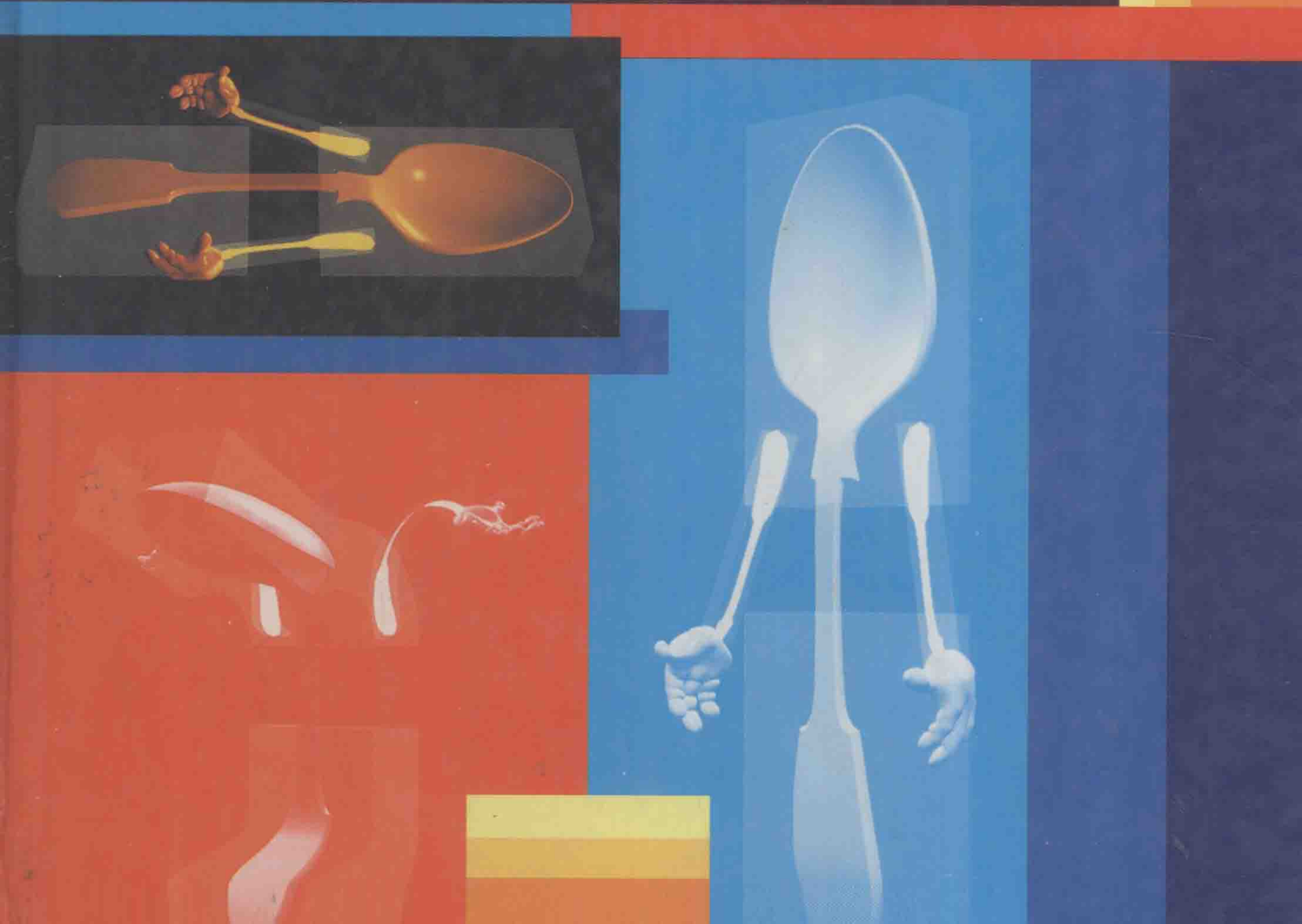
FUNDAMENTALS OF

THREE-DIMENSIONAL COMPUTER GRAPHICS

ALAN WATT



Fundamentals of Three-Dimensional Computer Graphics

Alan Watt
University of Sheffield

ADDISON-WESLEY PUBLISHING COMPANY
Wokingham, England · Reading, Massachusetts · Menlo Park, California
New York · Don Mills, Ontario · Amsterdam · Bonn
Sydney · Singapore · Tokyo · Madrid · San Juan

Cover designed by Crayon Design of Henley-on-Thames
using illustrations provided by Digital Pictures Ltd, London and
printed by The Riverside Printing Co. (Reading) Ltd.
Text designed by Lesley Stewart.
Typeset by Columns of Reading.
Printed and bound in Great Britain by The Bath Press, Avon.

First printed 1989.

British Library Cataloguing in Publication Data
Watt, Alan H.
Fundamentals of three-dimensional computer
graphics.
1. Computer systems. Graphic displays.
Three-dimensional images
I. Title
006.6

ISBN 0–201–15442–0

Library of Congress Cataloging in Publication Data
Watt, Alan H.
Fundamentals of three-dimensional computer graphics/Alan Watt.
p. cm.
Bibliography: p.
Includes index.
ISBN 0–201–15442–0
1. Computer graphics. 2. Three-dimensional display systems.
I. Title.
T385.W38 1989
006.6–dc20 89–34110
CIP

Fundamentals of Three-Dimensional Computer Graphics

Preface

This text is a comprehensive introduction to the techniques needed to produce shaded images of three-dimensional solids on a computer graphics monitor. The aim of the book is to give a theoretical understanding of these techniques together with the programming expertise required to implement them.

Three-dimensional computer graphics now embraces a large number of application areas, from the fantasy world of film and television to more practical areas such as CAD of mechanical engineering parts. In this sense three-dimensional graphics is possibly the most important aspect of computer graphics. While certain aspects are locked into particular areas – parametric surface techniques are used almost exclusively in engineering CAD – users as diverse as architects, molecular scientists and television animators use three-dimensional modelling and rendering techniques.

Many of the methods used in three-dimensional graphics are less than 10 years old and, while many excellent textbooks exist, these have mainly been general texts that have dealt with most of the mainstream topic areas in computer graphics. A good deal of the information needed in three-dimensional graphics is to be found only in research papers. This is particularly true of the techniques that have emerged in the last five years; for example, ray tracing and new reflection models. Implementing computer graphics methods from research papers is sometimes a difficult and tedious business. Details that are important to the implementer are quite rightly omitted from such publications, and would-be image synthesizers sometimes find themselves spending an inordinate time rediscovering the important practical tricks that are necessary to produce a rendered image.

Because computer graphics has spawned a large number of different methods, as the applications have grown and diversified, it is now almost impossible to write a comprehensive but standard-length text on the subject. These observations point to the need for more specialist texts that concentrate on important and unifying topics in the subject. This is such a text. Its aim is to deal with one of the mainstream areas of computer graphics and also to provide a level of detail important to implementers. In this respect, nearly all of the technique illustrations have been produced by the author using programs described in the text. Pascal procedures, implementing the crucial parts of methods at their final level of detail, are included. The inclusion of procedures also goes some way to solving the problem of algorithm and method description.

Pascal is chosen for two reasons:

(1) Although it is not the lingua franca that its adherents often claim, it is at least as good as a pseudo-code, and, with the exception of data structures, its translation into other high level languages is straightforward. (A language that has recently become fashionable in computer graphics is C. Although appreciating the practical advantages of C, it is not clear to your author that C offers any significant advantages, as a vehicle for describing algorithms, over Pascal.)

(2) It enables effective data structures to be implemented. The shading and representation of three-dimensional solids are inseparable topics and a text that deals with rendering should also deal, at some length, with three-dimensional data structures.

Currently the most popular hardware arrangement used in high quality rendering applications is a host processor that performs high level programming, driving a graphics terminal that is little more than a screen memory. The host is either in the same box, in which case the term graphics workstation is used, or it is in a separate enclosure and the graphics facility is called a terminal. (By 'high quality' we mean 24-bit screen memories; the most popular arrangement for 8-bit screen memories is the ubiquitous PC, enhanced with some extra hardware.)

For rendering shaded three-dimensional objects the programmer requires just one utility:

Write_Pixel (x, y, R, G, B)

which transfers a three component colour into location (x, y) in the screen memory. Some algorithms also require the 'opposite' utility:

Read_Pixel (x, y, R, G, B)

which reads the current value of a location in the screen memory into a program. This is an advantageous situation for authors of computer graphics textbooks because it means that programming illustrations can be almost completely device independent. Indeed, the programs given in this book will run on any host that accepts standard Pascal and will drive, with trivial alterations, any general-purpose graphics terminal. (Incidentally, since only one or two graphics utilities are used in the entire text the question of graphics standards does not arise. Furthermore, GKS-3D, the recently adopted graphics standard, does not have a convenient equivalent to *Write_Pixel*.)

The processing division between the host and graphics terminal processors is changing rapidly. Terminal utility packages are beginning to offer, for example, interpolative polygon shading and Z-buffer hidden surface removal. The trend is towards integrated workstations and away from the host terminal metaphor. This move is reflected in proposed extensions to the graphics standard PHIGS. PHIGS+ is intended to reflect this trend and includes lighting and shading as well as new three-dimensional primitives.

Efficiency in computer graphics is an important topic. Studios producing three-dimensional animated sequences always have a fixed generation time limit per frame that is imposed by the length of the sequence, the hardware available and the rendering technique used (not to mention the budget). In this text the program procedures are written to illustrate techniques and although efficiency considerations are touched on from time to time, it is difficult to combine such factors with sound theoretical explanations. Efficiency in graphics programming is extremely context dependent and involves such factors as rendering at the lowest appropriate spatial resolution, varying the rendering resolution as a function of the size of the object, using the least expensive reflection model that is appropriate, writing critical code in assembly language and increasing the workload of the graphics terminal processor by loading heavily used routines into it. Although it has been tried, in the text, to avoid gross abuse of processing time, efficiency is generally sacrificed for programming transparency. Efficiency in Phong shading is treated as a topic in its own right.

A significant proportion of the text is devoted to reflection or shading models. The current thrust of research in this area is towards greater and greater realism. Reflection models are continually being refined. The oft-stated goal of such efforts is to produce images that are indistinguishable from images obtained from, say, a television camera. The price that is always paid for more accurate reflection models is, of course, significant increases in computation time. This may not be such a vital consideration in the future, with the continuing decrease in the cost of processing units, but the relentless pursuit of reality in computer graphics does seem to be, in one sense, aimless.

The explosive demand for processor power in computer graphics is also driven by the related factor of increased resolution. Not only do algorithms increase their complexity, but at the same time the spatial resolution of graphics terminals increases, resulting in resource requirements that tend to obey a growth law with an index somewhere between 2 and 3.

One of the major application areas of computer graphics is in the entertainment and advertising industry. Here accurate reality is not necessarily a desirable goal. The appeal of computer graphics in a television commercial or title sequence is the super-reality or computer signature of the images. Images are striking partly because they are recognizably computer generated. To 'reduce' this effect by increasing their authenticity is to reduce their appeal and utility. After all, it is much easier to digitize real scenes using a television camera. In this respect a recent trend is to mix video and computer graphics images.

The treatment in this text of reflection models is restricted to the commonly used methods, although the more advanced and recent methods are not totally ignored. Reflection models are introduced in a sequence that reflects both their historical emergence and their complexity. This is certainly not the best approach from the viewpoint of mathematical propriety and a mathematical unification and discussion of reflection models is given in Appendix E. This can be safely ignored by the reader interested only in implementing a particular method, rather than understanding the mathematical relationship of the model to physical reality.

One of the main aims of the book is to provide implementation details and this means selecting a particular method to implement each part of the rendering process, rather than cataloguing processes and briefly describing each. Hidden surface removal, for example, is a good illustration of a stage in rendering where a choice of algorithms is possible. Where there is such a choice the selection of technique has been motivated by its popularity, acceptance and ease of implementation. In most cases these aims do not conflict with programming efficiency.

Nearly all the screen images in the text have been produced using the basic software described in the text. Some of the images are based on simple and effective ideas or illustrations produced by other computer graphics workers. In such cases accreditation is given to the producer of the original image.

The chapter structure enables the book to be read in any order. Most chapters are self contained. Anti-aliasing is dealt with both in a separate chapter, where the underlying theory is discussed, and in Chapters 7, 10 and 13.

Finally, although the book aims at being a 'how to do it' manual rather than a 'how it has been done' book, there are certain topics which, because of their nature, resist this approach. In particular, the chapter on three-dimensional animation is a description of the major approaches to this topic.

Programs, teaching and learning

The code in this book is in one of three forms; Pascal pseudo-code for the algorithms where this seems appropriate, Pascal procedures for all other algorithms and a complete rendering system in Appendix B and Appendix C. This consists of a wireframe program complete with a viewpoint interface and a comprehensive data structure. Appendix C contains the additional procedures required for a Z-buffer based rendering system with Gouraud shading. The data for the Utah teapot is reproduced in Appendix D – even novice graphics programmers tire of cubes and spheres.

Appendices A, B and C are intended to be an integral part of the book that can be used for more detailed further study. In particular, Appendix A is an example of the use of three-dimensional linear transformations. Appendix B is a study of data structures, wireframe drawing and general viewing systems. Appendix C extends the second appendix with rendering techniques described in Chapter 5.

The software is designed to be a learning aid to the techniques in the text and, with the single exception of ray tracing, all techniques can be grafted onto the supplied rendering system.

If the text is used for teaching or self-study, the student can be supplied with the rendering software to gain basic viewpoint experience, and the techniques covered in each chapter used as exercises. All the procedures have been integrated into the basic rendering system and in most cases no modifications are required to the data structure. Except where otherwise stated all the colour pictures were produced by this software.

Projects, notes and suggestions

The projects are meant to be an integral part of the text and can be usefully read, even if you do not implement them. Many important points are examined that expand on topics in the chapters. Implementation points that are too detailed to be covered in the chapters are dealt with in the context of a project. Some useful background information is also given – Gauss–Seidel in the context of radiosity and Fourier theory for anti-aliasing.

Each project is classified with a heading. Projects marked (*) are fairly lengthy, contain some scope for original work and could be used as a major course assignment.

Have fun.

Acknowledgements

It would have been difficult to have produced this book without the prodigious contributions made by my research students; Keith Harrison, Dave Mitchell and Steve Maddock. They wrote the code, produced the line plotter illustrations and generated all the original colour plates. I am indebted to them for many useful suggestions and criticisms and for the donation of their time. Dave made extensive contributions to Chapter 5 and he and Keith developed the H-test. Steve assisted with Chapter 6. My colleague, Jim McGregor, wrote a recursive ray tracer as an example of recursion for undergraduates and contributed to Chapter 8. Sometime undergraduates – Andy Price and Nigel Rasberry developed the ASL animation project. Bill Sproson (formerly of the BBC, and author of the text *Colour Science in Television and Display Systems*) checked Chapter 14.

But I should mention my collaborators at Addison-Wesley. Sarah Mallen has been a source of encouragement in a world of slipped deadlines; the production staff and Lynne Balfe coped with the usual tiresome preproduction problems and endless author alterations with great efficiency and cheerfulness; and finally, Kathryn Kergozou – the world would be a duller place without her sox.

Alan Watt

University of Sheffield
August 1989

Contents

Preface		vii
Chapter One	**Basic three-dimensional theory**	1
1.1	Manipulating three-dimensional structures	1
1.2	The basics: linear transformations	2
1.3	Structure-deforming transformations	10
1.4	Projecting three-dimensional objects onto a view plane	13
1.5	Viewing systems	16
1.6	Three-dimensional viewing pipeline	26
1.7	An aside on graphics languages	29
1.8	Polygon mesh models and wireframe representation	31
	Projects, notes and suggestions	38
Chapter Two	**A basic reflection model**	45
2.1	Simple reflection – the Phong model	46
2.2	Diffuse reflection	48
2.3	Ambient light	49
2.4	Distance	50
2.5	Specular reflection	50
2.6	Geometric considerations	53
2.7	Colour	55
2.8	Summary of the Phong model	55
2.9	The 20 spheres – an example	56
2.10	Using look-up tables with reflection models	59
2.11	Empirical transparency	62
	Projects, notes and suggestions	63

Chapter Three A more advanced reflection model 65
3.1 The Cook and Torrance model 65
3.2 Illumination source models 78
Projects, notes and suggestions 81

Chapter Four Incremental shading techniques 83
4.1 Gouraud shading 84
4.2 Phong interpolation 87
4.3 Comparison of Gouraud and Phong shading 89
4.4 Speeding up Phong shading 90
Projects, notes and suggestions 93

Chapter Five The rendering process 97
5.1 Rasterization 98
5.2 Order of rendering 102
5.3 Hidden surface removal 104
5.4 Compositing three-dimensional images 111
Projects, notes and suggestions 112

Chapter Six Parametric representation of three-dimensional objects 115
6.1 Parametric representation of three-dimensional curves 116
6.2 Parametric representation of three-dimensional surfaces 127
6.3 Scan converting parametric surfaces 139
Projects, notes and suggestions 147

Chapter Seven Ray tracing 151
7.1 Basic algorithm 152
7.2 A historical digression – the optics of the rainbow 154
7.3 Recursive implementation of ray tracing 156
7.4 A remark on efficiency 161
7.5 Ray tracing geometry – intersections 162
7.6 Ray tracing geometry – reflection and refraction 165
7.7 Reflection–illumination model 167
7.8 Shadows and ray tracing 173
7.9 Distributed ray tracing 174
7.10 Ray tracing and anti-aliasing operations 176
Projects, notes and suggestions 178

Chapter Eight Advanced ray tracing 181
8.1 Adaptive depth control 181
8.2 Bounding volume extensions 184
8.3 First-hit speed up 187
8.4 Spatial coherence 187
8.5 Data structures for ray tracing: octrees 190
8.6 Data structures for ray tracing: BSP trees 195
Projects, notes and suggestions 198

Chapter Nine Diffuse illumination and the development of the radiosity method 201
9.1 Radiosity theory 202
9.2 Form factor determination 205
9.3 Further development of the radiosity method 209
Projects, notes and suggestions 216

Chapter Ten Further realism: shadows, texture and environment mapping 219
10.1 Shadows 219
10.2 Texture 227
10.3 Environment mapping 247
Projects, notes and suggestions 251

Chapter Eleven Functionally based modelling 255
11.1 Particle systems 256
11.2 Fractal systems 257
11.3 Functions suitable for three-dimensional texture 261
Projects, notes and suggestions 264

Chapter Twelve Anti-aliasing techniques 267
12.1 Aliasing artefacts and Fourier theory 269
12.2 Supersampling or postfiltering 273
12.3 Prefiltering or area sampling techniques 275
12.4 A mathematical comparison 277
12.5 Stochastic sampling 278
Projects, notes and suggestions 281

Chapter Thirteen Three-dimensional animation 285
13.1 Approaches: three-dimensional key frame systems 288
13.2 Approaches: parametric systems 291
13.3 Approaches: programmed animation and scripting systems 292
13.4 Approaches: simulated or model-driven systems 295
13.5 Temporal anti-aliasing 300
Projects, notes and suggestions 301

Chapter Fourteen Colour science and computer graphics 309
14.1 Applications of colour in computer graphics 311
14.2 Monitor models 313
14.3 Television transmission spaces 315
14.4 Colour models 318
14.5 Basic colorimetry concepts 331
14.6 The CIE standard 339
14.7 Realistic rendering and reflection models 353
Projects, notes and suggestions 354

Appendix A Viewing transformation for a simple four-parameter viewing system 357

Appendix B A wireframe system 363
B.1 Introduction 363
B.2 Data structure 364
B.3 Program 371

Appendix C An implementation of a renderer 389

Appendix D The Utah teapot 395

Appendix E Some theoretical concepts 399
E.1 Introduction 399
E.2 Useful definitions 400
E.3 Hall's model 400
E.4 The rendering equation 403

Appendix F Highlight detection – the H test 405
F.1 Introduction 405
F.2 The tests 407
F.3 Example timings 410

References 413

Index 421

CHAPTER ONE

Basic Three-Dimensional Theory

Linear transformations are important tools in generating three-dimensional scenes. They are used to move objects around in an environment, and also to construct a two-dimensional view of the environment for a display surface. This chapter deals with basic three-dimensional transformations, introduces some useful shape-changing transformations, looks at viewing techniques and considers techniques for representing and displaying a wireframe of an object.

1.1 Manipulating three-dimensional structures

In computer graphics the most popular method for representing an object is the polygon mesh model. This form of representation is either exact or an approximation depending on the nature of the object. A cube, for example, can be represented exactly by six squares. A cylinder, in contrast, can only be approximated by polygons; say six rectangles for the curved surface and two hexagons for the end faces. The number of polygons used in the approximation determines how accurately the object is represented and this has repercussions in modelling cost, storage and rendering cost and quality. The popularity of the polygon mesh modelling technique in computer graphics is undoubtedly due to its inherent simplicity and the development of inexpensive shading algorithms that work with such models.

A polygon mesh model consists of a structure of vertices, each vertex being a three-dimensional point in so-called world coordinate space or definition space. Later we shall be concerned with how vertices are connected to form polygons and how polygons are structured into complete objects, but to start with we shall consider objects just as a set of three-dimensional vertices and look at how these are transformed in three-dimensional space using linear transformations.

1.2 The basics: linear transformations

Objects are defined in a world coordinate system which is conventionally a right-handed system. Right-handed and left-handed three-dimensional coordinate systems are shown in Figure 1.1. As we shall see later, objects in the right-handed world coordinate system are transformed into a left-handed view plane and view surface coordinate system.

It is sometimes convenient to define objects in their own local coordinate system. There are three reasons for this. When a three-dimensional object is modelled it is useful to build up the vertices with respect to some reference point in the object. In fact a complex object may have a number of local coordinate systems, one for each subpart. It may be that the same object is to appear many times in a scene and a definition with a local origin is the only sensible way to set this up. Instancing an object by applying a mix of translations, rotation and scaling transformations can then be seen as transforming the local coordinate system of each object to the world coordinate system. Finally, when an object is to be rotated, it is easier if the rotation is defined with respect to a local reference such as an axis of symmetry. (This philosophy is adopted, for example, in the graphics standard PHIGS where the programmer defines structures in the modelling coordinate system. Modelling transformations define the mapping from this coordinate space to world coordinate space.)

A set of vertices or three-dimensional points belonging to an object can be transformed into another set of points by a linear transformation.

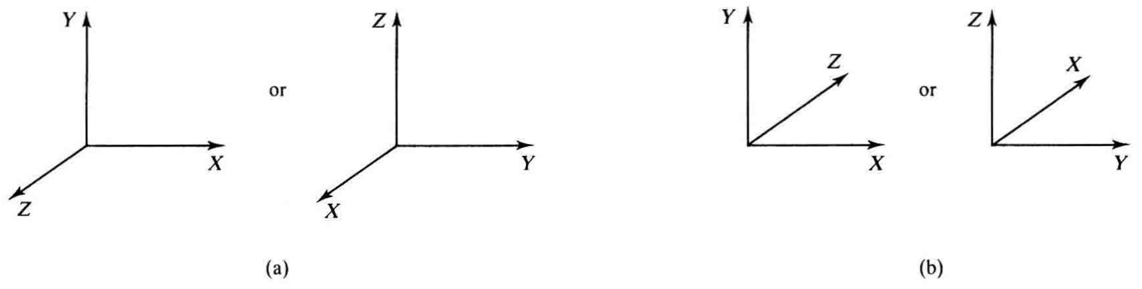


Figure 1.1
(a) Right-handed coordinate system and (b) left-handed system.