PROCEDURAL ELEMENTS FOR COMPUTER GRAPHICS

David F. Rogers

PROCEDURAL ELEMENTS FOR COMPUTER GRAPHICS

David F. Rogers

Professor of Aerospace Engineering
and
Director, Computer Aided Design
and Interactive Graphics
United States Naval Academy, Annapolis, Md.

McGraw-Hill Book Company

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

This book was computer phototypeset in Times Roman, by TYX Corporation.
The editors were Kiran Verma and David A. Damstra;
the production supervisor was Joe Campanella;
project supervision was by the author.
The cover was designed by Fern Logan and the author.
R. R. Donnelley & Sons Company was printer and binder.

PROCEDURAL ELEMENTS FOR COMPUTER GRAPHICS

1 2 3 4 5 6 7 8 9 0 D O C D O C 8 9 8 7 6 5 4

ISBN 0-07-053534-5

Cover illustration credits:

Front cover: A dimetric view of three blocks rendered by John O. Jenkins at the Johns Hopkins University Applied Physics Laboratory using a Watkins algorithm developed as part of a class project.

Back cover: Orthographic and isometric views of three blocks rendered by John O. Jenkins at The Johns Hopkins University Applied Physics Laboratory using a Watkins algorithm developed as part of a class project.

Library of Congress Cataloging in Publication Data

Rogers, David F., date
Procedural elements for computer graphics.

Includes bibliographical references and index.
1. Computer graphics. I. Title.
T385.R63 1985 001.64'43 83-24403
ISBN 0-07-053534-5

PREFACE

Computer graphics is now a mature discipline. Both hardware and software are available that facilitate the production of graphical images as diverse as line drawings and realistic renderings of natural objects. A decade ago the hardware and software to gen- ate these graphical images cost hundreds of thousands of dollars. Today, excellent facilities are available for expenditures in the tens of thousands of dollars and lower performance, but in many cases adequate facilities are available for tens of hundreds of dollars. The use of computer graphics to enhance information transfer and understanding is endemic in almost all scientific and engineering disciplines. Today, no scientist or engineer should be without a basic understanding of the underlying principles of computer graphics. Computer graphics is also making deep inroads into the business, medical, advertising, and entertainment industries. The presence in the boardroom of presentation slides prepared using computer graphics facilities as well as more commonplace business applications is considered the norm. Three-dimensional reconstructions using data obtained from CAT scans is becoming commonplace in medical applications. Television as well as other advertising media are now making frequent use of computer graphics and computer anima The entertainment industry has embraced computer graphics with applications diverse as video games and full-length feature films. Even art is not immune, as e enced by some of the photos included in this book.

It is almost a decade now since the appearance of the companion volume to this book, **Mathematical Elements for Computer Graphics**. During that time significant strides in raster scan graphics have been made. The present volume concentrates on these aspects of computer graphics. The book starts with an introduction to computer graphics hardware with an emphasis on the concep-

tual understanding of cathode ray tube displays and of interactive devices. The following chapters look at raster scan graphics including line and circle drawing, polygon filling, and antialiasing algorithms; two- and three-dimensional clipping including clipping to arbitrary convex volumes; hidden-line and hidden-surface algorithms including ray tracing; and finally, rendering, the "art" of making realistic pictures, including local and global illumination models, texture, shadows, transparency, and color effects. The book continues the presentation technique of its predecessor. Each thorough topic discussion is followed by presentation of a detailed algorithm or a worked example, and where appropriate both.

The material in the book can be used in its entirety for a semester-long first formal course in computer graphics at either the senior undergraduate or graduate level with an emphasis on raster scan graphics. If a first course in computer graphics based on the material in the companion volume **Mathematical Elements for Computer Graphics** is presented, then the material in this book is ideal for a second course. This is the way it is used by the author. If broader material coverage in a single-semester course is desired, then the two volumes can be used together. Suggested topic coverage is: Chapter 1 of both volumes, followed by Chapters 2 and 3 with selected topics from Chapter 4 of **Mathematical Elements for Computer Graphics**, then selected topics from Chapter 2 (e.g., 2-1 to 2-5, 2-7, 2-15 to 2-19, 2-22, 2-23, 2-28), Chapter 3 (e.g., 3-1, 3-2, 3-4 to 3-6, 3-9, 3-11, 3-15, 3-16), Chapter 4 (e.g., 4-1, part of 4-2 for backplane culling, 4-3, 4-4, 4-7, 4-9, 4-11, 4-13), and Chapter 5 (e.g., 5-1 to 5-3, 5-5, 5-6, 5-14) of the present volume. The book is also designed to be useful to professional programmers, engineers, and scientists. Further, the detailed algorithms and worked examples make it particularly suitable for self-study at any level. Sufficient background is provided by college level mathematics and a knowledge of a higher-level programming language. Some knowledge of data structures is useful but not necessary.

There are two types of algorithms presented in the book. The first is a detailed procedural description of the algorithm, presented in narrative style. The second is more formal and uses an algorithmic 'language' for presentation. Because of the wide appeal of computer graphics, the choice of an algorithmic presentation language was especially difficult. A number of colleagues were questioned as to their preference. No consensus developed. Computer science faculty generally preferred PASCAL but with a strong sprinkling of C. Industrial colleagues generally preferred FORTRAN for compatibility with existing software. The author personally prefers BASIC because of its ease of use. Consequently, detailed algorithms are presented in pseudocode. The pseudocode used is based on extensive experience teaching computer graphics to classes that do not enjoy knowledge of a common programming language. The pseudocode is easily converted to any of the common computer languages. An appendix discusses the pseudocode used. The pseudocode algorithms presented in the book have all been either directly implemented from the pseudocode or the pseudocode has been derived from an operating program in one or more of

the common programming languages. Implementations range from BASIC on an Apple IIe to PL1 on an IBM 4300 with a number of variations in between. A suit of demonstration programs in available from the author.

A word about the production of the book may be of interest. The book was computer typeset using the TEX typesetting system at TYX Corporation of Reston, Virginia. The manuscript was coded directly from handwritten copy. Galleys and two sets of page proofs were produced on a laser printer for editing and page makeup. Final reproduction copy ready for art insertion was produced on a phototypesetter. The patience and assistance of Jim Gauthier and Mark Hoffman at TYX while the limits of the system were explored and solutions to all the myriad small problems found is gratefully acknowledged. The outstanding job done by Louise Bohrer and Beth Lessels in coding the handwritten manuscript is gratefully acknowledged. The usually fine McGraw-Hill copyediting was supervised by David Damstra and Sylvia Warren.

No book is ever written without the assistance of many individuals. The book is based on material prepared for use in a graduate level course given at the Johns Hopkins University Applied Physics Laboratory Center beginning in 1978. Thanks are due the many students in this and other courses from whom I have learned so much. Thanks are due Turner Whitted who read the original outline and made valuable suggestions. Thanks are expressed to my colleagues Pete Atherton, Brian Barsky, Ed Catmull, Rob Cook, John Dill, Steve Hansen, Bob Lewand, Gary Meyer, Alvy Ray Smith, Dave Warn, and Kevin Weiler, all of whom read one or more chapters or sections, usually in handwritten manuscript form, red pencil in hand. Their many suggestions and comments served to make this a better book. Thanks are extended to my colleagues Linda Rybak and Linda Adlum who read the entire manuscript and checked the examples. Thanks are due three of my students: Bill Meier who implemented the Roberts algorithm, Gary Boughan who originally suggested the test for convexity discussed in Sec. 3-7, and Norman Schmidt who originally suggested the polygon splitting technique discussed in Sec. 3-8. Thanks are due Mark Meyerson who implemented the splitting algorithms and assured that the technique was mathematically well founded. The work of Lee Billow and John Metcalf who prepared all the line drawings is especially appreciated.

Special thanks are due Steve Satterfield who read and commented on all 800 handwritten manuscript pages. Need more be said!

Special thanks are also due my eldest son Stephen who implemented all of the hidden surface algorithms in Chapter 4 as well as a number of other algorithms throughout the book. Our many vigorous discussions served to clarify a number of key points.

Finally, a very special note of appreciation is extended to my wife Nancy and to my other two children, Karen and Ransom, who watched their husband and father disappear into his office almost every weeknight and every weekend for a year and a half with never a protest. That is support! Thanks.

David F. Rogers

CONTENTS

CHAPTER

ONE

INTRODUCTION TO COMPUTER GRAPHICS

Computer graphics is now a maturing technology. The underlying elements of manipulative transformations and curve and surface descriptions are well understood and documented (see Refs. 1-1 to 1-3). Raster scan technology, clipping, hidden lines and hidden surfaces, color, shading, texture, and transparency effects are also understood but still developing. It is these latter topics which are of present interest.

1-1 OVERVIEW OF COMPUTER GRAPHICS

Computer graphics is a complex and diversified technology. To begin to understand the technology it is necessary to subdivide it into manageable parts. This can be accomplished by considering that the end product of computer graphics is a picture. The picture may, of course, be used for a large variety of purposes; e.g., it may be an engineering drawing, an exploded parts illustration for a service manual, a business graph, an architectural rendering for a proposed construction or design project, an advertising illustration, or a single frame from an animated movie. The picture is the fundamental cohesive concept in computer graphics. We must therefore consider how

Pictures are represented in computer graphics
Pictures are prepared for presentation
Previously prepared pictures are presented
Interaction with the picture is accomplished

Although many algorithms accept picture data as polygons or edges, each polygon or edge can in turn be represented by vertex points. Points, then, are the fundamental building blocks of picture representation. Of equal fundamental importance is the algorithm which explains how to organize the points. To

illustrate this consider a unit square in the first quadrant. The unit square can be represented by its four corner points (see Fig. 1-1)

$$P_1(0, 0), \quad P_2(1, 0), \quad P_3(1, 1), \quad P_4(0, 1)$$

An associated algorithmic description might be

Connect P_1 P_2 P_3 P_4 P_1 in sequence

The unit square can also be described by four edges

$$E_1 \equiv P_1P_2, \quad E_2 \equiv P_2P_3, \quad E_3 \equiv P_3P_4, \quad E_4 \equiv P_4P_1$$

Here the algorithmic description is

Display E_1 E_2 E_3 E_4 in sequence

Finally, either the points or edges can be used to describe the unit square as a single polygon, e.g.,

$$S_1 = P_1\ P_2\ P_3\ P_4\ P_1 \quad \text{or} \quad P_1\ P_4\ P_3\ P_2\ P_1$$

or

$$S_1 = E_1\ E_2\ E_3\ E_4$$

The fundamental building blocks, i.e. points, can be represented as either pairs or triplets of numbers depending on whether the data are two- or three-dimensional. Thus, (x_1, y_1) or (x_1, y_1, z_1) would represent a point in either two- or three-dimensional space. Two points would represent a line or edge, and a collection of three or more points a polygon. These points, edges, or polygons are collected or stored in a data base. The data used to prepare the picture for presentation is rarely the same as that used to present the picture. The data used to present the picture is frequently called a display file. The display file will represent some portion, view, or scene of the picture represented by the total data base. The displayed picture is usually formed by rotating, translating, scaling, and performing various projections on the data. These basic orientation or viewing preparations are generally performed using a 4×4 transformation matrix operating on the data represented in homogeneous

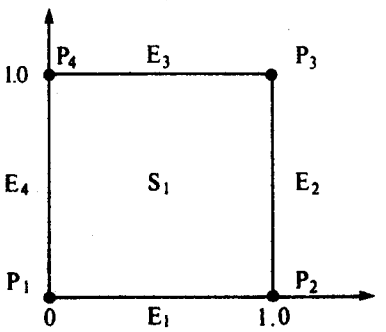

Figure 1-1 Picture data descriptions.

coordinates (see Ref. 1-1). Frequently these operations are implemented in hardware. Hidden line or hidden surface removal, shading, transparency, texture, or color effects may be added before final presentation. If the picture represented by the entire data base is not to be presented, the appropriate portion must be selected. This is a process called clipping. Clipping may be two- or three-dimensional as appropriate. In some cases the clipping window or volume may have holes in it or may be irregularly shaped. Clipping to standard two- and three-dimensional regions is frequently implemented in hardware.

Almost all pictures involve the presentation of textual material. Characters can be generated in either hardware or software. If generated in software, they can be manipulated and treated like any other portion of the picture. If generated in hardware, they are maintained as character codes until just prior to display. Usually only limited manipulative capabilities are provided; e.g., only limited rotations and sizes are available. Clipping of hardware-generated characters is generally not possible. Either the entire character is displayed or none of it is displayed.

1-2 TYPES OF GRAPHICS DEVICES

The display medium for computer graphics-generated pictures has become widely diversified. Typical examples are pen-and-ink plotters, dot matrix, electrostatic or laser printer plotters, film, storage tube, calligraphic refresh, and raster scan cathode ray tube (CRT) displays. Because the large majority of computer graphics systems utilize some type of CRT display and because most of the fundamental display concepts are embodied in CRT display technology, we will limit our discussion to CRT displays. Other display technologies are discussed in Refs. 1-1 to 1-3.

The three most common types of CRT display technologies are direct-view storage tube (line drawing), calligraphic (line drawing) refresh, and raster scan (point plotting) refresh displays. With recent advances, an individual display may incorporate more than one technology. In discussing the various displays we take a user's, or conceptual, point of view; i.e., we are generally concerned with functional capabilities and not with the details of the electronics.

1-3 STORAGE TUBE GRAPHICS DISPLAYS

The direct-view storage tube is conceptually the simplest of the CRT displays. The storage tube display, also called a bistable storage tube, can be considered a CRT with a long-persistence phosphor. A line or character will remain visible (up to an hour) until erased. A typical display is shown in Fig. 1-2. To draw a line on the display the intensity of the electron beam is increased sufficiently to cause the phosphor to assume its permanent bright "storage" state. The display is erased by flooding the entire tube with a specific voltage which causes the

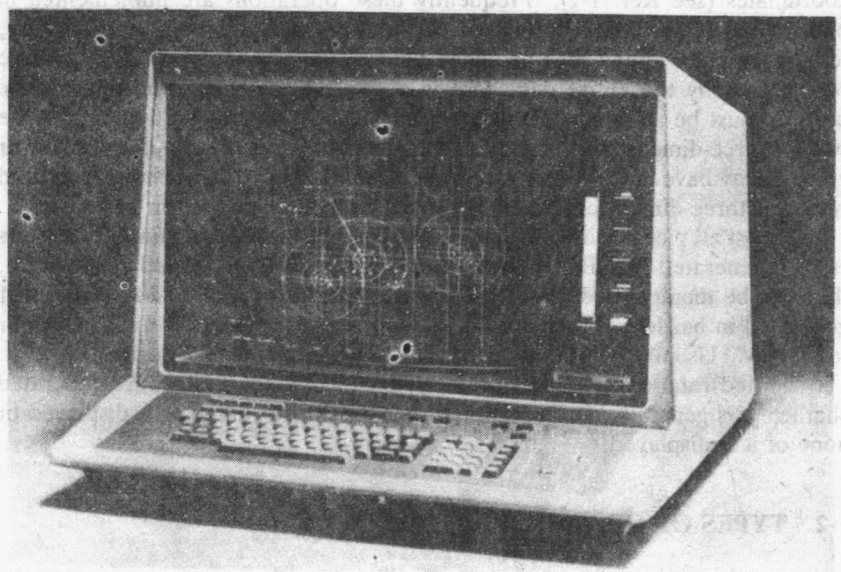

Figure 1-2 Storage tube graphics display.

phosphor to assume its dark state. Erasure takes about 1/2 second. Because the entire tube is flooded, all lines and characters are erased. Thus, individual lines and characters cannot be erased, and the display of dynamic motion or animation is not possible. An intermediate state (write-through mode) is sometimes used to provide limited refresh capability (see below). Here, the electron beam is intensified to a point that is just below the threshold that will cause permanent storage but is still sufficient to brighten the phosphor. Because the image in this mode does not store, it must be redrawn or repainted continuously in order for it to be visible.

A storage tube display is flicker-free (see below) and capable of displaying an "unlimited" number of vectors. Resolution is typically 1024 × 1024 addressable points (10 bits) on an 8 × 8 inch square (11-inch-diagonal CRT) or 4096 × 4096 (12 bits) on either a 14 × 14 inch square (19-inch-diagonal CRT) or an 18 × 18 inch square (25-inch-diagonal CRT). Typically only 78 percent of the addressable area is viewable in the vertical direction.

A storage tube display is a line drawing or random scan display. This means that a line (vector) can be drawn directly from any addressable point to any other addressable point. Hard copy is relatively easy, fast, and inexpensive

to obtain. Conceptually, a storage tube display is somewhat easier to program than a calligraphic or raster scan refresh display. Storage tube CRT displays can be combined with microcomputers into stand-alone computer graphics systems or incorporated into graphics terminals. When incorporated into terminals, alphanumeric and graphic information are passed to the terminal by a host computer over an interface. Although parallel interfaces are available, typically a serial interface which passes information 1 bit at a time is used. Because of the typically low interface speed and the erasure characteristics, the level of interactivity with a storage tube display is lower than with either a refresh or raster scan display.

1-4 CALLIGRAPHIC REFRESH GRAPHICS DISPLAYS

In contrast to the storage tube display, a calligraphic (line drawing or vector) refresh CRT display uses a very short-persistence phosphor. These displays are frequently called random scan displays (see below). Because of the short persistence of the phosphor, the picture painted on the CRT must be repainted or refreshed many times each second. The minimum refresh rate is at least 30 times each second, with a recommended rate of 40 to 50 times each second. Refresh rates much lower than 30 times each second result in a flickering image. The effect is similar to that observed when a movie film is run too slowly. The resulting picture is difficult to use and disagreeable to look at.

The basic calligraphic refresh display requires two elements in addition to the CRT. These are the display buffer and the display controller. The display buffer is contiguous memory containing all the information required to draw the picture on the CRT. The display controller's function is to repeatedly cycle through this information at the refresh rate. Two factors which limit the complexity (number of vectors displayed) of the picture are the size of the display buffer and the speed of the display controller. A further limitation is the speed at which picture information can be processed, i.e. transformed and clipped, and textual information generated.

Figure 1-3 shows two block diagrams of two high-performance calligraphic refresh displays. In both cases it is assumed that picture transformations such as rotation, translation, scaling, perspective, and clipping are implemented in hardware in the picture processor. In the first case (Fig. 1-3a) the picture processor is slower than the refresh rate for useful pictures (4000 to 5000 vectors). Thus, the picture data sent by the host central processing unit (CPU) to the graphics display is processed before being stored in the display buffer. Here the display buffer contains only those precise instructions which are required by the vector/character generator to draw the picture. Vectors are generally held in screen coordinates. The display controller reads information from the display buffer and sends it to the vector/character generator. When the display controller reaches the end of the display buffer, it returns to the beginning and cycles through the buffer again.

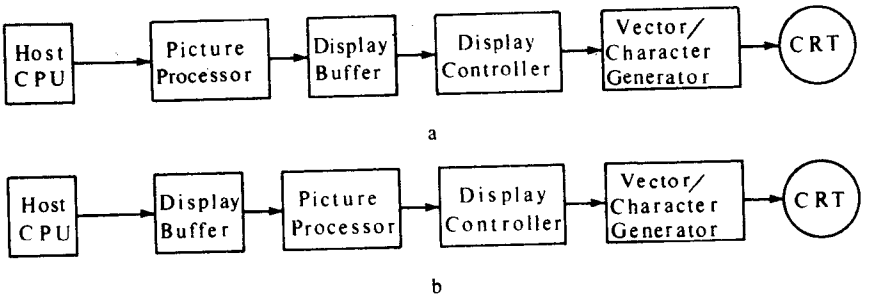

Figure 1-3 Conceptual block diagrams of calligraphic refresh displays.

This first configuration also gives rise to the concepts of double buffering and separate update and refresh rates. Since in this configuration the picture processor is too slow to generate a complex new or updated picture within one refresh cycle, the display buffer is divided into two parts. While an updated picture is being processed and written into one half of the buffer, the display controller is refreshing the CRT from the other half of the buffer. When the updated picture is complete, the buffers are swapped and the process is repeated. Thus, a new or updated picture may be generated every second, third, fourth, etc., refresh cycle. Double buffering prevents part of the old picture being displayed along with part of the new updated picture during one or more refresh cycles.

In the second configuration (see Fig. 1-3b) the picture processor is faster than the refresh rate for complex pictures. Here, the original picture data base sent from the host CPU is held directly in the display buffer. Vectors are generally held in user (world) coordinates as floating point numbers. The display controller reads information from the display buffer, passes it through the picture processor, and sends it to the vector generator in one refresh cycle. This implies that picture transformations are performed "on the fly" within one refresh cycle.

In either configuration, each vector, character, and picture drawing instruction exists in the display buffer. Hence, any individual element may be changed independent of any other element. This feature, in combination with the short persistence of the CRT phosphor, allows the display of dynamic motion. Figure 1-4 illustrates this concept. Figure 1-4 shows the picture displayed during four successive refresh cycles. The visible solid line is the displayed line for the current refresh cycle, and the invisible dotted line is for the previous refresh cycle. Between refresh cycles the location of the end of the line, B, is changed. The line will appear to rotate about the point A.

In many pictures only portions of the picture are dynamic. In fact, in many applications the majority of the picture is static. This leads to the concept of segmentation of the display buffer. Figure 1-5 illustrates this idea. Here, the baseline, the cross-hatching, and the letter A used to show the support for the line AB are static; i.e., they do not change from refresh cycle to refresh cycle.

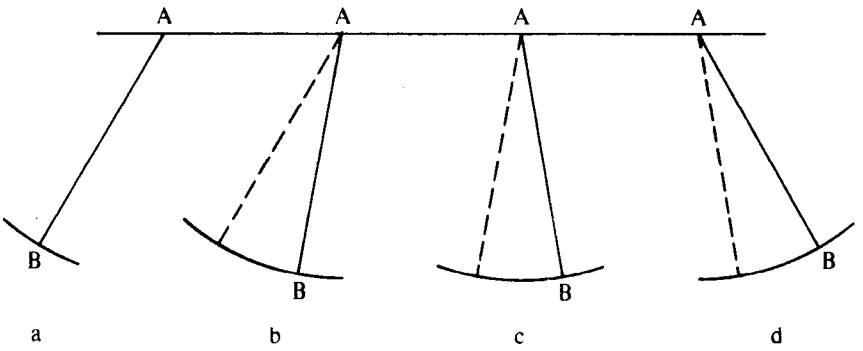

Figure 1-4 Dynamic motion.

In contrast, the location of the end of the line *AB* and the letter *B* change from refresh cycle to refresh cycle to show dynamic motion. These separate portions of the picture data base are placed in separate segments of the display buffer. Since the static segment of the display buffer does not change, it can be ignored by the picture processor for the configuration shown in Fig. 1-3a. This significantly reduces the work load on the picture processor when updating a picture. In this case, only the picture in the dynamic segment need be updated. Further, it reduces the amount of data that need be transmitted from the host CPU to the picture processor during each picture update.

For the configuration shown in Fig. 1-3b a different type of segmentation is possible. Recall that for this configuration the picture data base is stored in the display buffer in world (user) coordinates and picture processing occurs on the fly once each refresh cycle. For the picture in Fig. 1-5 two segments are created in the display buffer, a static and a dynamic segment. However, picture processing occurs on the fly. Dynamic update of the information in the

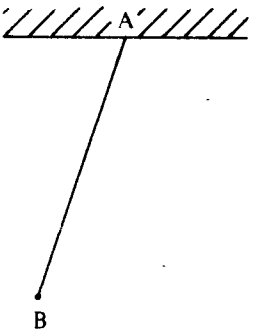

Figure 1-5 Display buffer segmentation.

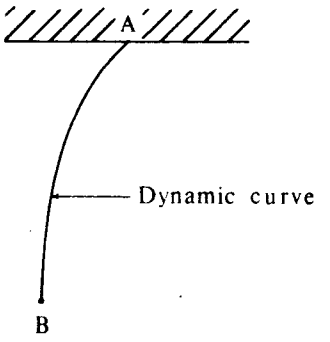

Figure 1-6 Intelligent display buffer segmentation.

dynamic segment can be accomplished using functions available in the picture processor. Thus, picture update can occur locally within the graphics device, and communication with the host CPU is unnecessary. For the particular case shown in Fig. 1-5 the only picture processor function required for local dynamic update is rotation about the point *A*.

Figure 1-6 illustrates a picture for which dynamic update requires communication with the host CPU, i.e., intelligent update of the picture. Again two segments are created, a static segment comprised of the baseline, crosshatching, and the letter *A*, and a dynamic segment comprised of the curve *AB* and the letter *B*. Assume that the shape of the curve *AB* will change from refresh cycle to refresh cycle depending upon physical factors. Thus, the shape must be computed by an application program running in the host CPU. In order to update the dynamic picture segment new data, e.g. curve shape, must be sent to and stored in the display buffer.

Although the concept of picture segmentation has been introduced through dynamic motion examples, it is not limited to dynamic motion or animation. Any picture can be segmented. Picture segmentation is particularly useful for interactive graphics programs. The concept is similar to modular programming. The choice of modular picture segments, their size, and their complexity depends on the particular application. Individual picture elements can be as simple as single points or as complex as complete object descriptions. Reference 1-3 provides additional discussion.

To illustrate the importance of the communication speed, or bandwidth, between the host CPU and the graphics device consider the requirements for intelligently updating a curved line with 250 segments or points describing it. Each point is described by three coordinates. If we assume that a floating point representation with six significant figures (characters) is used, and that a single 8-bit byte is used to represent a character, then for a refresh rate of 30 frames per second and an update every refresh cycle the required communication bandwidth is

$$30[(no.\ points)(no.\ coor./point)(no.\ of\ sig.\ figs./point)(no.\ bits/char.)]$$

or

$$30(250)(3)(6)(8) = 1{,}080{,}000\ bits/s$$

Thus, the required bandwidth can easily exceed 1 megabit per second. For complicated three-dimensional sculptured surfaces, the required bandwidth can easily exceed 10 times this, i.e., 10 megabits per second. In most cases this dictates a parallel or direct memory access (DMA) interface between the host CPU and the graphics device to support real-time intelligent dynamic graphics. A typical calligraphic refresh display is shown in Fig. 1-7.

1-5 RASTER REFRESH GRAPHICS DISPLAYS

Both the storage tube CRT display and the random scan refresh display are line drawing devices. That is, a straight line can be drawn directly from any addressable point to any other addressable point. In contrast is the raster CRT