MICRO COMPUTER GRAPHICS

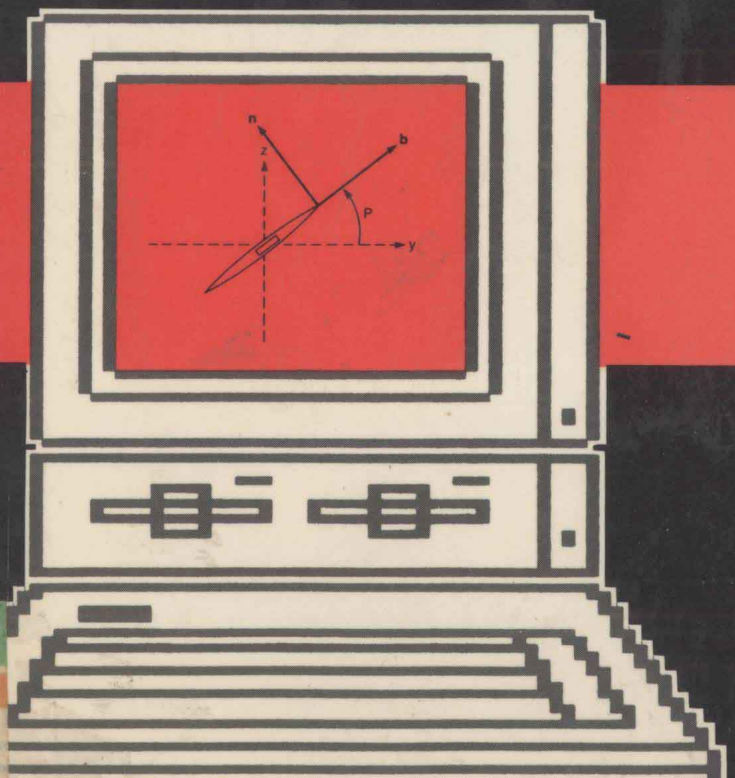
USING

PASCAL

for the

APPLE II Family

Richard Halpern



Microcomputer Graphics Using Pascal

For the Apple II Family

RICHARD HALPERN
Bergen Community College

HARPER & ROW, PUBLISHERS, New York
Cambridge, Philadelphia, San Francisco,
London, Mexico City, Sao Paulo, Singapore, Sydney

Sponsoring Editor: John Willig
Project Coordination: Total Concept Associates
Cover Design: Sullivan Studios
Text Art: Textbook Art Associates
Production: Willie Lane
Compositor: Ampersand Publisher Services, Inc.
Printer and Binder: R.R. Donnelley & Sons, Company

Microcomputer Graphics Using Pascal: For the Apple II Family

Library of Congress Cataloging in Publication Data

Halpern, Richard P.
Microcomputer graphics using PASCAL for the Apple II family.

Bibliography: p.
Includes index.
1. Apple II (Computer)—Programming. 2. PASCAL (Computer program language) 3. Computer graphics.
I. Title.
QA76.8.A662H274 1985 001.64'43 85-758
ISBN 0-06-042583-0

85 86 87 88 9 8 7 6 5 4 3 2 1

Microcomputer Graphics Using Pascal

For the
Apple II Family

Preface

Welcome to the world of computer graphics! If you know little about the subject and wish to learn more, I think you will find this book useful. In this introduction, we will briefly discuss what this book is about and what you should know to make good use of it.

What Is Computer Graphics Good for, Anyway?

The simplest answer to this question is that it's good for quite a bit. Computer graphics is used in many different fields. It is a phenomenon that is bound to increase as the price of good graphics equipment continues to drop. The following are representative examples of computer graphics applications:

Business. *Bargraphs* and *pie charts* are often used in business, usually to summarize large quantities of data that would otherwise be time-consuming and painful to read.

Science/Research. Various plots, such as *line graphs*, *scatter diagrams*, and *histograms*, are used to summarize patterns and tendencies found in research results. *Contour plots* are ideal candidates for computers: not only can the computer plot the desired quantity, but it can also do the complex calculations often associated with these plots. *Simulations* of physical phenomena, such as fluid flow, can be carried out and observed by means of computer graphics.

Engineering/Design. A wide variety of projects, from large power plants to delicate optical systems, are designed with the aid of computer-generated drawings. Not only can something be designed, but its performance can be simulated using the computer. If need be, the design can be modified and redrawn, and this procedure can be repeated as often as necessary.

Image Processing. Computers can be programmed to analyze the information content and enhance the quality of certain images. The most widely publicized examples of this are pictures sent back from space probes . The computer can remove spurious information that is inevitably present in long-range transmissions and improve contrast. In a more down-to-earth application, computer-assisted tomography—CAT scan—allows a physician to assemble a three-dimensional model derived from X rays taken from various perspectives.

Computer-Generated Movies. The slick commercials on television that are characterized by space-age effects are mostly computer-generated. So are such films as *Tron*.

Games. Computer games are examples of clever graphics programming. At first available only on arcade machines, they can now be played on very inexpensive home machines. This is a good example of how plummeting prices make graphics-related equipment available to a wider audience.

What Can You Expect from This Book?

The purpose of this book is to explain the *principles* behind computer graphics. The chapter summary that follows will give you a rough idea of what lies ahead. Note that the book is *not* a compendium of graphics programs. Although numerous ideas are illustrated with Pascal routines, it is up to you to do the vast bulk of the programming .The book will give you the foundation and the framework; you must build the rest of the house .

Chapter 1 is a discussion of hardware. A knowledge of how a system puts an image on the screen is very useful, especially if you want to exploit the features of the system through advanced programming. The chapter doesn't discuss specific brands of hardware. Rather, it answers general questions: How does a CRT project an image on its screen ? How can a computer store an image? What are the major input and output devices , and how do they work ?

Chapter 2 introduces the essential core of graphics commands that are needed to apply the principles covered in the book.

Chapter 3 presents two types of graphs that provide points of departure for the discussion of two important problems. The line graph introduces a

discussion of how to handle the wide variety of coordinate systems that exist in the real world. The bar graph is used to introduce the general problem of drawing polygons. This chapter also discusses the problem of filling polygons—that is, lighting all the interior pixels.

Chapter 4 discusses two-dimensional transformations. These are the mathematical operations that allow us to execute certain changes on an object, such as enlargement or various types of movement.

The focus of chapter 5 is the organization of graphics data: How do we arrange graphics information in the memory so that we can use it to make a picture? How do we store a picture on a disk and retrieve it from the disk?

Chapter 6 discusses the concepts of windows and viewports. You will learn how to restrict your attention to a particular area of a picture and how to limit your drawing to a certain part of the screen.

Chapter 7 begins the treatment of three-dimensional graphics. After some mathematical preliminaries, which extend familiar two-dimensional concepts to three dimensions, the chapter discusses projections, which enable us to represent real-world objects, which are in three dimensions, on a two-dimensional screen.

Chapter 8 continues the discussion of projections, extending your capabilities to include viewing an object from various positions.

Chapter 9, the last chapter, is devoted to the task of hidden line removal. This is important because only the parts of an object that we can see should be projected on the screen.

Finally, appendix A provides a review of mathematics ,and appendix B presents additional p-System Turtlegraphics commands.

What Does This Book Expect from You?

Needless to say, to do computer graphics, you need a graphics-capable computer. This book assumes that you have an Apple and the p-System and that you can program in Pascal. I have been careful throughout the book to build the routines on a small group of basic commands: this allows you to concentrate on the graphics, not on complete mastery of the p-System. Readers who wish to explore the entire Turtlegraphics system should refer to appendix B.

Most areas of computer graphics require some mathematics. At the very least, you should feel comfortable with basic algebra and trigonometry. In addition, you should either understand or be able to learn about vectors and matrices. If you have any doubts in these areas, you should read appendix A first.

The surest way to become proficient in computer graphics is to *do* computer graphics. A book can teach you the principles, but it cannot convey

the many details and subtleties involved in developing a useful graphics project. At the very least, try the exercises at the end of each chapter. They provide a good review of what you have learned (or think you have learned). Better yet, think of some projects that are interesting to *you*. Such projects tend to be the most motivating, and they will often stimulate you to probe beyond what you have already learned. Let your imagination go free—and have fun!

Richard Halpern

Contents

Chapter 1

Hardware Fundamentals

When you have finished this chapter, you will understand how a microcomputer can produce an image from dots on a cathode ray tube (CRT). You also will have been introduced to various useful computer graphics hardware devices.

OBJECTIVES

1. To understand the essentials of microcomputer operation.
2. To understand the operation of raster scan and vector scan displays and DVST systems.
3. To understand the operation of hard-copy output devices.
4. To understand the operation of various graphics input devices.

BASIC COMPUTER OPERATION

Although a microcomputer may appear to be a jungle of integrated circuits, we can think of it as composed of three distinct units: a **memory unit**, a **central processing unit (CPU)**, and an **input/output (I/O) unit**. Basically, the memory stores information, the CPU does the calculations, and the I/O

unit acts as the interface between computer and human. We will elaborate on each of these units shortly. For electronic communication, the units are linked together by many parallel wires called a **bus**. A schematic diagram of a computer system is shown in figure 1.1.

How is information in a computer represented? If we could investigate the voltage at every point along the computer's lines of communication, we would turn up only two possible values: 0 and a fixed value, such as 5 volts. This situation is represented most naturally by the binary number system, whose values, called **bits**, are 0 and 1. The 0 represents the absence of a voltage, and the 1 represents the presence of a voltage. Therefore, all information in a digital computer can be represented as a series of 1's and 0's.

Computers handle bits in groups called **words**. A word usually has 32 bits for large computers, 8 or 16 bits for smaller computers. It is important to note that two words that have the same combination of 1's and 0's do not necessarily have the same meaning. Computers interpret words differently in different situations. Consider the word 0 1 1 0 1 0 0 1. At times, it is interpreted as the binary number 105. At other times, it is interpreted as the character *i*. Fortunately, computers are designed to provide the correct interpretation of words at all times.

Memory

The function of a computer's memory is simply to store information. The memory unit consists of thousands of tiny electronic cells, called **locations**,

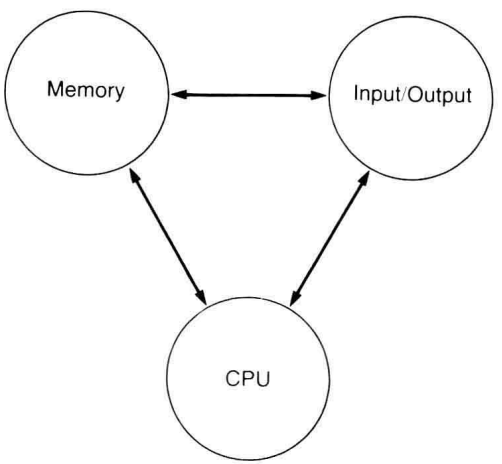


Figure 1.1 Schematic diagram of a computer. Arrows symbolize the bus, or wires, connecting the units.

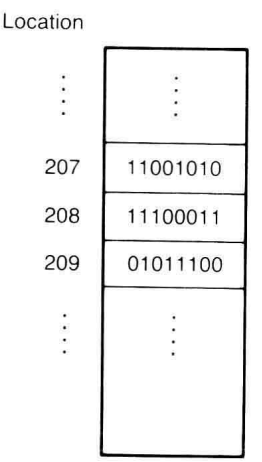


Figure 1.2 Memory locations and their contents.

each capable of storing eight bits, or one **byte**, of information. Every location is identified by a unique number, called an **address**, that indicates physical location within the memory. Figure 1.2 is a schematic diagram of a section of memory. Stored information, referred to as the **contents** of a location, falls into one of two categories: **instructions**, which tell the computer what to do and are normally stored in consecutive locations; and **data**, which consist of information that is processed in some way according to the instructions.

Two types of operations can be performed on a memory location: a **write** operation, during which the contents of a location are replaced by a different byte (figure 1.3); and a **read** operation, during which the contents of a location are copied, but not changed, by other units of the computer (figure 1.4). In either case, the memory unit must be sent signals indicating the address of the desired location and the nature of the operation (read or write). The memory is equipped with special circuitry that decodes these signals and responds accordingly. Note that all locations of the memory can be addressed with equal facility; this capacity is known as **random access**.

Memory that can be used for both read and write operations is called **random access memory (RAM)**, but it should be thought of as read-and-write memory, since, as noted, all machine memory is random access. When the power is turned off, the contents of RAM are lost. Memory with instructions permanently in place is called **read only memory (ROM)**. One cannot write to or alter ROM. Note that ROM retains its contents even if the power goes off. It is usually desirable to keep certain information permanently in memory. For example, a particular set of instructions is needed by the

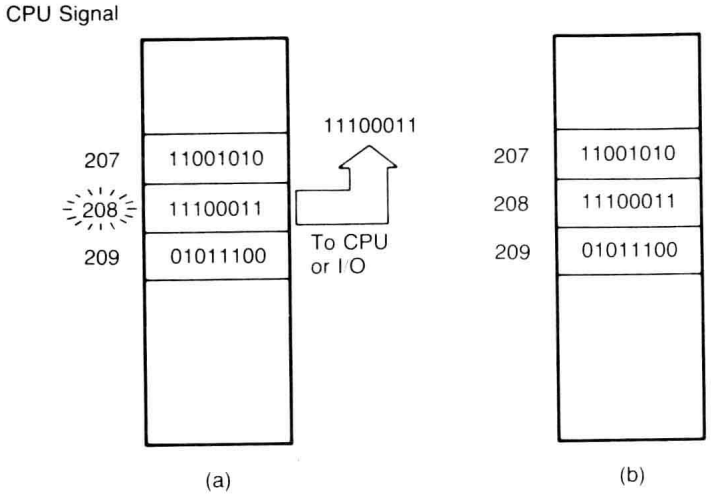


Figure 1.3 The write operation. (a) Just before writing to location 208. (b) After writing; note new contents of location 208.

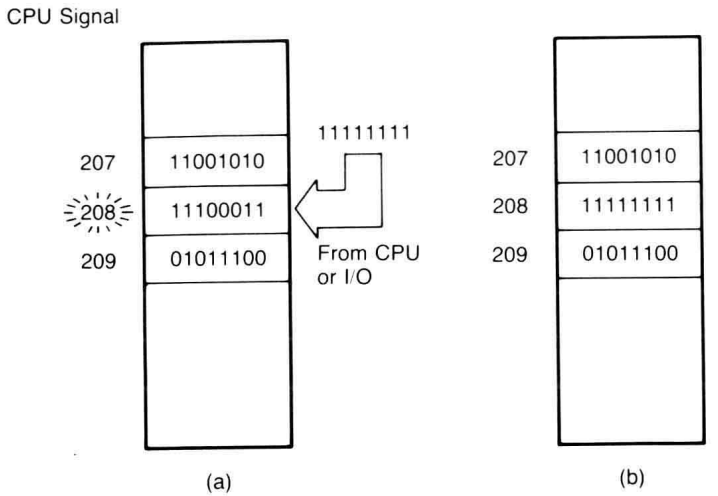


Figure 1.4 The read operation. (a) Reading location 208. (b) Memory after reading: location 208 still has its original contents.

computer to decode a keypress. Manufacturers make these instructions a permanent part of the computer so that each time a key is pressed, the instructions in this permanent memory are carried out and the keypress is decoded.

Central Processing Unit (CPU)

The CPU is the "brain" of the computer. It performs various arithmetic, logic control, and decision-making functions. Basically, the CPU divides its time between two **modes**, or phases, of operation. In the **fetch** phase, the CPU reads a byte from memory and then interprets it as either an instruction or data. In the **execute** phase, it carries out the instruction, or, if the fetched byte is data, processes it according to a previous instruction.

The CPU does its job with the aid of several special-purpose memory locations called **registers**, which are located in the CPU itself. The **instruction register** holds an instruction until it is decoded by a logic circuit. Another register, the **accumulator**, is used to carry out operations on two quantities. For example, suppose that the computer has to add 3 and 7. First, the 3 is fetched and placed in the accumulator. Next, the 7 is fetched and placed at one input of a CPU circuit called the **arithmetic-logic unit (ALU)**. Then the 3, which is in the accumulator, is placed at a second input of the ALU. The ALU performs the operation and stores a 10 back in the accumulator. The CPU can then use the 10 for another operation or store it in the memory.

How does the CPU step through a list of instructions? It is assisted by another register, the **program counter**, which always contains the address of the next byte to be fetched by the CPU. In the fetch phase, the CPU signals the memory for the address given by the contents of the program counter. Then the program counter is incremented by one. This crucial step ensures that when the next fetch takes place, the CPU gets the next address on the list. If the program counter is initialized to the address of the first instruction, and the last instruction is an `END` command, the CPU automatically executes all the instructions, and no more.

In graphics systems, specially designed processors have registers that are devoted solely to graphics functions. For example, by simply changing the contents of a color-mapping register, a programmer can rapidly change the colors of large areas of the screen. (After studying the raster scan system later in this chapter, you will appreciate the power of this capacity.)

Input/Output (I/O) Unit

Computers would not be useful if they were not able to communicate with the outside world. Such communication is the function of the I/O unit. There are several ways to manage I/O, but only one is of particular interest for computer graphics: **memory-mapped I/O**. The concept of memory mapping is quite simple: each input or output device is connected to a unique memory location, and the CPU reads from or writes to that location. For example, suppose that a keyboard is connected to location 10. When a key is pressed, its bit code is placed in location 10, and the CPU can determine what key was pressed by reading from location 10. Similarly, suppose that a video display is connected to location 20, using circuitry that flashes the contents of that location on the screen. To produce a byte of visible output, the CPU simply writes to location 20. Thus, input and output operations appear to the CPU as ordinary data transfers.

A given I/O device can be connected to more than one memory location. For example, a video display that can show twenty-five lines of eighty characters each represent 2,000 unique areas on the screen, each of which can be assigned its own memory location. Then a character can be made to appear in any screen area by writing its code to the corresponding memory location (see figure 1.5).

OUTPUT DEVICES

For our purposes, the most important output device is the video display. This is basically a television set that can display characters and graphics with great clarity. The heart of a video display is the **cathode ray tube (CRT)**.

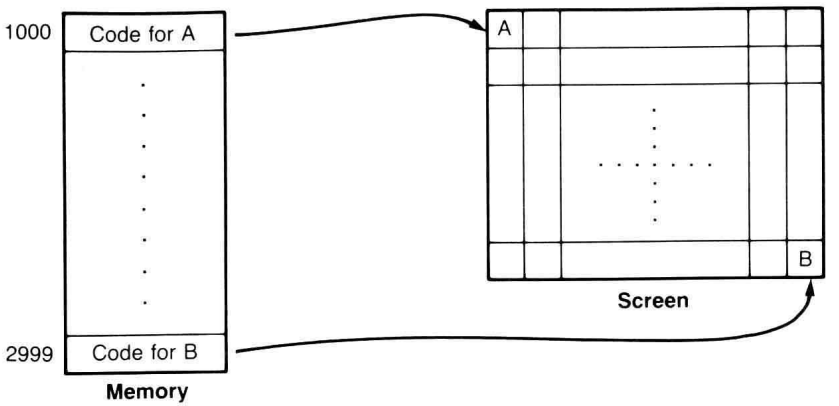


Figure 1.5 Hypothetical memory-mapped video display. The first byte of a 2,000-byte section maps to the upper left; the last byte maps to the lower right.

Basically, a CRT produces electrons at one end and accelerates them toward a phosphor-coated screen at the other end. The phosphor glows wherever electrons strike it.

Figure 1.6 shows the various elements of a CRT. A component called a **cathode** emits electrons when it is heated. Another component, the **anode**, is maintained at a high positive voltage. The anode attracts the electrons and causes them to accelerate rapidly toward the screen. A third component, called a **focusing device**, concentrates the electrons into a beam so that by the time they reach the screen, they have converged to a tiny, dotlike area.

An image is created by causing dots to light at selected points on the

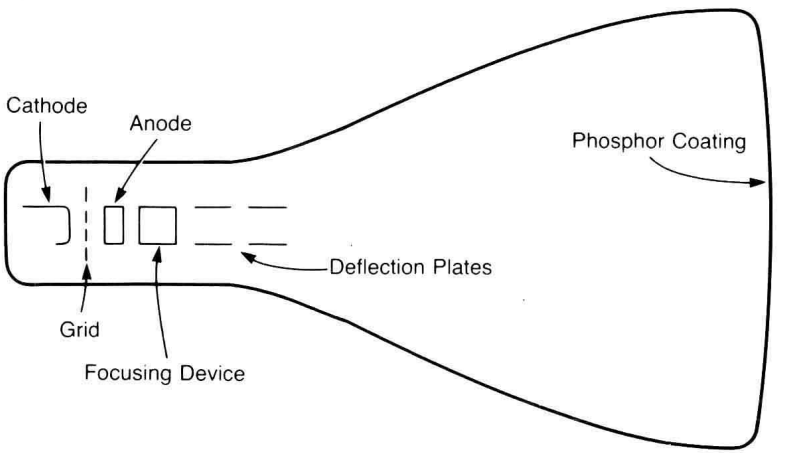


Figure 1.6 A cathode ray tube (CRT).