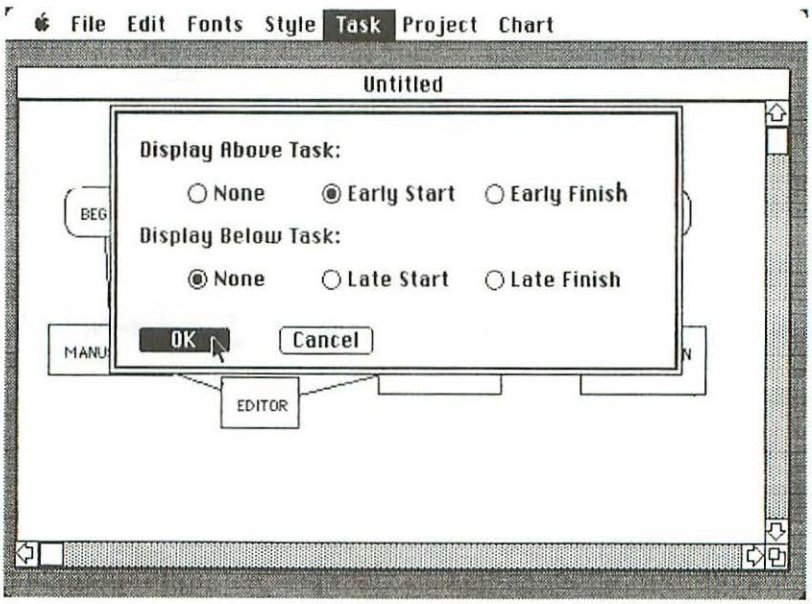


Finally we reach the *PUBLICATION* task (see following illustration). It is planned to last three weeks, and to call upon a large publishing house and a printer as its principal resources.



Notice that we have pulled down the Task menu and chosen a **Show Schedule** command.

The MacProject entity known as a *schedule* is simply a dialog box, as shown in the following illustration:

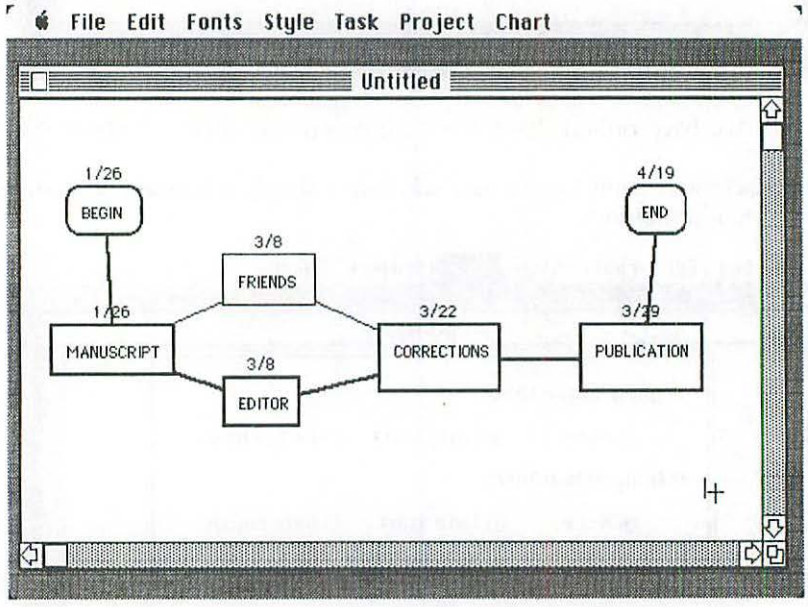


The three upper phone buttons concern the presence (or absence) of an indication to the top of the rectangles representing the tasks, and the three lower buttons concern things you might like to display underneath the task rectangles.

Four notions are involved, and you can grasp them intuitively by imagining, say, your arrival at the office in the morning, and your departure in the afternoon, as follows:

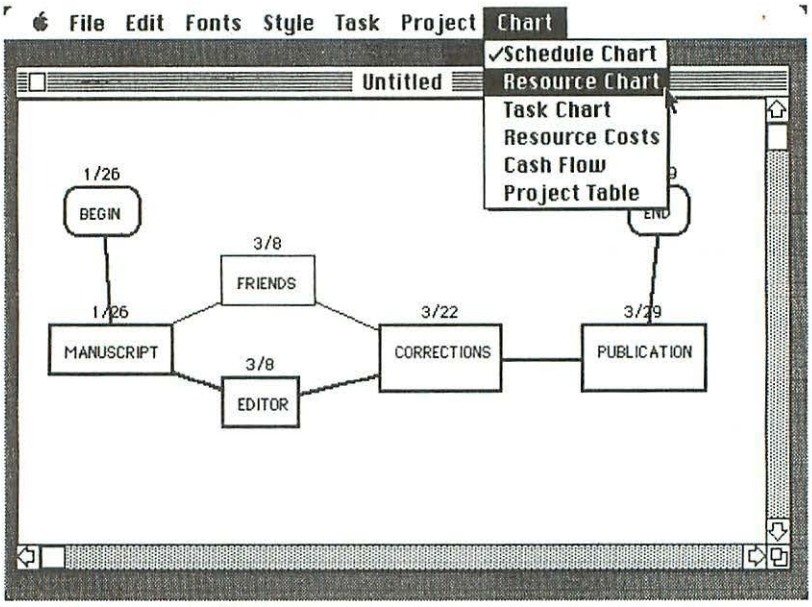
- *ES* (Early Start): The earliest time you could arrive in the morning and hope to be let in the door to start work.
- *EF* (Early Finish): The earliest time you could hope to leave in the afternoon after having completed your day's work.
- *LS* (Late Start): The latest time you could arrive in the morning and still be able to envisage the possibility of completing a full day's work.
- *LF* (Late Finish): The latest time you could be obliged to stay in order to complete a full day's work.

We have chosen to indicate the *ES* date above the rectangles, but nothing else. The result is as follows:

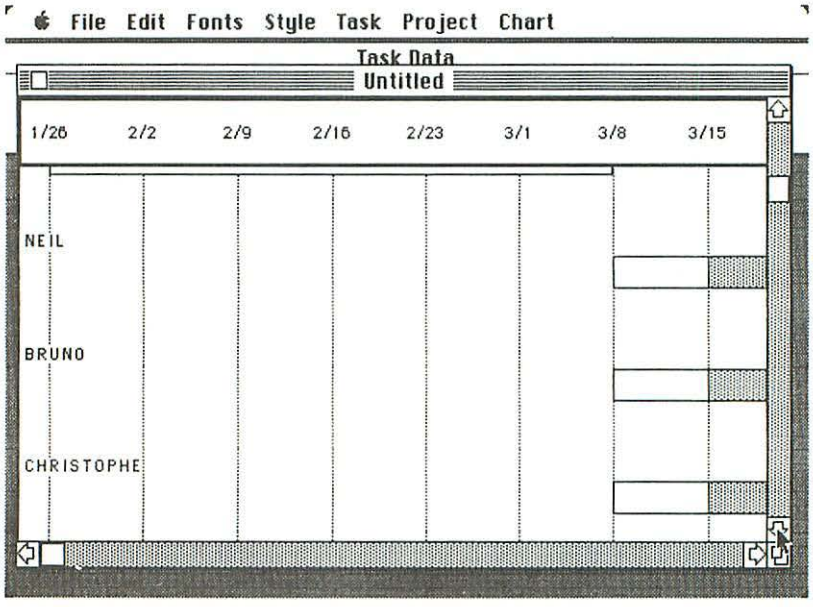


In other words, if all the collaborators work as planned, the author should start to write his manuscript on January 26, and the book should be on sale by April 19.

There is a **Chart** menu (see following illustration) that introduces several interesting overall viewpoints on MacProject features.



The **Resource Chart**, for example, provides us with a bar-chart indicating the exact periods when the various resources of the model are being exploited. Since this bar-chart is too big to fit on the Macintosh screen all at one time, you have the opportunity of scrolling through it. A little piece of this chart, concerning the first two months of the project, is shown in the following illustration:



For MacProject users who are interested in costing out the various tasks, the options of the Chart menu called **Resource Costs** and **Cash Flow** could be of great service.

Conclusions on MacProject

Two distinguished American critics who reviewed Lisa several months ago described LisaProject — the ancestor of MacProject — as "perhaps the most fascinating of the software packages included with the Lisa" (Roger Wagner & Joe Holt, *Softalk*, September 1983). It's tempting to follow their example — were it not for marvelous MacPaint! — and to declare that MacProject is in many ways one of the most *sophisticated* kinds of software that has ever found its way into the personal computing world up until now. It might be said that the illustrious cousins named MacWrite and Multiplan "merely" manipulate, respectively, such entities as words and numbers, whereas MacProject and its likely descendants seem all geared to tackle the manipulation (?) of real-life affairs such as the construction of cities and cathedrals and computers . . . .

Let's refrain from hyperbole, and conclude on a realistic note. MacProject can already perform in a useful manner at the following three levels:

1 For a simple project involving mere *a posteriori* bookkeeping (concerning events that have already taken place), MacProject could be a handy documentation tool.
2 MacProject can handle many "What if . . .?" situations and questions just as well as Multiplan. One might simulate and study, for example, the possible consequences of an exceptional time delay at some strategic point in a forthcoming project.
3 In the case of a complex real-life project, this tool could be used in its original role as an instrument for *critical path analysis*. This term, "critical path", can be applied to any phase of a project in which the slightest delay would automatically affect the outcome, timewise, of the global project.

For these three reasons, MacProject must be considered as an interesting product in the present Macintosh catalog, and as a likely taste of things to come. . . .

EPILOG

We have examined six application tools, four of which — MacWrite, MacPaint, MacDraw and MacProject — are proposed by Apple Computer, whereas the other two — Microsoft Multiplan and Chart — have been created by a software company. These six products make up a pretty solid basis on which to start out using Macintosh as a productivity tool. But the story doesn't end there, because a great many software developers throughout the world are actively engaged in the creation of Macintosh products in many domains.

As far as Apple Computer itself is concerned, there is another major software tool for Macintosh: *MacTerminal*, which enables Macintosh to emulate IBM, DEC and TTY terminals.

Besides Multiplan and Chart, the Microsoft Corporation is offering Macintosh versions of their record-keeping product called *File* and their word processing system called *Word*.

The Software Publishing Corporation is implementing the entire range of their well-known *Pfs* tools on Macintosh: *Pfs: File* for file management, *Pfs: Report* for printing reports, *Pfs: Write* for word processing, and *Pfs: Graph* for graphics.

The Lotus Development Corporation is creating a Macintosh version of their best-selling *1-2-3* program.

And many other software companies and independent developers are working on Macintosh products that are destined to reach the market in the course of "Macintosh Year": 1984.

Apple Computer, in any case, has been investing heavily in its relationships with third-party software developers, and encouraging them by every means to create products for the new machine. This is understandable in the sense that — as the chairman of the board, Steve Jobs, puts it — "Apple's future is all Macintosh".

appendix A
Software Development Using Lisa

Programs for most microcomputers, up until now, have been written using the same machine on which they are intended to be run. But this familiar method of software development — which appears to be almost "natural" — can create problems at times. After all, it's a bit like attempting to build a kitchen table in the kitchen itself, while using the half-built table as a workbench; it's theoretically possible, but things would be ever so much easier if the builder had a workshop and a proper set of tools for the job, including a good solid workbench.

Most professional software designers are accustomed to the method that consists of using a "big" computer to develop programs for a "little" machine.

Microsoft, for example, used a DEC PDP-11 to develop software for the Apple II. The advantages of the method are obvious: the relatively powerful facilities of the development machine serve as a sort of *metalanguage* enabling the software designer to freely manipulate the language of the target machine.

In the case of the target machine called Macintosh, the perfect development machine is *Lisa 2*.

Development context

Lisa 2 — announced by Apple Computer on exactly the same day as Macintosh —is an improved version of Lisa, the revolutionary machine that had been on the market for about a year. The most striking change is that Lisa's two (5¼") disk drives have been replaced, on Lisa 2, by a single drive that accepts the Sony (3½") disks used by Macintosh.

The basic model of Lisa 2 has a main memory of 512K, and its only disk device is the above-mentioned Sony drive. But a more powerful model, called *Lisa 2/5*, is required for the development of Macintosh software. Lisa 2/5 differs from the basic model in that it has an external *Profile* disk drive with a capacity of 5M. To develop Macintosh programs on this Lisa 2/5, you also have to purchase a kit that extends its main memory to 1M, and it's not a bad idea to have a second Profile. This means that we are speaking of a development machine that costs over three or four times as much as the target machine.

To write programs for Macintosh, using Lisa 2/5, you can use either *Pascal* or *assembly language*, or both. Not only can assembly language routines be nested inside Pascal modules; inversely, too, Pascal routines can be called by assembly language modules.

Lisa Pascal is quite close to — but more powerful than — the language defined by Kathleen Jensen and Niklaus Wirth in the *Pascal User Manual and Report* (2nd edition), Springer-Verlag. The precise specifications of the Lisa version of Pascal are provided in an Apple document entitled *Pascal Reference Manual for the Lisa*. Assembly language programmers can obtain information on the machine instructions of the MC68000 processor in the book by Lance A. Leventhal entitled *68000 Assembly Language Programming*, Osborne/McGraw-Hill, or in the smaller *68000 Microprocessor Handbook* by Gerry Kane, at the same publishing house.

In order to simplify the presentation of this subject, the only development language that will be mentioned in the rest of this appendix is Pascal. But it must be remembered that there is an assembly-language version of every aspect of Macintosh software development on Lisa 2/5.

The actual program development activities — such as editing, compiling, assembling, linking and debugging — are carried out on the Lisa 2/5 in an environment called the *Workshop*, which is a set of tools booted from the ProFile. A complete description of the functions of this development environment is provided in an Apple document entitled *Workshop User's Guide for the Lisa*.

The Macintosh program that is being developed can be transferred periodically from Lisa 2/5 to the target machine, by means of a cable, in order to be run and tested.

Pascal particularities

An experienced Pascal programmer should be able to use Lisa Pascal, to create Macintosh software, with no particular difficulties.

It's necessary to introduce a few technical remarks here concerning the machine itself. The memory of Macintosh is made up of two parts: a 128K *random-access memory* (RAM), which will almost certainly be upgraded to 512K as soon as suitable chips become commercially available in large quantities and at a reasonable cost; and a 64K *read-only memory* (ROM). The tightly-coded information written into this huge chunk of ROM, during the manufacturing process, constitutes one of the major assets of Macintosh. One third of it constitutes part of the Macintosh *operating system*; another third, called *QuickDraw*, is a set of graphic routines which provide the basis of all the screen display activity of Macintosh; and the final third, referred to as the *Toolbox*, is a library of management routines used by software developers to take care of all the sophisticated features that you find in top-quality applications such as *MacWrite* and *MacPaint*.

As far as the programmer is concerned, these ROM routines — which are divided into fifteen *units* (in the Pascal sense of the term) — constitute a stock of 466 *procedures* and *functions* whose identifiers can be used in Lisa Pascal programs.

The voluminous documentation for developers, entitled *Inside Macintosh*, provides a complete semantic and syntactic description of each of these procedures and functions, together with the definitions of various predefined Pascal *types* and *constants* that can be used by the programmer.

There is, however, one particular subject — *dynamic storage allocation* — that may appear a little strange to some programmers, since Lisa Pascal introduces several concepts that are not present in the conventional language.

Dynamic variables and *pointers* exist already in standard Pascal, where they are used above all for building up sets of linked cells for list processing. But Lisa Pascal goes one step further, and introduces the notion of so-called *handles*, which are in fact pointers that point to . . . other pointers!

Why are such complicated entities necessary? The answer can be found in the section of the documentation that deals with an important part of the Macintosh operating system, housed in the ROM, known as the *Memory Manager*.

In Lisa Pascal terminology, storage is allocated to dynamic variables in a region of main memory referred to as the *heap*, which is a collection of contiguous variable-length *blocks*, each of which — at any particular instant during the execution of a program — is either *allocated* or *free*. Whenever a block is freed, a "hole" appears in the heap. Now, since many of the blocks are *relocatable* (i.e., the information they contain remains meaningful, no matter where they are located in the heap), the Memory Manager can use housekeeping procedures to shift them around in the heap, in order to "gather together the holes" and to utilize this freed space as efficiently as possible.

In view of this constant reshuffling of the relocatable variables in the heap, Lisa Pascal had to offer the programmer a convenient device for pinning down the exact address of such-and-such a dynamic variable at any particular instant. The solution works as follows, using a method that is referred to in the documentation as *double indirection*. Whenever the Memory Manager allocates relocatable heap storage to a dynamic variable, it creates a *master pointer* that represents the current address of the variable in the heap. Every time that the variable changes its location in the heap, because of housekeeping activities, the Memory Manager corrects the master pointer. Now, all that the programmer has to do is to keep in touch with this master pointer, and this is done by using a pointer — the so-called "handle" — that points at the master pointer.

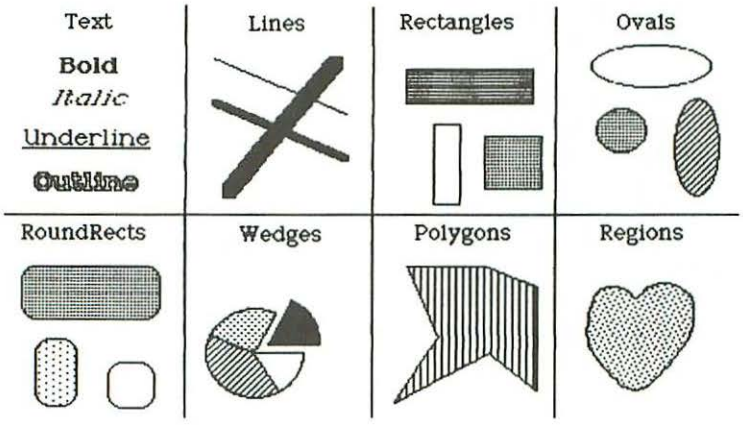
Once you've gotten the hang of this method of indirect addressing — which is not nearly as complicated as it appears to be, at first sight — you no longer need to worry about memory management, and the rest of the job of Macintosh software development is just plain Pascal programming.

QuickDraw

The Macintosh designers did not waste any time looking around for a novel approach to graphics; they simply "borrowed" — lock, stock and barrel — the splendid *QuickDraw* package that a member of the team, Bill Atkinson, had created earlier on for Lisa. Complete details of QuickDraw are contained in an appendix attached to the above-mentioned Lisa Pascal reference manual.

Insofar as everything that appears on the Macintosh screen, including text, must be thought of, in the final analysis, as *graphic* material, it is easy to understand why QuickDraw plays such a fundamental role in the Lisa/Macintosh context. In fact, it's hardly an exaggeration to describe the *MacPaint* and *MacDraw* tools as mere "implementations" of QuickDraw.

By means of QuickDraw, the Macintosh screen can be organized into a number of individual areas, and various graphic objects can be drawn inside each area, as shown in the following illustration:



QuickDraw operates within the framework of a two-dimensional grid that provides the background for three mathematical constructs: *points*, *rectangles* and *regions*. For each construct, QuickDraw proposes a Pascal type. The Point type is defined in terms of integer coordinates, and the Rect type in terms of points, while the Region type is defined as one or several arbitrarily-shaped areas lying within the bounds of a rectangle.

Next, QuickDraw introduces the notion of a *bit image*, which is simply a contiguous set of words in main storage, conceptualized as a matrix in a row-wise sense (with the lowest byte in storage being associated with the upper left corner of the matrix). A *bit map* is a triad — giving rise to another Pascal type — composed of a pointer to a particular bit image, an integer that provides the row length of that bit image, and a boundary rectangle that fixes its height. The next QuickDraw graphic entity is the *pattern*, an 8-by-8-bit rectangle which is used to define a repeating design. Finally, QuickDraw describes a graphic entity referred to as the *cursor* . . . which is simply our familiar Macintosh pointer, viewed as a 16-by-16-bit rectangle and defined in precise terms as a Pascal type.

At this stage, QuickDraw introduces a fundamental record type called *GrafPort*, incorporating no less than 25 fields, which specifies a complete drawing environment. The best way of approaching GrafPorts is to imagine that there is one such entity for every window that might be opened onto the desktop. Two important subsets of the GrafPort fields specify, respectively, the *pen characteristics* for drawing, and the *text characteristics* for data of keyboard origin.

The basic concepts of QuickDraw are completed by the *Picture* type, used for handling "elastic" images that may be stretched or shrunk in order to fit into various rectangular "picture frames", and the *Polygon* type, representing a set of connected points inside a rectangular boundary.

All three primitive shape constructs in QuickDraw — regions, pictures and polygons — give rise to dynamically-allocated storage, and must be accessed through handles.

On these foundations, the QuickDraw package then "explodes" into an amazing assortment of 145 Pascal procedures and functions, which belong to the following nineteen categories:

- GrafPort routines (15)
- Cursor-handling routines (5)
- Pen and line-drawing routines (13)
- Text-drawing routines (12)
- Drawing in "color" (3)
- Calculations with rectangles (10)
- Graphic operations on rectangles (5)
- Graphic operations on ovals (5)
- Graphic operations on rounded-corner rectangles (5)
- Graphic operations on arcs and wedges (5)
- Calculations with regions (18)
- Graphic operations on regions (5)
- Bit transfer operations (2)
- Pictures (5)

— Calculations with polygons (4)
— Graphic operations on polygons (5)
— Calculations with points (6)
— Miscellaneous utilities (8)
— Customizing QuickDraw operations (14).

Toolbox

Of the 466 routines in the Macintosh ROM that can be invoked by Pascal or assembly language programs, 216 belong to what is called the *Toolbox*, which can be broken down into the following eleven modules (presented in alphabetical order):

1 *Control Manager* (26) In Macintosh terminology, controls are special objects on the screen that you can manipulate using the mouse. One typical example is the scroll bar that is used to change the contents of an active window. Another example is provided by the various buttons and check boxes that you encounter inside dialog boxes. Every control is associated with a specific window. Since storage space for relocatable control records is allocated from the heap, they are referred to by means of a control handle.
2 *Desk Manager* (4) These routines enable your software to support the desk accessories: Note Pad, Scrapbook, Calculator, Clock, Puzzle. They can be described as "mini-applications" that can be run concurrently with any Macintosh application.
3 *Dialog Manager* (22) In Macintosh terminology, a dialog box is a rectangle that appears on the desktop whenever the machine needs to communicate with the user in order to obtain essential information. To minimize the effort required on the part of the user, Macintosh usually expects you to simply click some of the buttons and/or check boxes in the dialog box. Quite often there's an **OK** button to let Macintosh know that your reply is ready.
4 *Event Manager* (11) Whenever you perform the slightest act — such as touching the mouse, typing something on the keyboard, or inserting a disk into the drive — Macintosh thinks of it as an event. So, these routines actually manage the man/machine relationship, constantly, at the most primordial level.
5 *Font Manager* (5) Software developers can use these routines to gain access to the same variety of typographical fonts that you encounter in MacWrite.
6 *Menu Manager* (32) One of the nicest things about Macintosh, compared with most other personal computers, is the fact that you can browse through all the menus simply by dragging the mouse along the menu bar at the top of the screen, with no obligation to select anything at all. The entire menu setup is simple and elegant, with its dimmed and check-marked items, its highlighting, etc. This quite large collection of routines allows the software developer to make use of all these user-friendly features.
7 *Resource Manager* (26) In Macintosh terminology, a resource might be defined, rather negatively, as something that is not intimately related to the application being run, even though its presence is necessary. The message in a

dialog box, for example, might be in English, or French, or Spanish. . . . All these elements of secondary information can be placed in modifiable resource files. There are over twenty standard resource types (menus, icons, fonts, pointers, etc.), without counting those that might be invented by you yourself, within the context of a specific application.

8 *Scrap Manager* (6) The cut & paste concept, which is fundamental in the Macintosh context, makes constant use of a device — associated with the desktop, and common to every application — called the Clipboard. Software developers can of course build private cut & paste techniques into their applications, but the routines in this manager allow them to also use the Clipboard.

9 *TextEdit* (21) This is a set of data types and routines for text formatting and editing. They can be inserted in Macintosh application programs whenever it's a matter of entering information from the keyboard.

10 *Window Manager* (42) Windows, in the Macintosh context, are the basic outlet by means of which the machine transmits information to the user. They can be moved across the desktop, overlapped, reduced in size, expanded, hidden, activated, deactivated, etc. This big set of routines makes it easy for software developers to manipulate Macintosh windows.

11 *Toolbox Utilities* (21) To use the Toolbox routines, many manipulations on numbers and strings must be carried out. These routines make it possible to perform such tasks in an optimal manner.

Besides having to understand the semantics and syntax of all these procedures and functions, the software developer is expected to respect various conventions, so that all Macintosh application programs "look alike" (to a certain extent, at least) to the end user. There is no obligation to do so, of course, but it is to everybody's advantage if things turn out this way. Take the case of the I-beam pointer, for example. Because of *MacWrite*, we have all become accustomed to the use of this pointer whenever text is being entered from the keyboard. Now, nothing prevents a software developer from using a different style of pointer in such a situation, or from using the I-beam pointer for a quite different purpose, but this would be regrettable.

One of the fundamental tenets of the Macintosh philosophy — as it was pointed out already at the beginning of this book — is that the user should get to know how to manipulate the machine through, say, *MacWrite* and *MacPaint*, and that all other applications should then appear to the user as "second nature".

Macintosh operating system

Besides QuickDraw and the Toolbox, the 64K Macintosh ROM contains part of the *operating system*. And this, too, can be accessed to a certain extent by software developers.

There are 105 operating system procedures and functions which are accessible by Pascal or assembly language programs. They can be broken down into the following four modules:

1 *Memory Manager* (36) These routines control the dynamic allocation of memory space in the heap. Insofar as the notions of pointers and handles are widely used in the context of Macintosh software development, this module will be frequently invoked.
2 *Segment Loader* (4) Using these routines, you can divide the code of an application into several parts, called segments. The Finder starts an application by asking this module to load the main segment into memory. Further segments are loaded automatically when needed, and storage containing an unwanted segment can be freed by the programmer.
3 *File Manager* (53) This big set of procedures and functions deals with the important question of disk files. There are three ways of handling file I/O on Macintosh: (a) higher-level Pascal calls provide you with simplified facilities that are sufficient in many cases; (b) lower-level Pascal calls enable you to use the File Manager to its fullest capacity, but they are more difficult to handle; and (c) assembly language calls resemble lower-level Pascal calls.
4 *Print Manager* (12) These routines enable the programmer to specify the exact manner in which information is to be sent to the printer attached to Macintosh.

The concept of software components

Back in 1969, a computer specialist named M. D. McIlroy made the following precursory statement: "I would like to see components become a dignified branch of software engineering. I would like to see standard catalogs of routines. . ."

The basic idea behind this remark — which was taken up, much later on, by the creators of the Ada programming language — is that we have long been accustomed to *hardware* components (such as those that you find today on the circuit boards of a microcomputer); so, why shouldn't we be able to adopt the same building-block strategy in the *software* domain?

This is exactly what has happened as far as Macintosh is concerned. Those 466 procedures and functions housed in the Macintosh ROM provide a real and perfect example of software building-blocks with which the developer can construct all sorts of new and unusual applications.

Imagine, for a moment, a Macintosh without any of these ROM-based routines. And imagine that you intend to use this emasculated machine, programmable in 68000 assembly language, in order to develop, say, a graphic design package like *MacPaint*. What a gigantic job! And what a global waste of time and energy if every present and future software developer were obliged to carry on "reinventing the wheel" with respect to such everyday things as menus, windows, scrolling, text editing, graphics, etc.

Thankfully, in the case of Macintosh, the long-awaited concept of software components has finally been brought to fruition. All this tightly-coded ROM-based "MacWare" puts the software developer face-to-face with the oldtime department store slogan: "You name it. We've got it."

appendix B

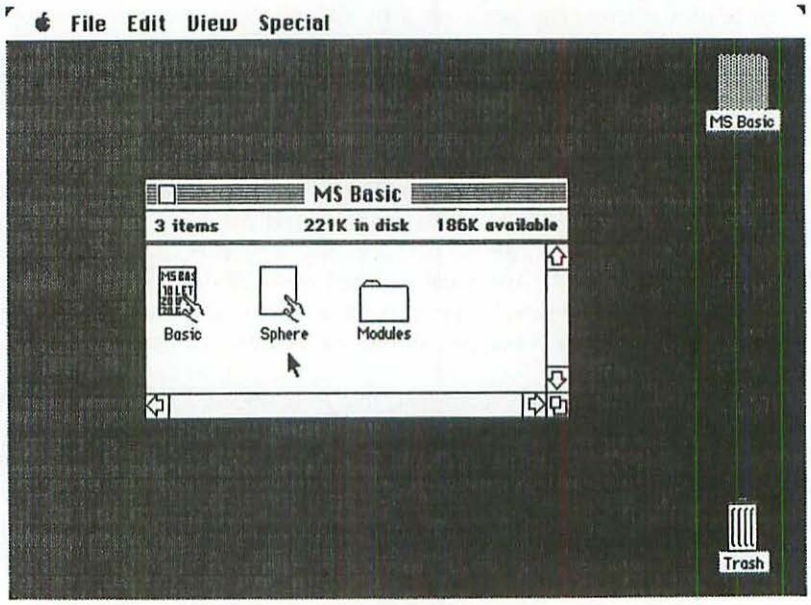
Software Development Using Macintosh

There are quite a few language systems — either existent or imminent — that enable you to develop software on the Macintosh itself: Macintosh Pascal, Macintosh Basic, Logo, Prolog, Lisp, Forth or even 68000 assembly language. . . .

The one that we have decided to present in this final appendix is a surprisingly powerful interpreted language called *Microsoft Basic*. Admittedly, it can't produce quite the same effects as if you had a Lisa 2/5, and you worked in Lisa Pascal along the lines described in the preceding appendix. But almost. . . .

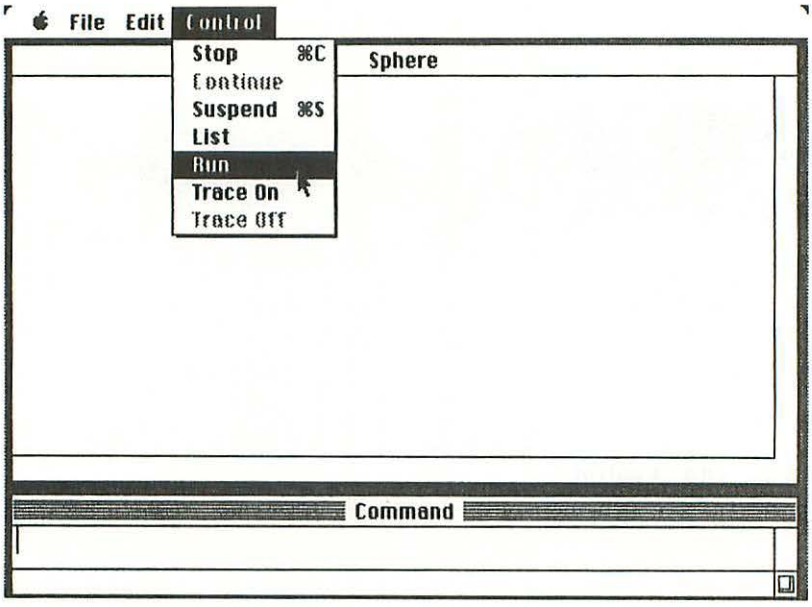
Demonstration

Let us start out with the excellent demonstration program on the *Microsoft Basic* startup disk, "MS Basic", whose directory is shown in the following illustration:

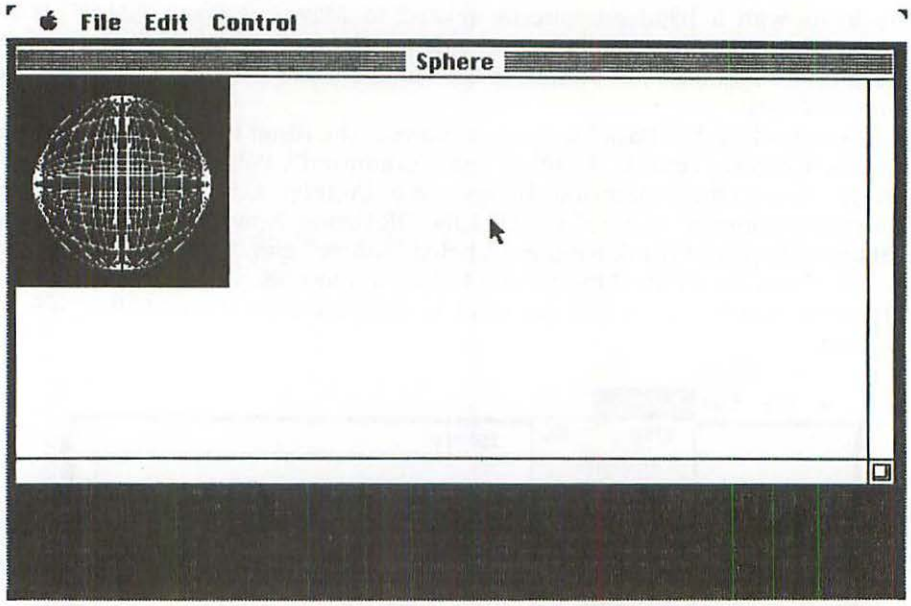


The icons with a hand are directly related to *Microsoft Basic*: "Basic" is the interpreter itself, whereas "Sphere" is the demonstration program that we are about to examine. "Modules", of course, is the familiar folder holding six elements of system software.

Double-click the "Basic" icon to get started. The result is an uninteresting pair of blank windows labeled "Untitled" and "Command". Pull down the File menu and choose the Open command. Things start to liven up: a dialog box asks us for a filename. We answer "Sphere" and click the **OK** button. Now we're greeted with an uninteresting pair of blank windows labeled "Sphere" and "Command". This time we pull down the **Control menu** (see following illustration) and choose the **Run** command, which means that we want to start the execution of the "Sphere" program.

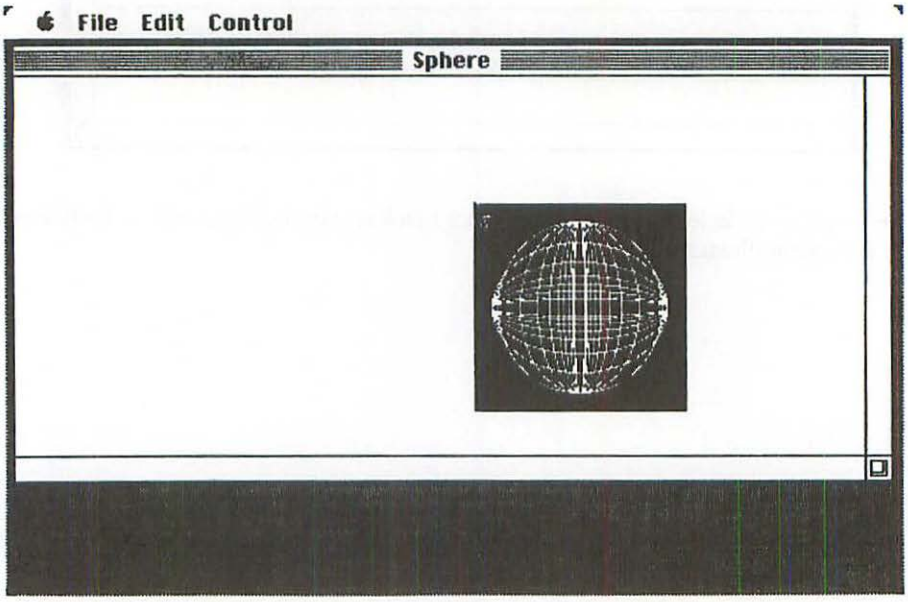


The result — a skeletal white sphere on a black square background — is shown in the following illustration:

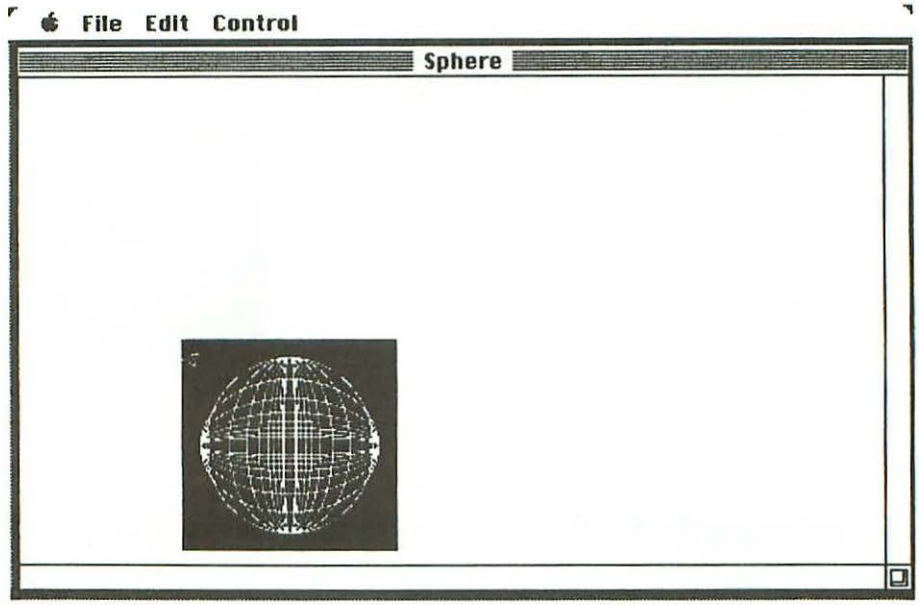


If we move the pointer to some other place on the "Sphere" window (see following illustration) and click the mouse button, the picture instantly reappears there. Alternatively, we can drag the picture around on the window.

Notice that the image is displayed only within the bounds of the *output window* entitled "Sphere". If you drag the image across the boundaries of this window, the part outside the window is no longer visible.

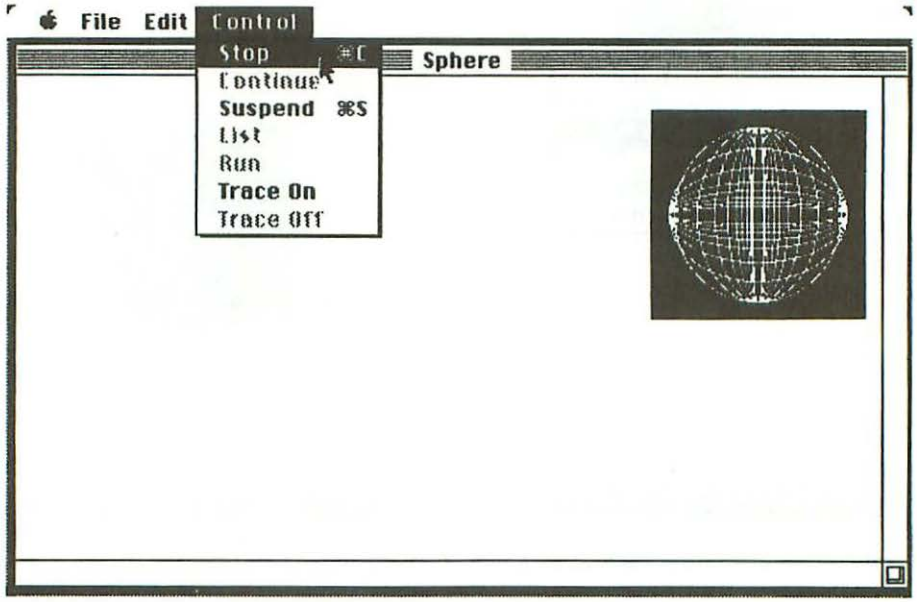


Since the "Sphere" window has a size box, we can enlarge the area of the window to cover the entire screen, as shown in the following illustration:

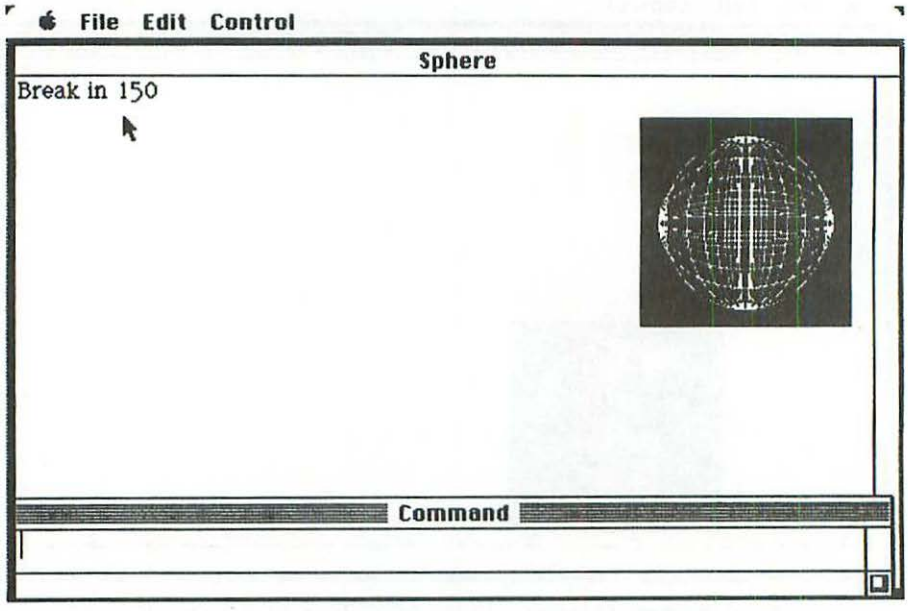


This demonstration program proves at least two things concerning *Microsoft Basic*. First, it is obvious that high-quality graphics can be obtained through this language. And second, it appears that the language can handle mouse input.

But let us interrupt the demonstration by pulling down the Control menu and choosing the **Stop** command, as shown in the following illustration:

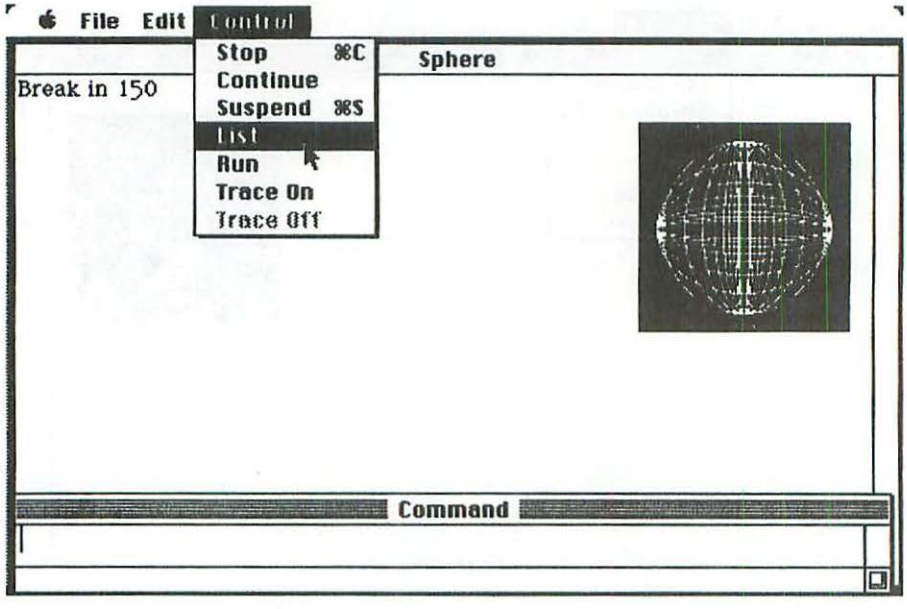


The program stops, as expected, and a message in the output window informs us (see following illustration) that a break has been introduced in the program at instruction no. 150.

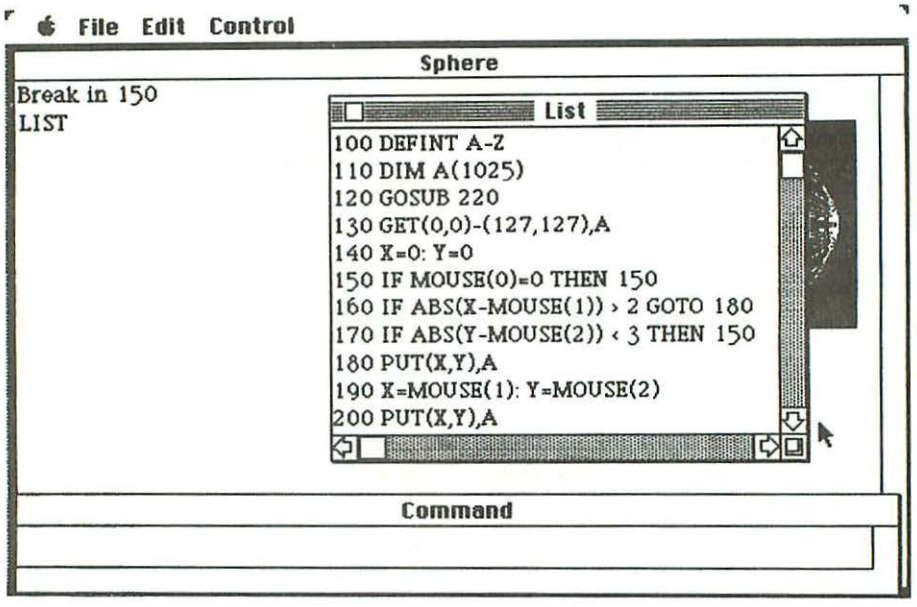


Notice that the empty window labeled "Command" has reappeared at the bottom of the screen, and that it is in fact the active window on the desktop.

It's time to take a look at the Basic program behind all that. Pull down the Control menu and choose the List command, as shown below:

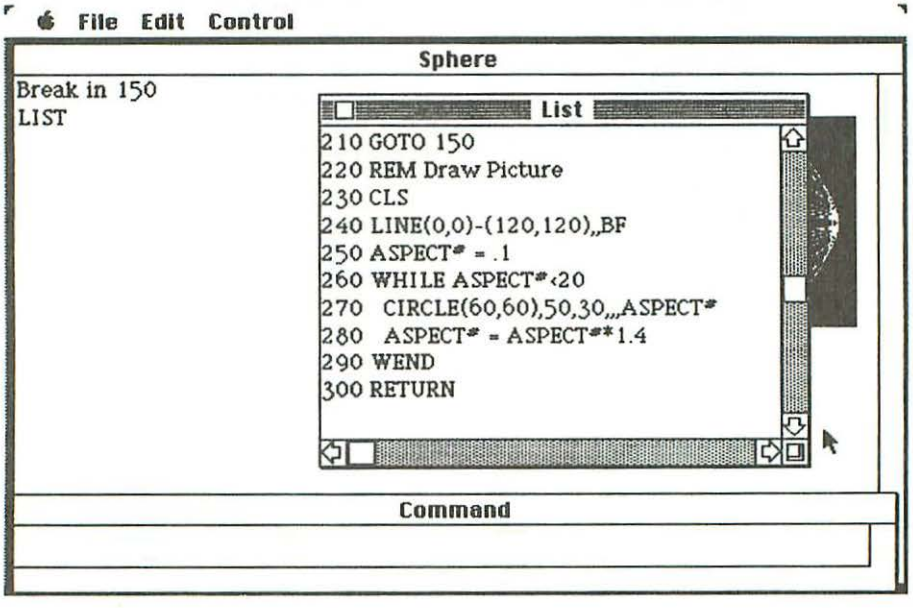


The following illustration shows the first eleven lines of the program, which appear in a window entitled "List":



Notice that the command "LIST" has been displayed in the output window.

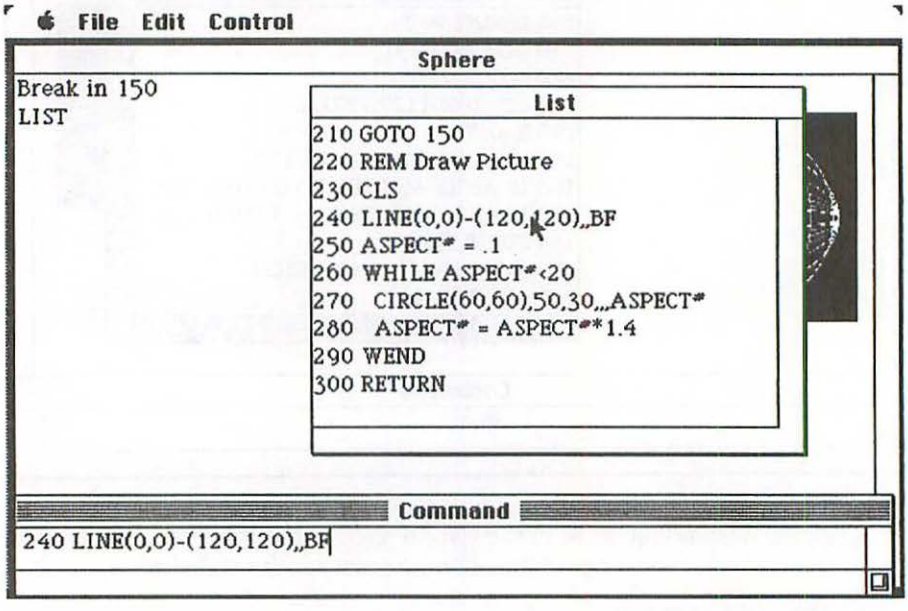
Click the descending arrow in the vertical scroll bar to move towards the end of the program. The final ten lines of this short program are shown below:



No doubt the most surprising aspect of this demonstration is the fact that a Basic program of 21 lines can do so much!

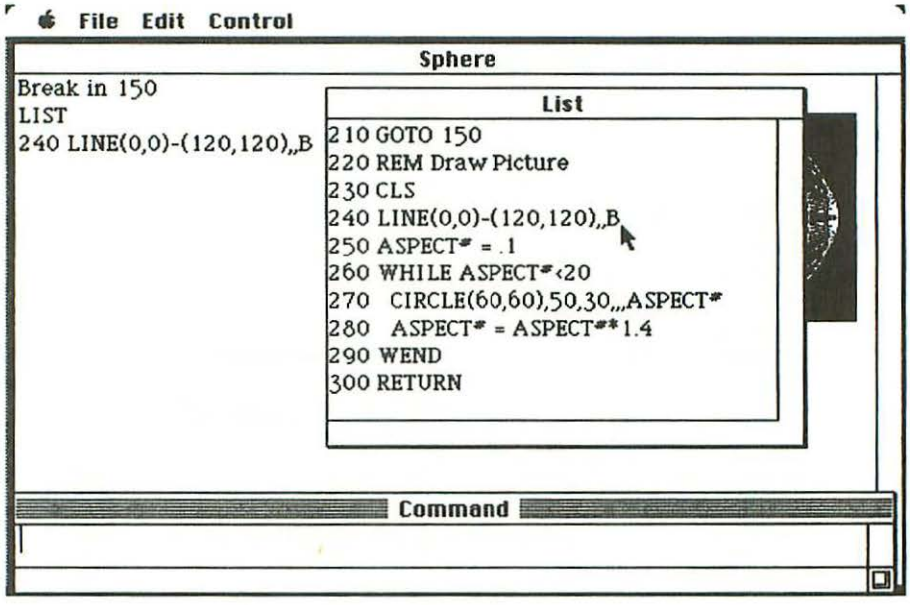
Let us now take a look at the procedure for modifying the program. In fact we shall change two statements in the program: lines 240 and 270.

Place the pointer anywhere on top of the statement in line 240 (see following illustration) and click the mouse button.

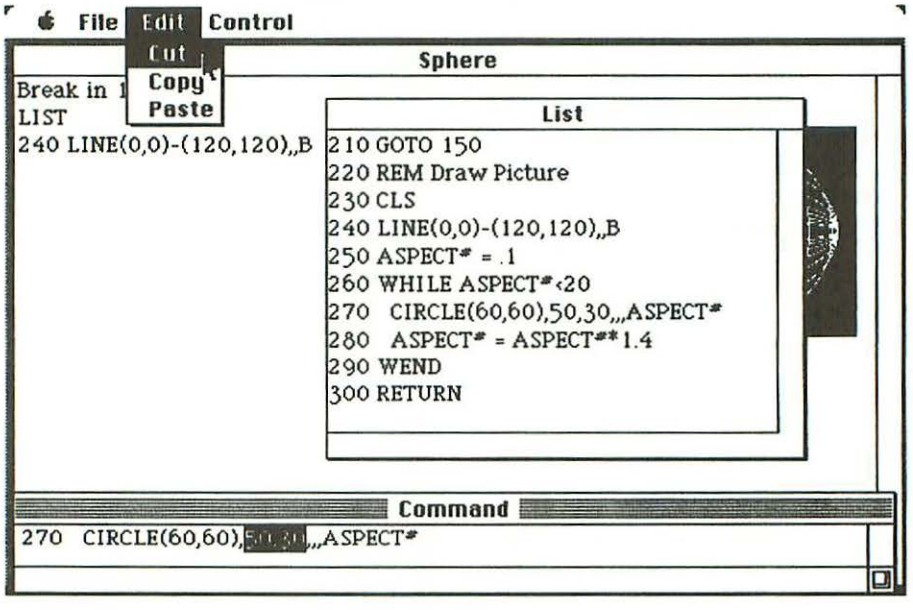


The selected line reappears instantly in the active Command window, with a blinking vertical bar just after the last character to indicate the insertion point.

The modification that we want to carry out on this line is very simple. Instead of the BF option of the Line statement (Box Fill, which places the image on a black background), we want to have B (Box, which gives rise to a white background). So, we hit <Backspace> once, and then <Return> to tell the machine that our editing operation is completed. Three things then happen simultaneously (see following illustration): the modified line is sent back to its context in the List window, it is also displayed in the output window, and the Command window is blanked.



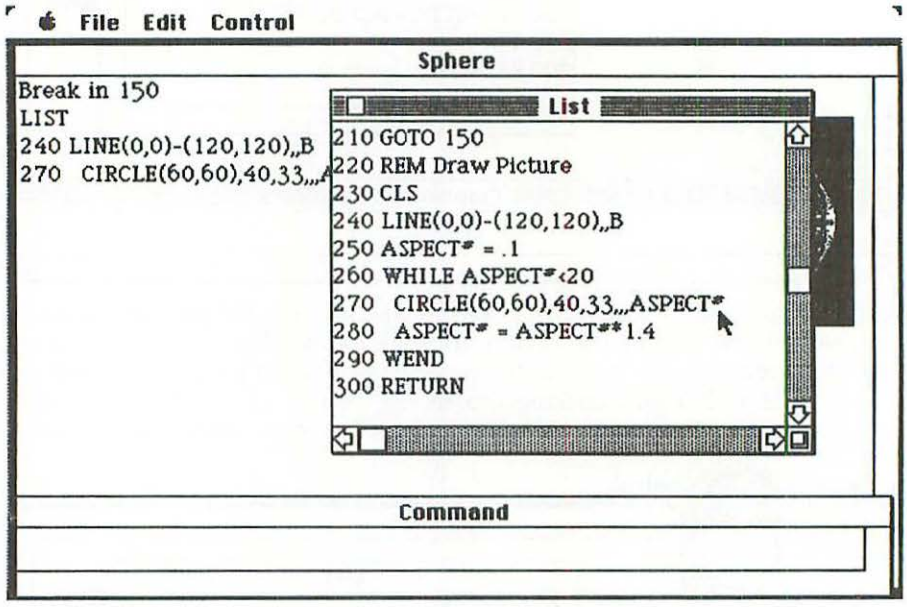
Next we want to modify line 270. First, you have to move the pointer onto the List window and click the mouse button, to make the window active. Then you can select the part of the program to be modified by placing the pointer somewhere on top of line 270 and clicking the mouse button. Line 270 is immediately displayed in the active Command window, as shown in the following illustration:



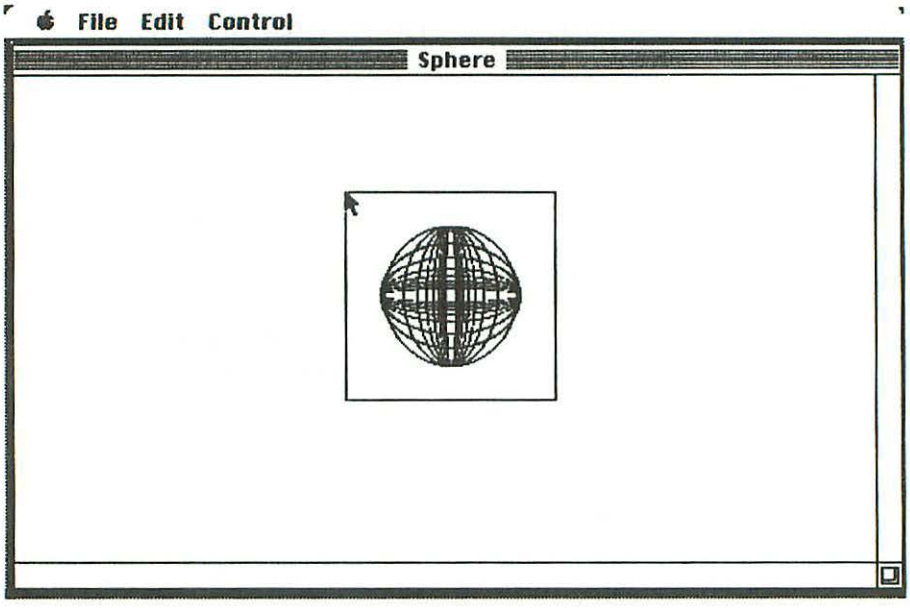
What we want to do this time is to replace the numbers 50,30 in the statement by 40,33. (The first number indicates the radius of the ellipse to be drawn. The second number represents the color of the drawn circles: 30 for white, 33 for black.)

First, you drag the pointer across the numbers 50,30 in the Command window, to select them. They are immediately highlighted in black. Then you pull down the Edit menu and choose the Cut command. Finally you type the numbers 40,33 and hit <**Return**> to let the machine know that the entire modification has been completed.

The desktop then appears as shown in the following illustration:



If you now pull down the Control menu and choose the Run command, the following result is obtained:



This demonstration is sufficient to point out the way in which *Microsoft Basic* works, and the interplay between the three windows. Let us now take a brief look at the language itself.

Line-by-line description of the demonstration program

100 Every variable whose name starts with a letter from A to Z is considered to be of integer type.

110 The A variable is a one-dimensional array with a maximum of 1026 elements. (The low value of the subscript is zero.)

120 Execute the subroutine that starts at line 220.

130 Screen GET statement: a special feature of the Macintosh version of Microsoft Basic. The A array receives the set of bits defining the square image whose upper left corner is located at the position (0,0) of the window, and whose lower right corner is located at (127,127). To stock all the bits between the positions (X1,Y1) and (X2,Y2), the number of bytes required is given by the following formula:

$$4 + (Y2 - Y1 + 1) * 2 * INT((X2 - X1 + 16) / 16)$$

where the INT function returns the largest integer that is equal to or less than the argument of the function. In the present case, the formula gives a result of 2052. Since each element of an integer array can stock two bytes, the A array must therefore be composed of 1026 elements.

140 Initialize the X and Y integers.

150 Repeat this line nonstop until the user presses the mouse button.

160 If the mouse has caused the pointer to be moved at least three points in a horizontal direction, then jump to line 180.

170 If the mouse has caused the pointer to be moved at least three points in a vertical direction, then proceed to line 180; otherwise, return to line 150.

180 Screen PUT statement: a special feature of the Macintosh version of Microsoft Basic. Erase the image at its current location.

190 Update the values of the X and Y coordinates.

200 Screen PUT statement: a special feature of the Macintosh version of Microsoft Basic. The A array gives rise to an image whose upper left corner is located at the spot on the screen specified by the X and Y coordinates.

210 Return to line 150.

220 Remark (REM): Draw Picture. First line of the subroutine.

230 Clear the complete window.

240 LINE statement: a special feature of the Macintosh version of Microsoft Basic. Display a square between the points (0,0) and (120,120). The BF option means that the square will be filled with black.

250 Because of the final # character, ASPECT# is recognized as the name of a double-precision numeric variable. This variable is set to 0.1.

260 Start of a loop. As long as the variable ASPECT# remains less than 20, repeat lines 270 and 280.

270 CIRCLE statement: a special feature of the Macintosh version of Microsoft Basic. Draw a white ellipse (parameter 30), with a radius of 50, whose center is located at the position (60,60). The ASPECT# parameter indicates the relationship — referred to as the *aspect ratio* — between the height and the width of the ellipse. As long as this ratio remains less than 1, it's the horizontal radius that is indicated in the statement; if the ratio is greater than 1, then it's the vertical radius that is indicated.

280 Increase the aspect ratio.

290 End of the WHILE loop that started at line 260.

300 End of the subroutine that started at line 220.

Note: Readers of this description of the demonstration program will have understood that the image that we have been calling a "sphere" is in fact a set of 16 ellipses, 7 of which are horizontal (like an egg on a table), while the 9 others are vertical (like an egg in an eggcup).

Super Basic

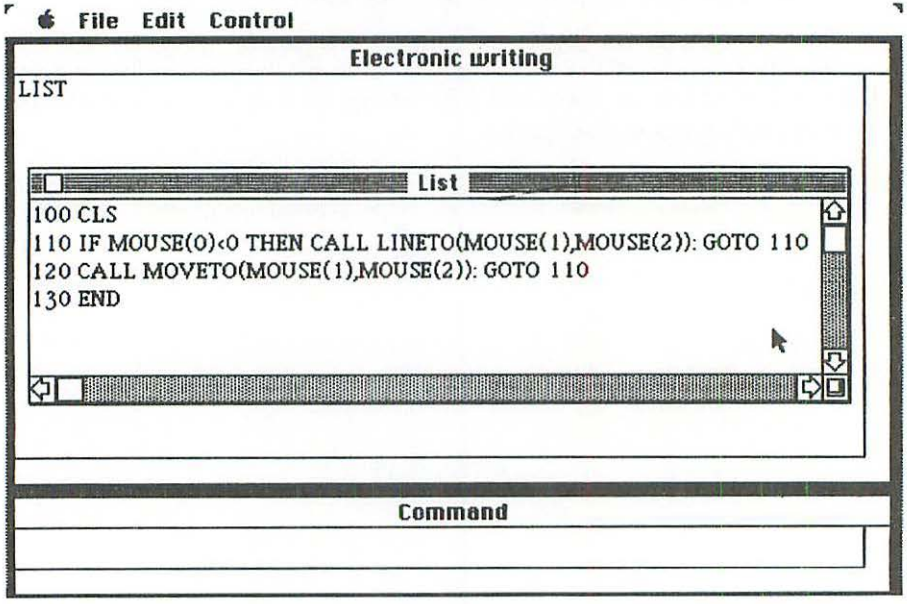
If we compare this latest version of the language with early generations, it's almost true to say that the principal thing in common seems to be the name "Basic"!

Microsoft Basic is a collection of about 140 statements, functions and commands. The Macintosh version is a "standard" language in the sense that the interpreter will run programs written in Microsoft Basic on other computers, but it naturally includes certain features that do not exist outside Macintosh.

There are two novel aspects of the language, above all, which deserve to be explained by way of an example: first, the *Mouse* function, and second, the possibility of explicitly calling 41 of the 145 *QuickDraw* routines in the Macintosh ROM.

The function Mouse(n) has seven distinct meanings, depending on the value of n. For example, Mouse(0) returns a value that indicates the status of the mouse button, whereas Mouse(1) and Mouse(2) return respectively the current X and Y coordinates of the pointer. The other possibilities — Mouse(3) through to Mouse(6) — return the starting and ending coordinates of drag operations.

Look at the following example:

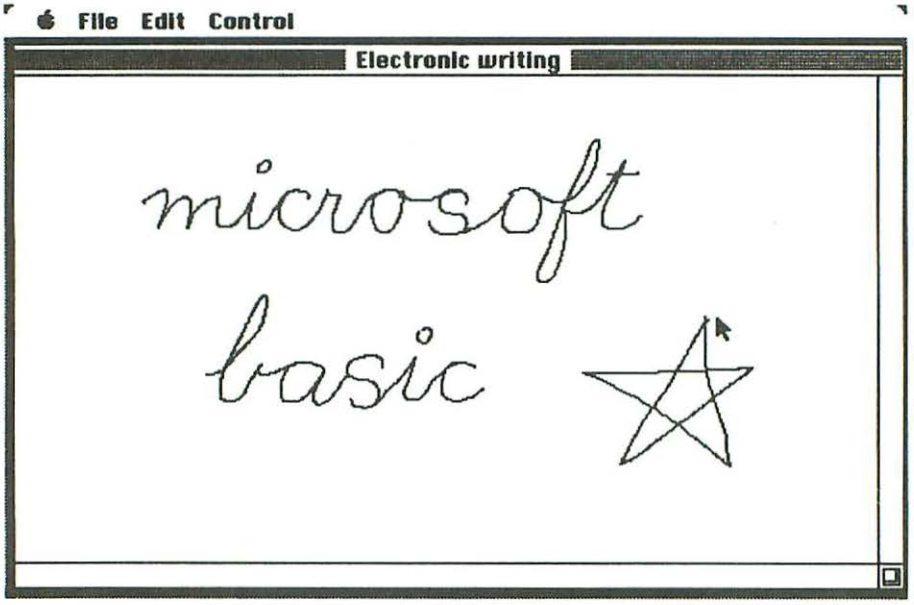


The statement in line 100 (Clear Screen) erases the contents of the output window and sets the pen position to the upper left corner of this window.

In the next line, the fact that Mouse(0) is negative indicates that the button is still being held down. In that case, the QuickDraw procedure LineTo is invoked, in order to carry on drawing.

As soon as the mouse button is no longer being held down, the program jumps to line 120, where the QuickDraw procedure MoveTo is invoked, to keep track of the position of the mouse.

The outcome (see following illustration) may not be fine art, in the truest sense of the term . . . but it certainly proves that *Microsoft Basic* provides a simple solution for getting your Macintosh to behave in a most interesting manner.



The conclusion seems to be that the development of marketable Macintosh software will no doubt be carried out exclusively by means of the Lisa 2/5 approach described in the preceding appendix, whereas personal tasks that concern only the individual Macintosh owner can probably be handled in a quite satisfactory manner by means of a powerful and easy-to-use language such as *Microsoft Basic.*

INDEX

PRENTICE
HALL
INTERNATIONAL
PERSONAL
COMPUTER
BOOK

A User's Guide to Apple's
Macintosh Computer

Contents include:

The Macintosh mouse
The Macintosh window
Cut and Paste: information transfer
Housekeeping: file management
'MacWrite': word processing
'MacPaint': image processing
'MacDraw': graph processing
'Multiplan': electronic worksheet
'Chart': business graphics
'MacProject': project planning

This book gives you a full
operational knowledge of
the Macintosh computer.

0-13-559527-4