3-D Computer Graphics

A Mathematical Introduction with OpenGL

This book is an introduction to 3-D computer graphics with particular emphasis on fundamentals and the mathematics underlying computer graphics. It includes descriptions of how to use the cross-platform OpenGL programming environment. It also includes source code for a ray tracing software package. (Accompanying software is available freely from the book's Web site.)

Topics include a thorough treatment of transformations and viewing, lighting and shading models, interpolation and averaging, Bézier curves and B-splines, ray tracing and radiosity, and intersection testing with rays. Additional topics, covered in less depth, include texture mapping and color theory. The book also covers some aspects of animation, including quaternions, orientation, and inverse kinematics. Mathematical background on vectors and matrices is reviewed in an appendix.

This book is aimed at the advanced undergraduate level or introductory graduate level and can also be used for self-study. Prerequisites include basic knowledge of calculus and vectors. The OpenGL programming portions require knowledge of programming in C or C++. The more important features of OpenGL are covered in the book, but it is intended to be used in conjunction with another OpenGL programming book.

Samuel R. Buss is Professor of Mathematics and Computer Science at the University of California, San Diego. With both academic and industrial expertise, Buss has more than 60 publications in the fields of computer science and mathematical logic. He is the editor of several journals and the author of a book on bounded arithmetic. Buss has years of experience in programming and game development and has acted as consultant for SAIC and Angel Studios.

3-D Computer Graphics

A Mathematical Introduction with OpenGL

SAMUEL R. BUSS
University of California, San Diego

CAMBRIDGE
UNIVERSITY PRESS

PUBLISHED BY THE PRESS SYNDICATE OF THE UNIVERSITY OF CAMBRIDGE
The Pitt Building, Trumpington Street, Cambridge, United Kingdom

CAMBRIDGE UNIVERSITY PRESS
The Edinburgh Building, Cambridge CB2 2RU, UK
40 West 20th Street, New York, NY 10011-4211, USA
477 Williamstown Road, Port Melbourne, VIC 3207, Australia
Ruiz de Alarcón 13, 28014 Madrid, Spain
Dock House, The Waterfront, Cape Town 8001, South Africa

http://www.cambridge.org

First published 2003
Reprinted with corrections 2004

Printed in the United States of America

Typefaces Times New Roman PS 10/12 pt. *System* LaTeX 2ε [TB]

A catalog record for this book is available from the British Library.

Library of Congress Cataloging in Publication Data

Buss, Samuel R.
3D computer graphics : a mathematical introduction with OpenGL / Samuel R. Buss.
p. cm.
Includes bibliographical references and index.
ISBN 0-521-82103-7
1. Computer graphics. 2. OpenGL. 3. Three-dimensional display systems. I. Title.
T385 .B8695 2003

006.6′93 – dc21 2002034804

ISBN 0 521 82103 7 hardback

To my family

Teresa, Stephanie, and Ian

Contents

Color art appears following page 256.

Preface

Computer graphics has grown phenomenally in recent decades, progressing from simple 2-D graphics to complex, high-quality, three-dimensional environments. In entertainment, computer graphics is used extensively in movies and computer games. Animated movies are increasingly being made entirely with computers. Even nonanimated movies depend heavily on computer graphics to develop special effects: witness, for instance, the success of the *Star Wars* movies beginning in the mid-1970s. The capabilities of computer graphics in personal computers and home game consoles have now improved to the extent that low-cost systems are able to display millions of polygons per second.

There are also significant uses of computer graphics in nonentertainment applications. For example, virtual reality systems are often used in training. Computer graphics is an indispensable tool for scientific visualization and for computer-aided design (CAD). We need good methods for displaying large data sets comprehensibly and for showing the results of large-scale scientific simulations.

The art and science of computer graphics have been evolving since the advent of computers and started in earnest in the early 1960s. Since then, computer graphics has developed into a rich, deep, and coherent field. The aim of this book is to present the mathematical foundations of computer graphics along with a practical introduction to programming using OpenGL. I believe that understanding the mathematical basis is important for any advanced use of computer graphics. For this reason, this book attempts to cover the underlying mathematics thoroughly. The principle guiding the selection of topics for this book has been to choose topics that are of practical significance for computer graphics practitioners – in particular for software developers. My hope is that this book will serve as a comprehensive introduction to the standard tools used in this field and especially to the mathematical theory behind these tools.

About This Book

The plan for this book has been shaped by my personal experiences as an academic mathematician and by my participation in various applied computer projects, including projects in computer games and virtual reality. This book was started while I was teaching a mathematics class at the University of California, San Diego (UCSD), on computer graphics and geometry. That course was structured as an introduction to programming 3-D graphics in OpenGL and to the mathematical foundations of computer graphics. While teaching that course, I became convinced of the need for a book that would bring together the mathematical theory underlying computer graphics in an introductory and unified setting.

The other motivation for writing this book has been my involvement in several virtual reality and computer game projects. Many of the topics included in this book are presented mainly because I have found them useful in computer game applications. Modern-day computer games and virtual reality applications are technically demanding software projects: these applications require software capable of displaying convincing three-dimensional environments. Generally, the software must keep track of the motion of multiple objects; maintain information about the lighting, colors, and textures of many objects; and display these objects on the screen at 30 or 60 frames per second. In addition, considerable artistic and creative skills are needed to make a worthwhile three-dimensional environment. Not surprisingly, this requires sophisticated software development by large teams of programmers, artists, and designers.

Perhaps it is a little more surprising that 3-D computer graphics requires extensive mathematics. This is, however, the case. Furthermore, the mathematics tends to be elegant and interdisciplinary. The mathematics needed in computer graphics brings together constructions and methods from several areas, including geometry, calculus, linear algebra, numerical analysis, abstract algebra, data structures, and algorithms. In fact, computer graphics is arguably the best example of a practical area in which so much mathematics combines so elegantly.

This book presents a blend of applied and theoretical topics. On the more applied side, I recommend the use of OpenGL, a readily available, free, cross-platform programming environment for 3-D graphics. The `C` and `C++` code for OpenGL programs that can freely be downloaded from the Internet has been included, and I discuss how OpenGL implements many of the mathematical concepts discussed in this book. A ray tracer software package is also described; this software can also be downloaded from the Internet. On the theoretical side, this book stresses the mathematical foundations of computer graphics, more so than any other text of which I am aware. I strongly believe that knowing the mathematical foundations of computer graphics is important for being able to use tools such as OpenGL or Direct3D, or, to a lesser extent, CAD programs properly.

The mathematical topics in this book are chosen because of their importance and relevance to graphics. However, I have not hesitated to introduce more abstract concepts when they are crucial to computer graphics – for instance, the projective geometry interpretation of homogeneous coordinates. A good knowledge of mathematics is invaluable if you want to use the techniques of computer graphics software properly and is even more important if you want to develop new or innovative uses of computer graphics.

How to Use This Book

This book is intended for use as a textbook, as a source for self-study, or as a reference. It is strongly recommended that you try running the programs supplied with the book and write some OpenGL programs of your own. Note that this book is intended to be read in conjunction with a book on learning to program in OpenGL. A good source for learning OpenGL is the comprehensive *OpenGL Programming Guide* (Woo et al., 1999), which is sometimes called the "red book." If you are learning OpenGL on your own for the first time, the *OpenGL Programming Guide* may be a bit daunting. If so, the *OpenGL SuperBible* (Wright Jr., 1999) may provide an easier introduction to OpenGL with much less mathematics. The book *OpenGL: A Primer* (Angel, 2002) also gives a good introductory overview of OpenGL.

The outline of this book is as follows. The chapters are arranged more or less in the order the material might be covered in a course. However, it is not necessary to read the material in order. In particular, the later chapters can be read largely independently, with the exception that Chapter VIII depends on Chapter VII.

Chapter I. Introduction. Introduces the basic concepts of computer graphics; drawing points, lines, and polygons; modeling with polygons; animation; and getting started with OpenGL programming.

Chapter II. Transformations and Viewing. Discusses the rendering pipeline, linear and affine transformations, matrices in two and three dimensions, translations and rotations, homogeneous coordinates, transformations in OpenGL, viewing with orthographic and perspective transformations, projective geometry, pixelization, Gouraud and scan line interpolation, and the Bresenham algorithm.

Chapter III. Lighting, Illumination, and Shading. Addresses the Phong lighting model; ambient, diffuse, and specular lighting; lights and material properties in OpenGL; and the Cook–Torrance model.

Chapter IV. Averaging and Interpolation. Presents linear interpolation, barycentric coordinates, bilinear interpolation, convexity, hyperbolic interpolation, and spherical linear interpolation. This is a more mathematical chapter with many tools that are used elsewhere in the book. You may wish to skip much of this chapter on the first reading and come back to it as needed.

Chapter V. Texture Mapping. Discusses textures and texture coordinates, mipmapping, supersampling and jittering, bump mapping, environment mapping, and texture maps in OpenGL.

Chapter VI. Color. Addresses color perception, additive and subtractive colors, and RGB and HSL representations of color.

Chapter VII. Bézier Curves. Presents Bézier curves of degree three and of general degree; De Casteljau methods; subdivision; piecewise Bézier curves; Hermite polynomials; Bézier surface patches; Bézier curves in OpenGL; rational curves and conic sections; surfaces of revolution; degree elevation; interpolation with Catmull–Rom, Bessel–Overhauser, and tension-continuity-bias splines; and interpolation with Bézier surfaces.

Chapter VIII. B-Splines. Describes uniform and nonuniform B-splines and their properties, B-splines in OpenGL, the de Boor algorithm, blossoms, smoothness properties, rational B-splines (NURBS) and conic sections, knot insertion, relationship with Bézier curves, and interpolation with spline curves. This chapter has a mixture of introductory topics and more specialized topics. We include all proofs but recommend that many of the proofs be skipped on the first reading.

Chapter IX. Ray Tracing. Presents recursive ray tracing, reflection and transmission, distributed ray tracing, backwards ray tracing, and cheats to avoid ray tracing.

Chapter X. Intersection Testing. Describes testing rays for intersections with spheres, planes, triangles, polytopes, and other surfaces and addresses bounding volumes and hierarchical pruning.

Chapter XI. Radiosity. Presents patches, form factors, and the radiosity equation; the hemicube method; and the Jacobi, Gauss–Seidel, and Southwell iterative methods.

Chapter XII. Animation and Kinematics. Discusses key framing, ease in and ease out, representations of orientation, quaternions, interpolating quaternions, and forward and inverse kinematics for articulated rigid multibodies.

Appendix A. Mathematics Background. Reviews topics from vectors, matrices, linear algebra, and calculus.

Appendix B. RayTrace Software Package. Describes a ray tracing software package. The software is freely downloadable.

Exercises are scattered throughout the book, especially in the more introductory chapters. These are often supplied with hints, and they should not be terribly difficult. It is highly recommended that you do the exercises to master the material. A few sections in the book, as well as some of the theorems, proofs, and exercises, are labeled with an asterisk (★). This indicates that the material is optional, less important, or both and can be safely skipped without affecting your understanding of the rest of the book. Theorems, lemmas, figures, and exercises are numbered separately for each chapter.

Obtaining the Accompanying Software

All software examples discussed in this book are available for downloading from the Internet at

`http://math.ucsd.edu/~sbuss/MathCG/.`

The software is available as source files and as PC executables. In addition, complete Microsoft Visual C++ project files are available.

The software includes several small OpenGL programs and a relatively large ray tracing software package.

The software may be used without any restriction except that its use in commercial products or any kind of substantial project must be acknowledged.

Getting Started with OpenGL

OpenGL is a platform-independent API (application programming interface) for rendering 3-D graphics. A big advantage of using OpenGL is that it is a widely supported industry standard. Other 3-D environments, notably Direct3D, have similar capabilities; however, Direct3D is specific to the Microsoft Windows operating system.

The official OpenGL Web site is `http://www.opengl.org`. This site contains a huge amount of material, but if you are just starting to learn OpenGL the most useful material is probably the tutorials and code samples available at

`http://www.opengl.org/developers/code/tutorials.html.`

The OpenGL programs supplied with this text use the OpenGL Utility Toolkit routines, called GLUT for short, which is widely used and provides a simple-to-use interface for controlling OpenGL windows and handling simple user input. You generally need to install the GLUT files separately from the rest of the OpenGL files.

If you are programming with Microsoft Visual C++, then the OpenGL header files and libraries are included with Visual C++. However, you will need to download the GLUT files yourself. OpenGL can also be used with other development environments such as Borland's C++ compiler.

The official Web site for downloading the latest version of GLUT for the Windows operating system is available from Nate Robin at

`http://www.xmission.com/~nate/glut.html.`

To install the necessary GLUT files on a Windows machine, you should put the header file `glut.h` in the same directory as your other OpenGL header files such as `glu.h`. You should likewise put the `glut32.dll` files and `glut32.lib` file in the same directories as the corresponding files for OpenGL, `glu32.dll`, and `glu32.lib`.

OpenGL and GLUT work under a variety of other operating systems as well. I have not tried out all these systems but list some of the prominent ones as an aid to the reader trying to run OpenGL in other environments. (However, changes occur rapidly in the software development world, and so these links may become outdated quickly.)

For Macintosh computers, you can find information about OpenGL and the GLUT libraries at the Apple Computer site

`http://developer.apple.com/opengl/`.

OpenGL and GLUT also work under the Cygwin system, which implements a Unix-like development environment under Windows. Information on Cygwin is available at `http://cygwin.com/` or `http://sources.redhat.com/cygwin/`.

OpenGL for Sun Solaris systems can be obtained from

`http://www.sun.com/software/graphics/OpenGL/`.

There is an OpenGL-compatible system, `Mesa3D`, which is available from `http://mesa3d.sourceforge.net/`. This runs on several operating systems, including Linux, and supports a variety of graphics boards.

Other Resources for Computer Graphics

You may wish to supplement this book with other sources of information on computer graphics. One rather comprehensive textbook is the volume by Foley et al. (1990). Another excellent recent book is Möller and Haines (1999). The articles by Blinn (1996; 1998) and Glassner (1999) are also interesting.

Finally, an enormous amount of information about computer graphics theory and practice is available on the Internet. There you can find examples of OpenGL programs and information about graphics hardware as well as theoretical and mathematical developments. Much of this can be found through your favorite search engine, but you may also use the ACM *Transactions on Graphics* Web site `http://www.acm.org/tog/` as a starting point.

For the Instructor

This book is intended for use with advanced junior- or senior-level undergraduate courses or introductory graduate-level courses. It is based in large part on my teaching of computer graphics courses at the upper division level and at the graduate level. In a two-quarter undergraduate course, I cover most of the material in the book more or less in the order presented here. Some of the more advanced topics would be skipped, however – most notably Cook–Torrance lighting and hyperbolic interpolation – and some of the material on Bézier and B-spline curves and patches is best omitted from an undergraduate course. I also do not cover the more difficult proofs in undergraduate courses.

It is certainly recommended that students studying this book get programming assignments using OpenGL. Although this book covers much OpenGL material in outline form, students will need to have an additional source for learning the details of programming in OpenGL. Programming prerequisites include some experience in C, C++, or Java. (As we write this, there is no standardized OpenGL API for Java; however, Java is close enough to C or C++ that students can readily make the transition required for mastering the simple programs included with this text.) The first quarters of my own courses have included programming assignments first on two-dimensional graphing, second on three-dimensional transformations based on the solar system exercise on page 40, third on polygonal modeling (students are asked to draw tori

of the type in Figure I.11(b)), fourth on adding materials and lighting to a scene, and finally an open-ended assignment in which students choose a project of their own. The second quarter of the course has included assignments on modeling objects with Bézier patches (Blinn's article (1987) on how to construct the Utah teapot is used to help with this), on writing a program that draws Catmull–Rom and Overhauser spline curves that interpolate points picked with the mouse, on using the computer-aided design program *3D Studio Max* (this book does not cover any material about how to use CAD programs), on using the ray tracing software supplied with this book, on implementing some aspect of distributed ray tracing, and then ending with another final project of their choosing. Past course materials can be found on the Web from my home page `http://math.ucsd.edu/~sbuss/`.

Acknowledgments

Very little of the material in this book is original. The aspects that are original mostly concern organization and presentation: in several places, I have tried to present new, simpler proofs than those known before. Frequently, material is presented without attribution or credit, but in most instances this material is due to others. I have included references for items I learned by consulting the original literature and for topics for which it was easy to ascertain the original source; however, I have not tried to be comprehensive in assigning credit.

I learned computer graphics from several sources. First, I worked on a computer graphics project with several people at SAIC, including Tom Yonkman and my wife, Teresa Buss. Subsequently, I have worked for many years on computer games applications at Angel Studios, where I benefited greatly, and learned an immense amount, from Steve Rotenberg, Brad Hunt, Dave Etherton, Santi Bacerra, Nathan Brown, Ted Carson, Jeff Roorda, Daniel Blumenthal, and others. I am particularly indebted to Steve Rotenberg, who has been my guru for advanced topics and current research in computer graphics.

I have taught computer graphics courses several times at UCSD, using at various times the textbooks by Watt and Watt (1992), Watt (1993), and Hill (2001). This book was written from notes developed while teaching these classes.

I am greatly indebted to Frank Chang and Malachi Pust for a thorough proofreading of an early draft of this book. In addition, I thank Michael Bailey, Stephanie Buss (my daughter), Chris Calabro, Joseph Chow, Daniel Curtis, Tamsen Dunn, Rosalie Iemhoff, Cyrus Jam, Jin-Su Kim, Vivek Manpuria, Jason McAuliffe, Jong-Won Oh, Horng Bin Ou, Chris Pollett, John Rapp, Don Quach, Daryl Sterling, Aubin Whitley, and anonymous referees for corrections to preliminary drafts of this book and Tak Chu, Craig Donner, Jason Eng, Igor Kaplounenko, Alex Kulungowski, Allen Lam, Peter Olcott, Nevin Shenoy, Mara Silva, Abbie Whynot, and George Yue for corrections incorporated into the second printing. Further thanks are due to Cambridge University Press for copyediting and final typesetting. As much as I would like to avoid it, the responsibility for all remaining errors is my own.

The figures in this book were prepared with several software systems. The majority of the figures were created using van Zandt's `pstricks` macro package for LaTeX. Some of the figures were created with a modified version of Geuzaine's program `GL2PS` for converting OpenGL images into PostScript files. A few figures were created from screen dump bitmaps and converted to PostScript images with Adobe Photoshop.

Partial financial support was provided by National Science Foundation grants DMS-9803515 and DMS-0100589.