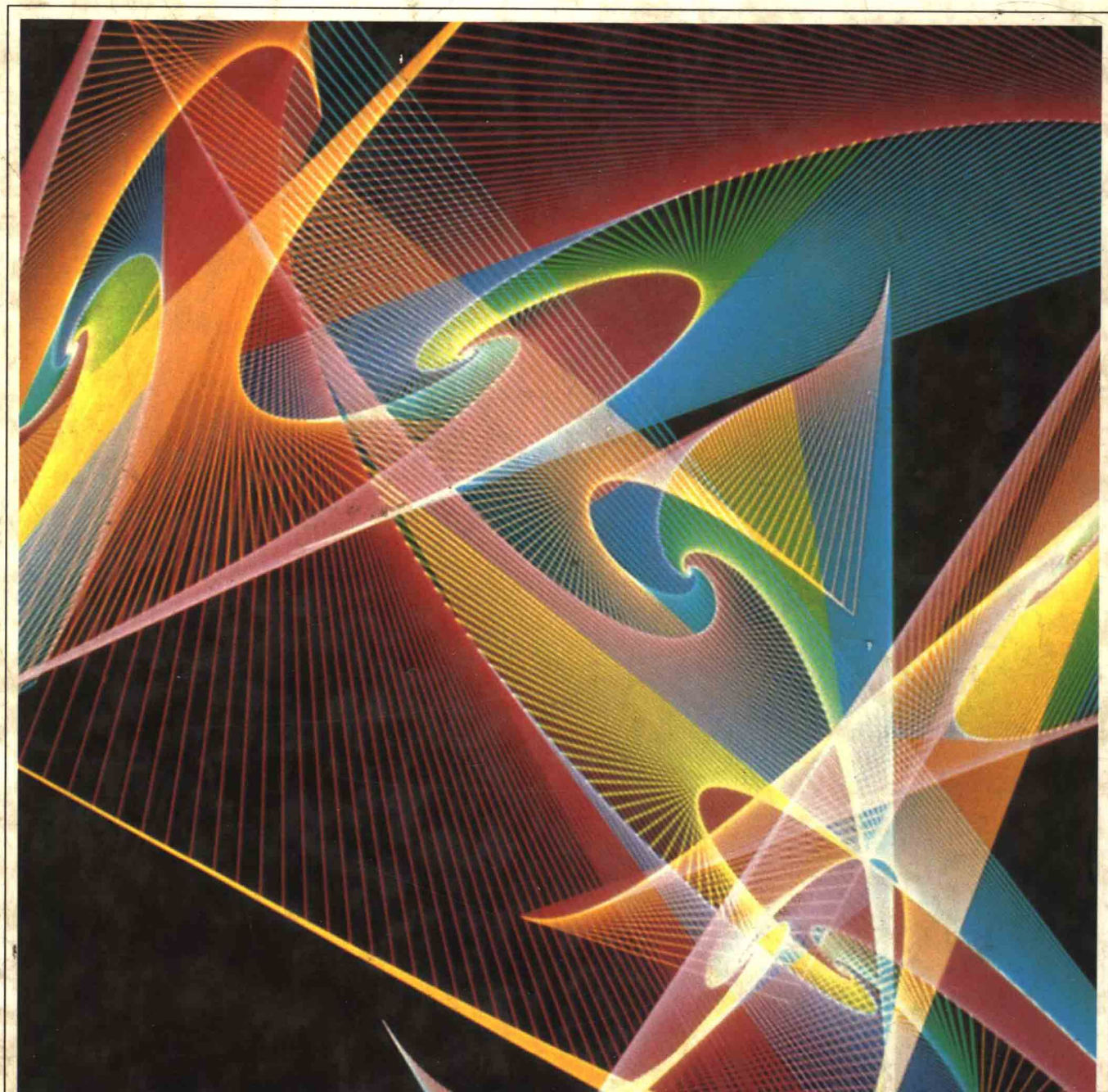
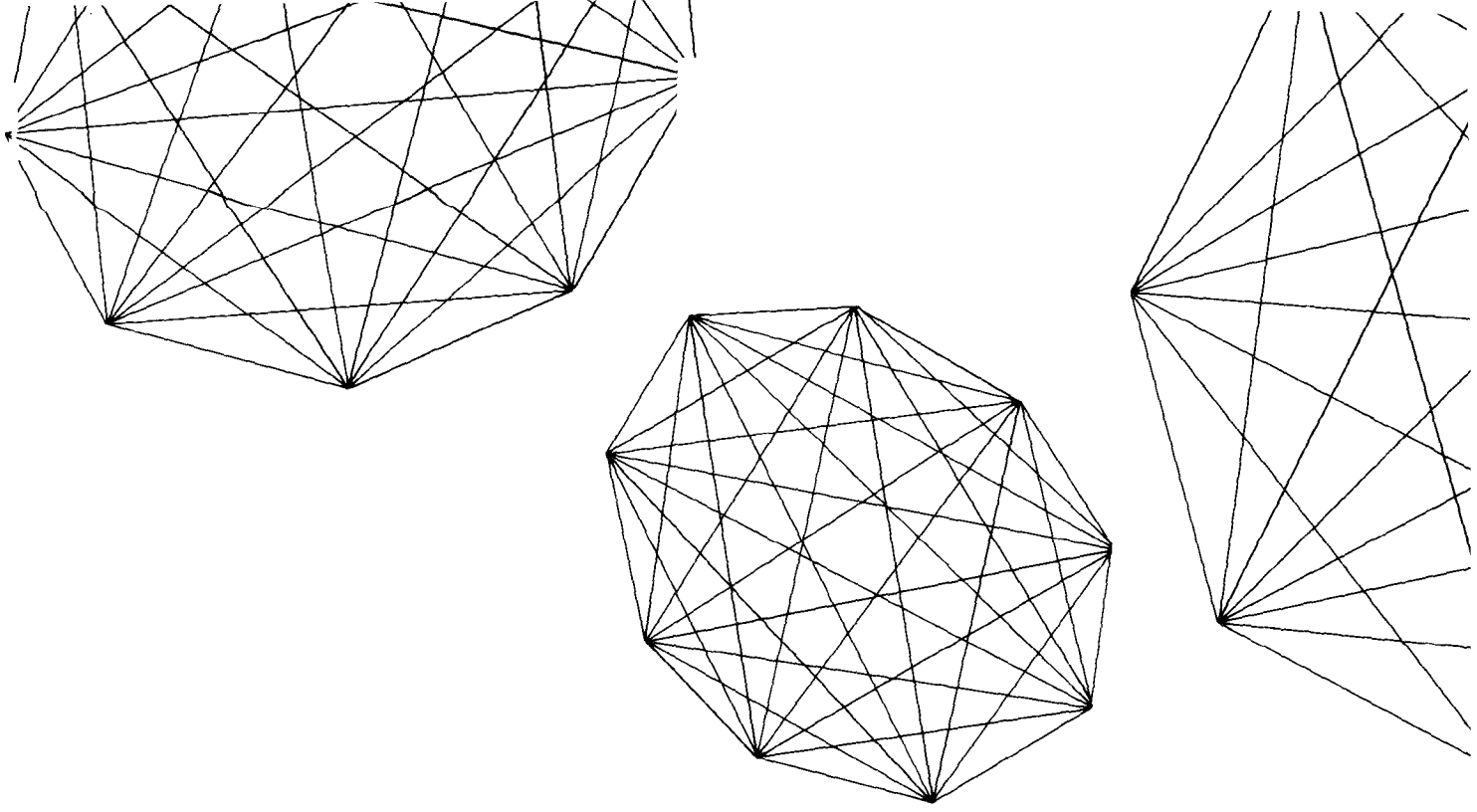
COMPUTER GRAPHICS

Donald Hearn / M. Pauline Baker





COMPUTER GRAPHICS

Donald Hearn

Department of Computer Science, University of Illinois

M. Pauline Baker

Department of Computer Science, Western Illinois University

PRENTICE-HALL, INC., Englewood Cliffs, New Jersey 07632

PREFACE

Computer graphics is one of the most exciting and rapidly growing fields in computer science. Some of the most sophisticated computer systems in use today are designed for the generation of graphics displays. We all know the value of a picture as an effective means for communication, and the ability to converse pictorially with a computer is revolutionizing the way computers are being used in all areas.

This book presents the basic principles for the design, use, and understanding of graphics systems. We assume that the reader has no prior background in computer graphics but is familiar with fundamental computer science concepts and methods. The hardware and software components of graphics systems are examined, with a major emphasis throughout on methods for the design of graphics packages. We discuss the algorithms for creating and manipulating graphics displays, techniques for implementing the algorithms, and their use in diverse applications. Programming examples are given in Pascal to demonstrate the implementation and application of graphics algorithms. We also introduce the reader to the Graphical Kernel System (GKS), which is now both the United States and the international graphics-programming language standard. GKS formats for graphics-routine calls are used in the Pascal programs illustrating graphics applications.

The material presented in this text was developed from notes used in graduate and undergraduate graphics courses over the past several years. All of this material could be covered in a one-semester course, but this requires a very hasty treatment of many topics. A better approach is to select a subset of topics, depending on the level of the course. For the self-study reader, early chapters can be used to provide an understanding of graphics concepts, with individual topics selected from the later chapters according to the interests of the reader.

Chapter 1 is a survey of computer graphics, illustrating the diversity of applications-areas. Following an introduction to the hardware and software components of graphics systems in Chapter 2, fundamental algorithms for the generation of two-dimensional graphics displays are presented in Chapters 3 and 4. These two chapters examine methods for producing basic picture components and techniques for handling color, shading, and other attributes. This introduces students to the programming techniques necessary for implementing graphics routines. Chapters 5 and 6 treat transformations and viewing algorithms. Methods for organizing picture components in segments and for interactive input are given in Chapters 7 and 8.

Three-dimensional techniques are introduced in Chapter 9. We then discuss the different ways that solid objects can be represented (Chapter 10) and manipulated (Chapter 11). Methods for forming three-dimensional views on a graphics display device are detailed in Chapter 12. The various algorithms for removing hidden surfaces of objects are discussed in Chapter 13, and models for shading and color are taken up in Chapter 14. These five chapters treat both the standard graphics methods and newer techniques, such as fractals, octrees, and ray tracing.

In Chapter 15, we explore techniques for modeling different systems. Modeling packages provide the structure for simulating systems, which is then passed to the graphics routines for display. Finally, methods for interfacing a graphics package to the user are examined in Chapter 16.

At the undergraduate level, an introductory course can be organized with a detailed treatment of fundamental topics from Chapters 2 through 8 plus an introduction to three-dimensional concepts and methods. Selected topics from the later chapters can be used as supplemental material. For a graduate course, the material on two-dimensional methods can be covered at a faster pace, with greater emphasis on the later chapters. In particular, methods for three-dimensional representations, three-dimensional viewing, hidden-surface removal, and shading and color models, can be covered in greater depth.

A great many people have contributed to this project in a variety of ways. To the many organizations and individuals who furnished photographs and other materials, we again express our appreciation. We are also grateful to our graphics students for their comments on the presentation of this material in the classroom. We thank the many people who provided comments on the manuscript, and we are especially indebted to Norman Badler, Brian Barsky, and Steve Cunningham for their helpful suggestions for improving the presentation of material. And a very special thanks goes to our editor, Jim Fegen, for his patience and encouragement during the preparation of this book, and to our production editors, Tracey Orbine and Kathy Marshak. Thanks also to our designer, Lee Cohen, and the Prentice-Hall staff for an outstanding production job.

Donald Hearn
M. Pauline Baker

CONTENTS

PREFACE xv

1 *A SURVEY OF COMPUTER GRAPHICS*

1-1 Computer-Aided Design 2
1-2 Graphs, Charts, and Models 8
1-3 Computer Art 12
1-4 Computer Animation 16
1-5 Graphical User Interfaces 20
1-6 Graphics for Home Use 21
1-7 Image Processing 22
References 26

2 *OVERVIEW OF GRAPHICS SYSTEMS 27*

2-1 Display Devices 28
Refresh Cathode-Ray Tubes 29
Random-Scan and Raster-Scan Monitors 31
Color CRT Monitors 33
Direct-View Storage Tubes 35
Plasma-Panel Displays 36
LED and LCD Monitors 38
Laser Devices 38
Three-Dimensional Monitors 39
2-2 Hard-Copy Devices 40
Printers 41
Plotters 42
2-3 Interactive Input Devices 44
2-4 Display Processors 46
Random-Scan Systems 47

DVST Systems 48
Raster-Scan Systems 48
2-5 Graphics Software 50
Coordinate Representations 50
Graphics Functions 51
Software Standards 51
References 52
Exercises 53

3 OUTPUT PRIMITIVES 55
3-1 Points and Lines 56
3-2 Line-Drawing Algorithms 56
DDA Algorithm 57
Bresenham's Line Algorithm 58
Loading the Frame Buffer 61
3-3 Antialiasing Lines 62
3-4 Line Command 63
3-5 Fill Areas 65
3-6 Circle-Generating Algorithms 65
Circle Equations 65
Bresenham's Circle Algorithm 67
Ellipses 69
3-7 Other Curves 70
3-8 Character Generation 70
3-9 Instruction Sets for Display Processors 72
Raster-Scan Systems 72
Random-Scan Systems 72
3-10 Summary 73
3-11 Applications 73
References 76
Exercises 76

4 ATTRIBUTES OF OUTPUT PRIMITIVES 78
4-1 Line Styles 79
Line Type 79
Line Width 80
Line Color 80

4-2 Color and Intensity 81
Color Tables 81
Gray Scale 82
4-3 Area Filling 83
Scan-Line Algorithm 83
Antialiasing Area Boundaries 91
Boundary-Fill Algorithm 92
Flood-Fill Algorithm 94
Area-Filling Commands 94
4-4 Character Attributes 96
Text Attributes 96
Marker Attributes 98
4-5 Inquiry Functions 98
4-6 Bundled Attributes 99
Line Attributes 100
Color and Intensity Attributes 100
Area-Filling Attributes 100
Text Attributes 101
Marker Attributes 102
4-7 Summary 102
References 103
Exercises 103

5 TWO-DIMENSIONAL TRANSFORMATIONS 106

5-1 Basic Transformations 107
Translation 107
Scaling 107
Rotation 108
5-2 Matrix Representations and Homogeneous Coordinates 109
5-3 Composite Transformations 111
Translations 111
Scalings 111
Rotations 111
Scaling Relative to a Fixed Point 112
Rotation About a Pivot Point 112
Arbitrary Scaling Directions 113
Concatenation Properties 114
General Transformation Equations 114

5-4 Other Transformations 116
Reflection 116
Shear 118
5-5 Transformation Commands 119
5-6 Raster Methods for Transformations 121
References 121
Exercises 122

6 WINDOWING AND CLIPPING 123

6-1 Windowing Concepts 124
6-2 Clipping Algorithms 127
Line Clipping 128
Area Clipping 134
Text Clipping 139
Blanking 139
6-3 Window-to-Viewport Transformation 140
References 141
Exercises 141

7 SEGMENTS 143

7-1 Segment Concepts 144
7-2 Segment Files 145
7-3 Segment Attributes 148
7-4 Multiple Workstations 150
7-5 Summary 152
References 153
Exercises 153

8 INTERACTIVE INPUT METHODS 154

8-1 Physical Input Devices 155
Keyboards 155
Touch Panels 157
Light Pens 158
Graphics Tablets 159
Joysticks 161
Trackball 162
Mouse 162
Voice Systems 163

8-2 Logical Classification of Input Devices 164
8-3 Locator Devices 164
8-4 Stroke Devices 166
8-5 String Devices 166
8-6 Valuator Devices 166
8-7 Choice Devices 167
8-8 Pick Devices 168
8-9 Interactive Picture-Construction Techniques 169
Basic Positioning Methods 169
Constraints 170
Grids 171
Gravity Field 171
Rubber-Band Methods 172
Sketching 173
Dragging 173
8-10 Input Functions 174
Input Modes 174
Request Mode 175
Sample Mode 176
Event Mode 176
Concurrent Use of Input Modes 178
8-11 Summary 178
References 179
Exercises 179

9 THREE-DIMENSIONAL CONCEPTS 181
9-1 Three-Dimensional Coordinate Systems 182
9-2 Three-Dimensional Display Techniques 183
Parallel Projection 184
Perspective Projection 184
Intensity Cuing 185
Hidden-Line Removal 185
Hidden-Surface Removal and Shading 185
Exploded and Cutaway Views 186
Three-Dimensional and Stereoscopic Views 187
9-3 Three-Dimensional Graphics Packages 187
References 188

10 THREE-DIMENSIONAL REPRESENTATIONS 189
10-1 Polygon Surfaces 190
Polygon Tables 190
Plane Equations 192
10-2 Curved Surfaces 193
Parametric Equations 194
Bézier Curves 195
Spline Curves 200
Bézier Surfaces 202
Spline Surfaces 204
Methods for Surface Generation 204
10-3 Fractal-Geometry Methods 205
10-4 Sweep Representations 213
10-5 Constructive Solid-Geometry Methods 213
10-6 Octrees 215
References 217
Exercises 217

11 THREE-DIMENSIONAL TRANSFORMATIONS 220
11-1 Translation 221
11-2 Scaling 222
11-3 Rotation 223
11-4 Rotation About an Arbitrary Axis 224
Review of Vector Operations 225
Transformation Matrices 226
11-5 Other Transformations 229
Reflections 230
Shears 230
Transformation of Coordinate Systems 230
11-6 Transformation Commands 231
References 233
Exercises 233

12 THREE-DIMENSIONAL VIEWING 235
12-1 Projections 236
Parallel Projections 237
Perspective Projections 240

12-2 Viewing Transformation 241
Specifying the View Plane 242
View Volumes 246
Clipping 247
12-3 Implementation of Viewing Operations 248
Normalized View Volumes 248
Clipping Against a Normalized View Volume 251
12-4 Hardware Implementations 253
12-5 Programming Three-Dimensional Views 255
12-6 Extensions to the Viewing Pipeline 257
References 257
Exercises 257

13 HIDDEN-SURFACE AND HIDDEN-LINE REMOVAL 260
13-1 Classification of Algorithms 261
13-2 Back-Face Removal 261
13-3 Depth-Buffer Method 262
13-4 Scan-Line Method 264
13-5 Depth-Sorting Method 265
13-6 Area-Subdivision Method 268
13-7 Octree Methods 270
13-8 Comparison of Hidden-Surface Methods 272
13-9 Hidden-Line Elimination 273
13-10 Curved Surfaces 273
13-11 Hidden-Line and Hidden-Surface Command 274
References 274
Exercises 274

14 SHADING AND COLOR MODELS 276
14-1 Modeling Light Intensities 277
Light Sources 277
Diffuse Reflection 277
Specular Reflection 279
Refracted Light 281
Texture and Surface Patterns 283
Shadows 284

14-2 Displaying Light Intensities 284
Assigning Intensity Levels 285
Halftoning 286
14-3 Surface-Shading Methods 289
Constant Intensity 289
Gouraud Shading 289
Phong Shading 291
Ray-Tracing Algorithms 291
Octree Methods 293
Fractal Surfaces 294
Antialiasing Surface Boundaries 294
14-4 Color Models 295
Properties of Light 295
Standard Primaries and the Chromaticity Diagram 298
Intuitive Color Concepts 299
RGB Color Model 299
CMY Color Model 301
Conversion Between RGB and CMY Models 302
HSV Color Model 302
Conversion Between HSV and RGB Models 303
HLS Color Model 305
Color Selection 306
References 306
Exercises 307

15 MODELING METHODS 309
15-1 Basic Modeling Concepts 310
Model Representations 310
Symbol Hierarchies 311
Modeling Packages 313
15-2 Master Coordinates and Modeling Transformations 314
Modeling Transformations 315
Modeling Symbol Hierarchies 318
Display Procedures 321
15-3 Structured Display Files 322
15-4 Symbol Operations 322
15-5 Combining Modeling and Viewing Transformations 324
Master Coordinate Clipping 324
Bounding Rectangles for Symbols 325

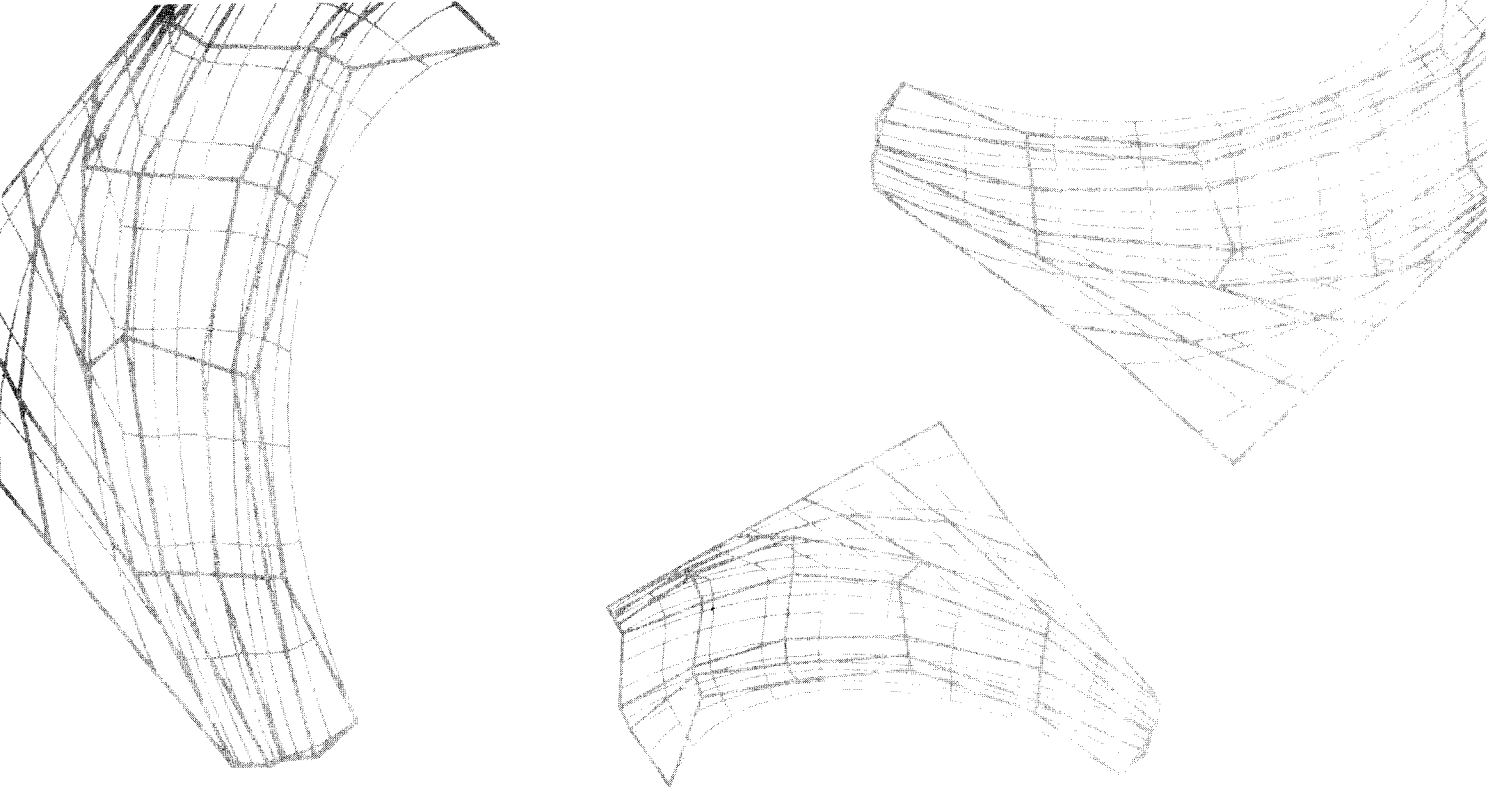
References 326
Exercises 326

16 ***DESIGN OF THE USER INTERFACE 328***
16-1 Components of a User Interface 329
16-2 User Model 329
16-3 Command Language 330
Minimizing Memorization 331
User Help Facilities 331
Backup and Error Handling 331
Response Time 332
Command Language Styles 332
16-4 Menu Design 334
16-5 Feedback 336
16-6 Output Formats 337
Icon and Symbol Shapes 337
Screen Layout 337
References 338
Exercises 339

BIBLIOGRAPHY 340

SUBJECT INDEX 347

FUNCTION INDEX 352



1 A SURVEY OF COMPUTER GRAPHICS

Computers have become a powerful tool for the rapid and economical production of pictures. There is virtually no area in which graphical displays cannot be used to some advantage, and so it is not surprising to find the use of computer graphics so widespread. Although early applications in engineering and science had to rely on expensive and cumbersome equipment, advances in computer technology have made interactive computer graphics a practical tool. Today, we find computer graphics used routinely in such diverse areas as business, industry, government, art, entertainment, advertising, education, research, training, and medicine. Figure 1–1 shows a few of the many ways that graphics is put to use. Our introduction to the field of computer graphics begins with a tour through a gallery of graphics applications.

The focusing system in a CRT is needed to force the electron beam to converge into a small spot as it strikes the phosphor. Otherwise, the electrons would repel each other, and the beam would spread out as it approaches the screen. Focusing is accomplished with either electric or magnetic fields. For electrostatic focusing, the electron beam passes through a metal cylinder with a positive voltage, as shown in Fig. 2–5. The positive voltage forces the electrons to stay along the axis of the beam. Similar focusing forces can be applied to the electron beam with electromagnetic fields set up by coils mounted around the outside of the CRT envelope.

Another type of focusing is used in high-precision systems to keep the beam in focus at all screen points. The distance that the electron beam must travel to different points on the screen varies because the radius of curvature for most CRTs is greater than the distance from the focusing system to the screen center. Therefore, the electron beam will be focused properly only at the center of the screen. As the beam moves to the outer edges of the screen, displayed images become blurred. To compensate for this, the system can adjust the focusing according to the screen position of the beam.

The maximum number of points that can be displayed without overlap on a CRT is referred to as the **resolution.** A more precise definition of resolution is the number of points per centimeter that can be plotted horizontally and vertically, although it is often simply stated as the total number of points in each direction. Resolution of a CRT is dependent on the type of phosphor used and the focusing and deflection systems. High-precision systems can display a maximum of about 4000 points in each direction, for a total of 16 million addressable screen points. Since a CRT monitor can be attached to different computer systems, the number of screen points that are utilized depends on the capabilities of the system to which it is attached.

An important property of video monitors is their **aspect ratio.** This number gives the ratio of vertical points to horizontal points necessary to produce equal-length lines in both directions on the screen. (Sometimes aspect ratio is stated in terms of the ratio of horizontal to vertical points.) An aspect ratio of 3/4 means that a vertical line plotted with three points has the same length as a horizontal line plotted with four points.

Random-Scan and Raster-Scan Monitors

Refresh CRTs can be operated either as random-scan or as raster-scan monitors. When operated as a **random-scan** display unit, a CRT has the electron beam directed only to the parts of the screen where a picture is to be drawn. Random-scan monitors draw a picture one line at a time and, for this reason, are also referred to as **vector** displays (or **stroke-writing** or **calligraphic** displays). The component lines of a picture can be drawn and refreshed by a random-scan system in any order specified (Fig. 2–6). A pen plotter operates in a similar way and is an example of a random-scan, hard-copy device.

Raster-scan video monitors shoot the electron beam over all parts of the screen, turning the beam intensity on and off to coincide with the picture definition. The picture is created on the screen as a set of points (Fig. 2–7), starting from the top of the screen. Definition for a picture is now stored as a set of intensity values for all the screen points, and these stored values are "painted" on the screen one row (scan line) at a time. The capability of a raster-scan system to store intensity information for each screen point makes it well suited for displaying shading

in storage space if a picture is to be constructed mostly with long runs of a single color each. A similar approach can be taken when pixel intensities change linearly. Another approach is to encode the raster as a set of rectangular areas (**cell encoding**). The disadvantages of encoding runs are that intensity changes are difficult to make and storage requirements actually increase as the length of the runs decreases. In addition, it is difficult for the display controller to process the raster when many short runs are involved.

2–5 Graphics Software

Programming commands for displaying and manipulating graphics output are designed as extensions to existing languages. An example of such a graphics package is the PLOT 10 system developed by Tektronix, Inc., for use with FORTRAN on their graphics terminals. Basic functions available in a package designed for the graphics programmer include those for generating picture components (straight lines, polygons, circles, and other figures), setting color and intensity values, selecting views, and applying transformations. By contrast, application graphics packages designed for nonprogrammers are set up so that users can produce graphics without worrying about how they do it. The interface to the graphics routines in such packages allows users to communicate with the programs in their own terms. Examples of such applications packages are the artist's painting programs and various business, medical, and CAD systems.

Coordinate Representations

Most graphics packages are designed to use Cartesian coordinate systems. More than one Cartesian system may be referenced by a package, since different output devices can require different coordinate systems. In addition, packages usually allow picture definitions to be set up in any Cartesian reference system convenient to the application at hand. The coordinates referenced by a user are called **world coordinates,** and the coordinates used by a particular output device are called **device coordinates,** or **screen coordinates** in the case of a video monitor. World coordinate definitions allow a user to set any convenient dimensions without being hampered by the constraints of a particular output device. Architectural layouts might be specified in fractions of a foot, while other applications might define coordinate scales in terms of millimeters, kilometers, or light-years. Once the world coordinate definitions are given, the graphics system converts these coordinates to the appropriate device coordinates for display.

A typical procedure used in graphics packages is first to convert world coordinate definitions to **normalized device coordinates** before final conversion to specific device coordinates. This makes the system flexible enough to accommodate a number of output devices (Fig. 2–31). Normalized x and y coordinates are each assigned values in the interval from 0 to 1. These normalized coordinates are then transformed to device coordinates (integers) within the range (0, 0) to (x_{max}, y_{max}) for a particular device. To accommodate differences in scales and aspect ratios, normalized coordinates can be mapped into a square area of the output device so that proper proportions are maintained. On a video monitor, the remaining area of the screen is often used to display messages or list interactive program options.

plications. Without standards, programs designed for one hardware system often cannot be transferred to another system without rewriting the software.

International and national standards-planning organizations in many countries have cooperated in an effort to develop a generally accepted standard for computer graphics. After considerable effort, this work on standards led to the development of the **Graphical Kernel System (GKS).** This system has been adopted as the graphics software standard by the International Standards Organization (ISO) and by various national standards organizations, such as the American National Standards Institute (ANSI). Although GKS was originally designed as a two-dimensional graphics package, a three-dimensional GKS extension was subsequently developed.

The final GKS functions, adopted as standards, were influenced by several earlier proposed graphics standards. Particularly important among these earlier proposals is the **Core Graphics System** (or simply **Core**), developed by the Graphics Standards Planning Committee of SIGGRAPH, the Special Interest Group on Computer Graphics of the Association for Computing Machinery (ACM).

Standard graphics functions are defined as a set of abstract specifications, independent of any programming language. To implement a graphics standard in a particular programming language, a **language binding** must be defined. This binding defines the syntax for accessing the various graphics functions specified within the standard. For example, GKS specifies a function to generate a sequence of connected straight line segments with the descriptive title

```
polyline (n, x, y)
```

In FORTRAN 77, this procedure is implemented as a subroutine with the name GPL. A graphics programmer, using FORTRAN, would invoke this procedure with the subroutine call statement

```
CALL GPL (N, X, Y)
```

GKS language bindings have been defined for FORTRAN, Pascal, Ada, C, PL/I, and COBOL. Each language binding is defined to make best use of the corresponding language capabilities and to handle various syntax issues, such as data types, parameter passing, and errors.

In the following chapters, we use the standard functions defined in GKS as a framework for discussing basic graphics concepts and the design and application of graphics packages. Example programs are presented in Pascal to illustrate the algorithms for implementation of the graphics functions and to illustrate also some applications of the functions. Descriptive names for functions, based on the GKS definitions, are used whenever a graphics function is referenced in a program.

Although GKS presents a specification for basic graphics functions, it does not provide a standard methodology for a graphics interface to output devices. Nor does it specify methods for real-time modeling or for storing and transmitting pictures. Separate standards have been developed for each of these three areas. Standardization for device interface methods is given in the **Computer Graphics Interface (CGI)** system. The **Computer Graphics Metafile (CGM)** system specifies standards for archiving and transporting pictures. And the **Programmer's Hierarchical Interactive Graphics Standard (PHIGS)** defines standard methods for real-time modeling and other higher-level programming capabilities not considered by GKS.

REFERENCES

A general treatment of display devices is available in Sherr (1979). The conceptual design of display devices is discussed in Haber and Wilkinson (1982) and in Myers (1984). Storage tubes are surveyed in Preiss (1978), and flat panel devices are dis-