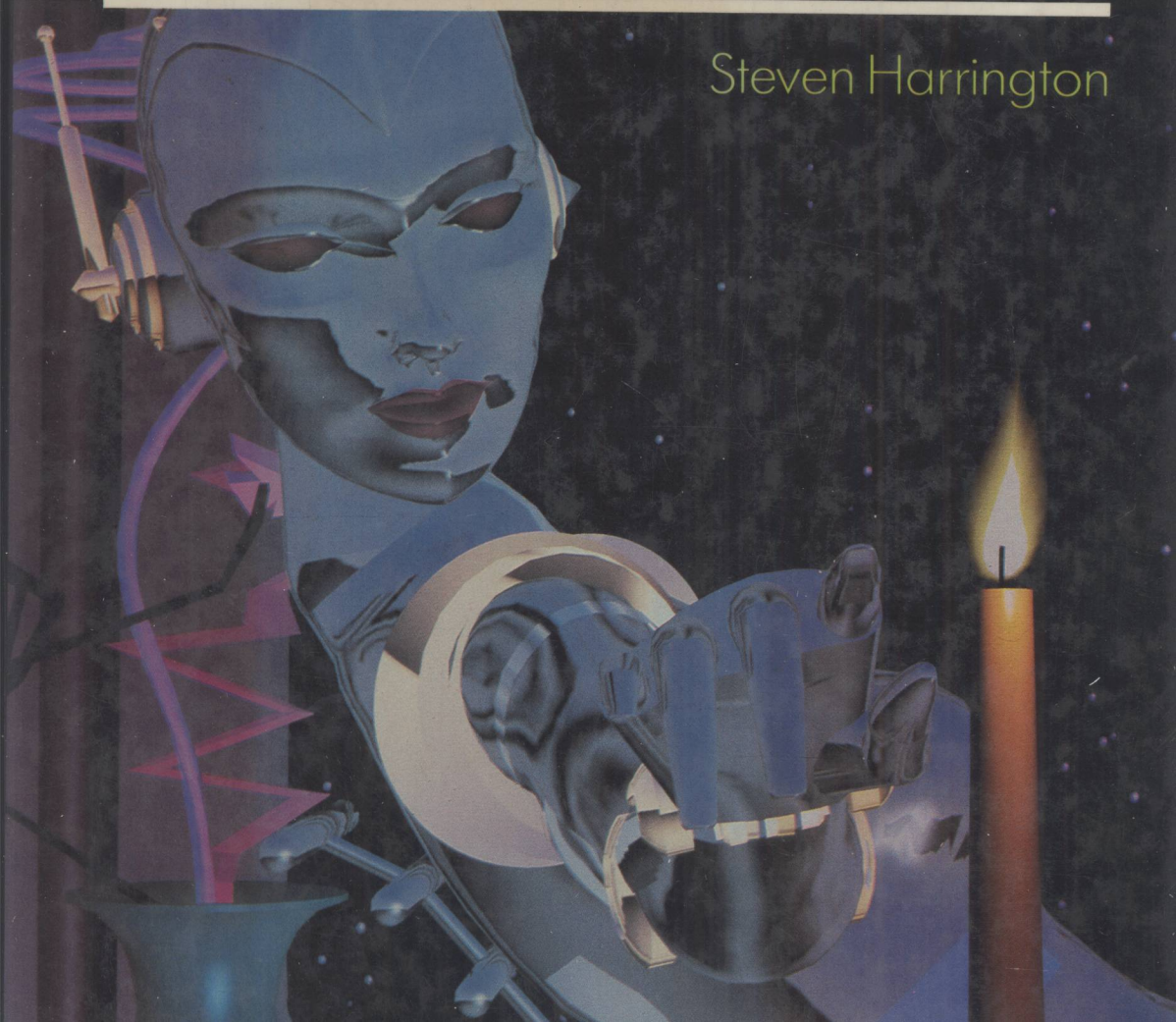
COMPUTER G·R·A·P·H·I·C·S

A PROGRAMMING APPROACH

Steven Harrington



COMPUTER GRAPHICS

A Programming Approach

Second Edition

Steven Harrington

Xerox Corporation

McGraw-Hill Book Company

New York St. Louis San Francisco Auckland Bogotá Hamburg
London Madrid Mexico Milan Montreal New Delhi Panama
Paris São Paulo Singapore Sydney Tokyo Toronto

This book was set in Times Roman by Publication Services.
The editors were Kaye Pace and Larry Goldberg;
the cover was designed by Amy Becker;
the production supervisor was Salvador Gonzales;
new drawings were done by Wellington Studios, Ltd.
R. R. Donnelley & Sons Company was printer and binder.

The cover photo was supplied by Abel Image Research, Los Angeles, California. This still image is from a 30-second commercial entitled *Brilliance*. It was produced by Robert Abel & Associates of Los Angeles for the canned food information council in collaboration with Ketchum Advertising of San Francisco.

COMPUTER GRAPHICS
A Programming Approach

3 4 5 6 7 8 9 0 DOCDOC 8 9 2 1 0 9 8

ISBN 0-07-026753-7

Library of Congress Cataloging-in-Publication Data

Harrington, Steven.
Computer graphics.

Includes bibliographies and index.
1. Computer graphics. 2. Programming (Electronic computers) I. Title.
T385.H34 1987 006.6 86–21295
ISBN 0–07–026753–7

PREFACE

HANDS-ON APPROACH

This book, the second edition of an introductory text on interactive computer graphics, presents the basic concepts of that field. The approach is somewhat different from that of previous texts in that it encourages a "hands-on," or "learn-by-doing," attitude. The book provides guidance for developing a graphics system, suggestions for modifications and extensions of the system, and application problems which make use of the system. The advantages of this approach are a high level of student involvement, an understanding of the systems aspects of graphics software, a realistic feeling for graphics system capabilities and ease of use, and a greater feeling of accomplishment.

PREREQUISITES

This book is written at a lower level than previous graphics texts. It is intended for students with a knowledge of high school algebra, analytic geometry, and trigonometry, and with at least one high-level language programming course. Although the necessary information on vectors, matrices, and data structures is presented in the text, students familiar with these topics will find the material easier to follow. The text should be suitable for junior college and undergraduate college levels. Most of computer graphics is based on the mathematics of analytic geometry; a knowledge of this area is appropriate and necessary. This book presents a graphics system as a series of programming problems, so a familiarity with some high-level language is required. The text attempts to present its algorithms in a language-independent fashion and does not require the knowledge of any one particular language. It has been used successfully with Pascal, C, and Fortran.

TIME REQUIREMENTS

The material can be covered at a rate of about one chapter per week. Slightly more time is needed to cover Chapters 8 to 11. Each chapter also provides a programming assignment for a component of the graphics system. The requirement of a weekly pro-

gram is a heavy but not excessive load for students. If necessary, the less instructive portions of programming projects may be provided by the instructor. At the end of the course, students have not only a working knowledge of computer graphics fundamentals but also a graphics system, or a collection of graphics programs, of which they can rightly be proud. There are several suggested programming problems at the end of each chapter. Some of these problems are much more difficult than others. The difficult problems are marked with an asterisk; very difficult problems have two asterisks.

ORGANIZATION

The text is organized so that each chapter introduces a new topic in computer graphics. The progression is such that new chapters build on previous ones, so the order of presentation is rather fixed. The first chapter begins with a very elementary review of analytic geometry and a discussion of vector generation. The second chapter considers line- and character-drawing commands and provides the interface between the specific display devices being used and the rest of the system. It also presents the basic concept of the display file, or metafile. The third chapter presents polygon surfaces and the rasterization problem. The fourth chapter discusses the basic transformations of translation, scaling, and rotation. The fifth talks about display file segmentation and visibility. These first five chapters, dealing with the management and interpretation of the display file, form a natural group when considering intelligent graphics terminals which maintain their own display files. The sixth chapter completes the discussion of two-dimensional graphics with a discussion of windowing and clipping. The instructor may choose to cover Chapter 6 before Chapter 5. This groups the image construction topics together, followed by the data structuring topic. In the seventh chapter, interactive techniques are discussed. Here again, routines act as an interface between the rest of the system and the particular graphics input hardware available. In Chapter 8 the system is generalized to three-dimensional graphics. The topics of three-dimensional geometry, viewing transformations, parallel projection, perspective projection, and three-dimensional clipping are discussed. Chapter 9 considers hidden surface and line removal. It presents two methods in detail: The first is the simple check for removal of back faces; the second is a priority sort for implementation of the painter's algorithm. The tenth chapter considers shading and color. The eleventh and final chapter considers the drawing of curved lines and surfaces. Arc generations, interpolations, and fractals are covered.

Throughout the text, the generic masculine pronoun "he" has been used solely on account of the ease of expression it affords. Such use of the masculine pronoun should not be interpreted as a wish to exclude women from the use of this text or from the field in general; in fact, all areas of computer science provide excellent career opportunities for women. "The programmer...he" is used only because it is less cumbersome than "the programmer...he or she."

RELATION TO STANDARDS

The original edition of this text was modeled after the GSPC CORE system, with extensions to raster graphics, hidden surfaces, shading, and curves. Although the new

edition has modified and extended the system, the basic object-oriented structure and basic input and output primitives have been preserved. Objects are modeled in world-coordinate space, and views of the objects are transformed to normalized device coordinates. Individual graphics primitives can be grouped into segments. This approach is common to many of the current standards, so the student will feel at home with the GKS, PHIGS, CGI, and CGM standards.

SIMPLICITY VS. EFFICIENCY

There are many places in this book where several different algorithms are possible, where the problem may be solved many different ways. As a rule, only one solution is presented in great detail, although others may be described. The particular solution chosen may not be the most clever or the most efficient. The primary objective was not to create the best graphics system but to teach the basic graphics principles. The algorithms selected are, therefore, those which were felt to most readily convey these principles and which were easiest to understand.

APPROACHES TO A GRAPHICS COURSE

There are several ways of using this book. One method is to have students implement the described graphics system. With this approach, the student gets a good understanding of how the components of a graphics system work and interact. Since each extension builds on the previous week's work, the instructor may need to provide solutions to previous assignments. One danger in providing algorithms as complete as those given here is a temptation to copy them without really understanding them. This should be countered with classroom explanation and by assigning some of the exercises which are included.

A second approach is to use the presented graphics system as a base for extensions and modifications. Alternative algorithms and methods may be explored. The book provides a well-documented basic support system, or test bed, on which these methods can be actually programmed. The actual implementation and use of an approach can provide insights into the details of its behavior and its system implications, which may be missed when the algorithm is discussed in isolation. Programming problems suggesting extensions have been included.

A third approach is to emphasize *using* a graphics system rather than building one. With this approach, the algorithms serve as examples of graphics system routines, but not as implementation guides. Students should have this system or some other equivalent system available. Classroom lectures can still explain graphics systems and how they work, but programming projects use these features rather than build them. Applications-oriented programming problems are provided for this purpose.

CHANGES IN THE SECOND EDITION

Computer graphics is a rapidly developing field, and even though this is an introductory text, some revision is needed in order to stay current. In addition, weaknesses in the original text have been addressed. Actual changes include a discussion of the popu-

lar Bresenham vector generation algorithm; an introduction to antialiasing techniques and imaging with pixel arrays and patterns; a discussion of the Cohen-Sutherland Outcode clipping algorithm and use of the Blinn shading model; discussions of ray tracing, halftones, and several color topics; and an introduction to Bezier curves, surface patches, and fractals. Three-dimensional clipping was included with the other three-dimensional extensions in Chapter 8, instead of occupying its own chapter. The hidden surface and line chapter was completely redone, providing much more discussion of alternate approaches and less on the algorithm details. It follows Fuch's approach to sorting polygons, simplifying comparison techniques, and does not attempt to discuss polygon decomposition. Other changes include a greatly expanded set of references, an increased selection of exercises and programming problems, and an eight-page insert of full-color photographs (Plates 1 to 16) in Chapter 10.

FURTHER SUPPORT

One thing learned through the first edition of the book is that the text creates the need for information about implementations of the graphics system described. What implementations are available? What languages and computing environments have been tried? What problems or bugs were discovered? What extensions have been carried out? With this edition I shall attempt to act as a clearinghouse for this information. As such, I shall welcome hearing of implementations and requests for implementations. All inquiries can be sent to Steven Harrington, Xerox Corporation, W128-29E, 800 Phillips Rd., Webster, NY 14580.

APPRECIATION

I would like to express my appreciation and thanks to those who reviewed this edition of the text: Steve Cunningham, Rollin Dix, William Grosky, David McAllister, and Spencer Thomas. Their comments and suggestions were very valuable, and I am most grateful for their assistance.

Steven Harrington

CONTENTS

CHAPTER ONE

GEOMETRY AND LINE GENERATION

INTRODUCTION

Perhaps our age will be known as the Information Revolution or the Computer Revolution, for we are witnessing a remarkable growth and development of computer technology and applications. The computer is an information processing machine, a tool for storing, manipulating, and correlating data. We are able to generate or collect and process information on a scope never before possible. This information can help us make decisions, understand our world, and control its operation. But as the volume of information increases, a problem arises. How can this information be efficiently and effectively transferred between machine and human? The machine can easily generate tables of numbers hundreds of pages long. But such a printout may be worthless if the human reader does not have the time to understand it. Computer graphics strikes directly at this problem. It is a study of techniques to improve communication between human and machine. A graph may replace that huge table of numbers and allow the reader to note the relevant patterns and characteristics at a glance. Giving the computer the ability to express its data in pictorial form can greatly increase its ability to provide information to the human user. This is a passive form of graphics, but communication can also be a two-way process. It may be convenient and appropriate to input graphical information to the computer. Thus there are both graphical input and graphical output devices. It is often desirable to have the input from the user alter the output presented by the machine. A dialogue can be established through the graphics medium. This is termed *interactive computer graphics* because the user interacts with the machine. Computer graphics allows communication through pictures, charts, and diagrams. It offers

a vital alternative to the typewriter's string of symbols. The old adage "A picture is worth a thousand words" is certainly true. Through computer graphics we can pilot spaceships; walk through buildings which have yet to be built; and watch bridges collapse, stars being born, and the rotations of atoms. There are many applications for computer graphics. Management information may be displayed as charts and diagrams. Scientific theories and models may be described in pictorial form. (See Plates 1 through 4.) In computer-aided design we can display an aircraft wing, a highway layout, a printed circuit board, a building "blueprint," or a machine part. (See Plate 15.) Maps can be created for all kinds of geographic information. Diagrams and simulations can enrich classroom instruction. The computer has become a new tool for the artist and animator. (See Plates 5 through 14.) And in video games, computer graphics provides a new form of entertainment.

Over the years many graphics display devices have been developed. There are also many software packages and graphics languages. The problem with such diversity is that it makes it difficult to transfer a graphics program from one installation to another. In the late 1970s, the Graphics Standards Planning Committee of the Association for Computing Machinery developed a proposal for a standard graphics system called the CORE system. This system provided a standardized set of commands to control the construction and display of graphic images. The commands were independent of the device used to create or to display the image and independent of the language in which the graphics program was written. The CORE system defined basic graphics primitives from which more complex or special-purpose graphics routines could be built. The idea was that a program written for the CORE system could be run on any installation using that system. The CORE system contained mechanisms for describing and displaying both two-dimensional and three-dimensional structures. However, it was developed just before the reduction in the cost of computer memory made possible economical raster displays (which allow solid and colored areas to be drawn). It therefore lacked the primitives for describing areas and could only create line drawings. Extensions were soon proposed to provide the CORE system with raster imaging primitives.

A second standard called the graphics kernel system (GKS) was developed in Europe, and it has been steadily gaining in popularity. The GKS system was heavily influenced by CORE, and the minimal GKS implementation is essentially identical to CORE's two-dimensional subset. The GKS standard did contain primitives for imaging areas and colors, but it did not contain the constructs for three-dimensional objects. It introduced the concept of a workstation, which allowed a single graphics program to control several graphics terminals.

Another graphics standard is the programmer's hierarchical interactive graphics standard (PHIGS). It takes input and output functions and viewing model from CORE and GKS, but it is a programmer's toolbox, intended for programming graphics applications. It contains enhanced graphics program structuring features. Two other graphics standards are the computer graphics metafile (CGM) and the computer graphics interface (CGI). The CGM is a file format for picture information that allows device-independent capture, storage, and transfer. The CGI is a companion standard which provides a procedural interface for the CGM primitives.

In this book we are going to present the algorithms for constructing a graphics system which have the flavor of the CORE and GKS standards and, in some areas, go beyond them.

We begin our discussion of computer graphics with the fundamental question of how to locate and display points and line segments. There are several hardware devices (graphics terminals) which may be used to display computer-generated images. Some of these will be discussed in Chapter 2. Before we talk about the devices which display points, we shall review the basic geometry which underlies all of our techniques. We shall consider what points and lines are and how we can specify and manipulate them. We conclude this chapter with a discussion of how the mathematical description of these fundamental geometric building blocks can be implemented on an actual display device. Algorithms are presented for carrying out such an implementation for a line printer or common cathode ray tube (CRT) display. These algorithms will allow us (if needed) to use the line printer or CRT as a somewhat crude, but effective, graphics display device for demonstrating the graphics principles described in the rest of the text.

LINES

We can specify a point (a position in a plane) with an ordered pair of numbers (x, y), where x is the horizontal distance from the origin and y is the vertical distance. Two points will specify a line. Lines are described by equations such that if a point (x, y) satisfies the equations, then the point is on the line. If the two points used to specify a line are (x_1, y_1) and (x_2, y_2), then an equation for the line is given by

$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1} \tag{1.1}$$

This says that the slope between any point on the line and (x_1, y_1) is the same as the slope between (x_2, y_2) and (x_1, y_1).

There are many equivalent forms for this equation. Multiplying by the denominators gives the form

$$(x - x_1)(y_2 - y_1) = (y - y_1)(x_2 - x_1) \tag{1.2}$$

A little more algebra solving for y gives

$$y = \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) + y_1 \tag{1.3}$$

or

$$y = mx + b \tag{1.4}$$

where

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

and

$$b = y_1 - mx_1$$

This is called the *slope-intercept* form of the line. The *slope* m is the change in height divided by the change in width for two points on the line (the rise over the run). The *intercept* b is the height at which the line crosses the y axis. This can be seen by noting that the point (0, b) satisfies the equation of the line.

A different form of the line equation, called the *general form*, may be found by multiplying out the factors in Equation 1.2 and collecting them on one side of the equal sign.

$$(y_2 - y_1)\,x - (x_2 - x_1)\,y + x_2y_1 - x_1y_2 = 0 \tag{1.5}$$

or

$$rx + sy + t = 0 \tag{1.6}$$

where possible values for r, s, and t are

$$\begin{aligned} r &= (y_2 - y_1) \\ s &= -(x_2 - x_1) \\ t &= x_2y_1 - x_1y_2 \end{aligned}$$

We say *possible* values because we see that multiplying r, s, and t by any common factor will produce a new set of r′, s′, and t′ values which will still satisfy Equation 1.6 and, therefore, also describe the same line. The values for r, s, and t are sometimes chosen so that

$$r^2 + s^2 = 1 \tag{1.7}$$

Comparing Equations 1.4 and 1.6 we see that

$$m = -\frac{r}{s}$$

and (1.8)

$$b = -\frac{t}{s}$$

Can we determine where two lines will cross? Yes, it is fairly easy to determine where two lines will cross. By the two lines crossing we mean that they share some point in common. That point will satisfy both of the equations for the two lines. The problem is to find this point. Suppose we give the equations for the two lines in their slope-intercept form:

$$\begin{aligned} &\text{line 1:} \quad y = m_1x + b_1 \\ &\text{line 2:} \quad y = m_2x + b_2 \end{aligned} \tag{1.9}$$

Now if there is some point (x_i, y_i) shared by both lines, then

$$y_i = m_1x_i + b_1 \quad \text{and} \quad y_i = m_2x_i + b_2 \tag{1.10}$$

will both be true. Equating over y_i gives

$$m_1x_i + b_1 = m_2x_i + b_2 \tag{1.11}$$

Solving for x_i yields

$$x_i = \frac{b_2 - b_1}{m_1 - m_2} \tag{1.12}$$

Substituting this into the equation for either line 1 or line 2 gives

$$y_i = \frac{b_2 m_1 - b_1 m_2}{m_1 - m_2} \tag{1.13}$$

Therefore, the point

$$\left(\frac{b_2 - b_1}{m_1 - m_2}, \frac{b_2 m_1 - b_1 m_2}{m_1 - m_2} \right) \tag{1.14}$$

is the intersection point. Note that two parallel lines will have the same slope. Since such lines will not intersect, it is not at all surprising that the above expression results in a division by zero. When no point exists, we cannot solve for it.

If the Equation 1.6 form is used to describe the lines, then similar algebra yields an intersection point which is given by

$$\left(\frac{s_1 t_2 - s_2 t_1}{s_2 r_1 - s_1 r_2}, \frac{t_1 r_2 - t_2 r_1}{s_2 r_1 - s_1 r_2} \right) \tag{1.15}$$

LINE SEGMENTS

What are *line segments*? Our equations for lines specify all points in a given direction. The lines extend forever both forward and backward. This is not exactly what we need for graphics. We would like to display only pieces of lines. Let's consider only those points on a line which lie between two endpoints p_1 and p_2. (See Figure 1-1.)

This is called a line segment. A line segment may be specified by its two endpoints. From these endpoints we can determine the equation of the line. From this equation and the endpoints we can decide if any point is or is not on the segment. If the endpoints are $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ and these yield equation $y = mx + b$ (or $rx + sy + t = 0$), then another point $p_3 = (x_3, y_3)$ lies on the segment if

1. $y_3 = mx_3 + b$ (or $rx_3 + sy_3 + t = 0$)
2. $\min(x_1, x_2) \leq x_3 \leq \max(x_1, x_2)$
3. $\min(y_1, y_2) \leq y_3 \leq \max(y_1, y_2)$

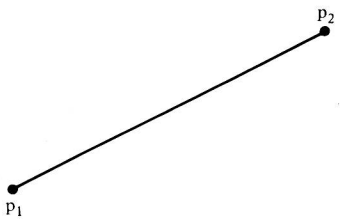


FIGURE 1-1
A line segment.