PREFACE

NEW TO THE SIXTH EDITION

- Use of modern Shader-Based OpenGL throughout
- No use of OpenGL functions deprecated with OpenGL 3.1
- Increased detail on implementing transformations and viewing in both application code and shaders
- Consistency with OpenGL ES 2.0 and WebGL
- Use of C++ instead of C
- Addition of vector and matrix classes to create application code compatible with the OpenGL Shading Language (GLSL)
- Discussion of per-vertex and per-fragment lighting
- Addition of polygon and Delaunay triangularization
- Introduction to volume rendering
- All code examples redone to be compatible with OpenGL 3.1
- New co-author, Dave Shreiner, author of the *OpenGL Programming Guide*

This book is an introduction to computer graphics, with an emphasis on applications programming. The first edition, which was published in 1997, was somewhat revolutionary in using a standard graphics library and a top-down approach. Over the succeeding 13 years and five editions, this approach has been adopted by most introductory classes in computer graphics and by virtually all the competing textbooks.

The major changes in graphics hardware over the past few years have led to major changes in graphics software. For its first fifteen years, new OpenGL versions were released with new versions always guaranteeing backward compatibility. The ability to reuse code as the underlying software was upgraded was an important virtue, both for developers of applications and for instructors of graphics classes. OpenGL 3.0 announced major changes, one of the key ones being that, starting with OpenGL 3.1, many of the most common functions would be deprecated. This radical departure from previous versions reflects changes needed to make use of the capabilities of the

latest programmable graphics units (GPUs). These changes are also part of OpenGL ES 2.0, which is being used to develop applications on mobile devices such as cell phones and tablets, and WebGL, which is supported on most of the latest browsers.

As the authors of the previous five editions of this textbook (EA) and of the *OpenGL Programming Guide* and *OpenGL ES 2.0 Programming Guide* (DS), we were confronted with a dilemma as to how to react to these changes. We had been writing books and teaching introductory OpenGL courses at SIGGRAPH for many years. We found that almost no one in the academic community, or application programmers outside the high-end game world, knew about these changes, and those of our colleagues who did know about them did not think we could teach these concepts at the beginning level. That was a challenge we couldn't resist. We started by teaching a half-day short course at SIGGRAPH, which convinced us that we could teach someone without previous graphics programming experience how to write a nontrivial shader-based OpenGL application program with just a little more work than with earlier versions of OpenGL.

As we developed this edition, we discovered some other reasons that convinced us that this approach is not only possible but even better for students learning computer graphics. Only a short while ago, we touted the advantages OpenGL gave us by being available for Windows, Linux, and Mac OS X so we could teach a course in which students could work in the environment they preferred. With OpenGL ES and WebGL they can now develop applications for their cell phones or Web browsers. We believe that this will excite both students and instructors about computer graphics and open up many new educational and commercial opportunities.

We feel that of even greater importance to the learning experience is the removal of most defaults and the fixed function pipeline in these new versions of OpenGL. At first glance, this removal may seem like it would make teaching a first course much harder. Maybe a little harder; but we contend much better. The tendency of most students was to rely on these functions and not pay too much attention to what the textbook and instructor were trying to teach them. Why bother when they could use built-in functions that did perspective viewing or lighting? Now that those functions are gone and students have to write their own shaders to do these jobs, they have to start by understanding the underlying principles and mathematics.

A Top-Down Approach

These recent advances and the success of the first five editions continue to reinforce our belief in a top-down, programming-oriented approach to introductory computer graphics. Although many computer science and engineering departments now support more than one course in computer graphics, most students will take only a single course. Such a course is placed in the curriculum after students have already studied programming, data structures, algorithms, software engineering, and basic mathematics. A class in computer graphics allows the instructor to build on these topics in a way that can be both informative and fun. We want these students to be programming three-dimensional applications as soon as possible. Low-level algorithms,

such as those that draw lines or fill polygons, can be dealt with later, after students are creating graphics.

John Kemeny, a pioneer in computer education, used a familiar automobile analogy: You don't have to know what's under the hood to be literate, but unless you know how to program, you'll be sitting in the back seat instead of driving. That same analogy applies to the way we teach computer graphics. One approach—the algorithmic approach—is to teach everything about what makes a car function: the engine, the transmission, the combustion process. A second approach—the survey approach—is to hire a chauffeur, sit back, and see the world as a spectator. The third approach—the programming approach that we have adopted here—is to teach you how to drive and how to take yourself wherever you want to go. As the old auto-rental commercial used to say, "Let us put *you* in the driver's seat."

Programming with OpenGL and C++

When Ed began teaching computer graphics over 25 years ago, the greatest impediment to implementing a programming-oriented course, and to writing a textbook for that course, was the lack of a widely accepted graphics library or application programmer's interface (API). Difficulties included high cost, limited availability, lack of generality, and high complexity. The development of OpenGL resolved most of the difficulties many of us had experienced with other APIs (such as GKS and PHIGS) and with the alternative of using home-brewed software. OpenGL today is supported on all platforms. It is bundled with Microsoft Windows and Mac OS X. Drivers are available for virtually all graphics cards. There is also an OpenGL API called Mesa that is included with most Linux distributions.

A graphics class teaches far more than the use of a particular API, but a good API makes it easier to teach key graphics topics, including three-dimensional graphics, lighting and shading, client–server graphics, modeling, and implementation algorithms. We believe that OpenGL's extensive capabilities and well-defined architecture lead to a stronger foundation for teaching both theoretical and practical aspects of the field and for teaching advanced concepts, including texture mapping, compositing, and programmable shaders.

Ed switched his classes to OpenGL about 15 years ago, and the results astounded him. By the middle of the semester, *every* student was able to write a moderately complex three-dimensional program that required understanding of three-dimensional viewing and event-driven input. In previous years of teaching computer graphics, he had never come even close to this result. That class led to the first edition of this book.

This book is a textbook on computer graphics; it is not an OpenGL manual. Consequently, it does not cover all aspects of the OpenGL API but rather explains only what is necessary for mastering this book's contents. It presents OpenGL at a level that should permit users of other APIs to have little difficulty with the material.

Unlike previous editions, this one uses C++ rather than C as the dominant programming language. The reason has to do with the OpenGL Shading Language (GLSL) used to write shaders, the programs that control the GPU. GLSL is a C-like language with atomic data types that include vectors and matrices, and overloaded

basic operators to manipulate them. All the modern versions of OpenGL require every application to provide two shaders; hence students need to use these features, which are supported in C++. By using just this part of C++ (simple classes, constructors, overloaded operators), we can implement fundamental graphics concepts, such as transformations and lighting, in either the application or in a shader with virtually identical code. In addition, using the simple matrix and vector classes that are provided on the book's Web site leads to much clearer, shorter code. Students who have started with Java or C should have little trouble with the code in the book.

Intended Audience

This book is suitable for advanced undergraduates and first-year graduate students in computer science and engineering and for students in other disciplines who have good programming skills. The book also will be useful to many professionals. Between us, we have taught well over 100 short courses for professionals; our experiences with these nontraditional students have had a great influence on what we have chosen to include in the book.

Prerequisites for the book are good programming skills in C++, Java, or C; an understanding of basic data structures (linked lists, trees); and a rudimentary knowledge of linear algebra and trigonometry. We have found that the mathematical backgrounds of computer science students, whether undergraduates or graduates, vary considerably. Hence, we have chosen to integrate into the text much of the linear algebra and geometry that is required for fundamental computer graphics.

Organization of the Book

The book is organized as follows. Chapter 1 provides an overview of the field and introduces image formation by optical devices; thus, we start with three-dimensional concepts immediately. Chapter 2 introduces programming using OpenGL. Although the first example program that we develop—each chapter has one or more complete programming examples—is two-dimensional, it is embedded in a three-dimensional setting and leads to a three-dimensional extension. We also introduce interactive graphics and develop event-driven graphics programs. Chapters 3 and 4 concentrate on three-dimensional concepts: Chapter 3 is concerned with defining and manipulating three-dimensional objects, whereas Chapter 4 is concerned with viewing them. Chapter 5 introduces light–material interactions and shading. These chapters should be covered in order and can be taught in about 10 weeks of a 15-week semester.

The next six chapters can be read in almost any order. All six are somewhat open ended and can be covered at a survey level, or individual topics can be pursued in depth. Chapter 6 surveys rasterization. It gives one or two major algorithms for each of the basic steps, including clipping, line generation, and polygon fill. Chapter 7 introduces many of the new discrete capabilities that are now supported in graphics hardware and by OpenGL. All these techniques involve working with various buffers. It concludes with a short discussion of aliasing problems in computer graphics. Chapters 6 and 7 conclude the discussion of the standard viewing pipeline used by all interactive graphics systems.

Chapter 8 contains a number of topics that fit loosely under the heading of hierarchical modeling. The topics range from building models that encapsulate the relationships between the parts of a model, to high-level approaches to graphics over the Internet. It includes an introduction to scene graphs and Open Scene Graph. Chapter 9 introduces a number of procedural methods, including particle systems, fractals, and procedural noise. Curves and surfaces, including subdivision surfaces, are discussed in Chapter 10. Chapter 11 surveys alternate approaches to rendering. It includes expanded discussions of ray tracing and radiosity, and an introduction to image-based rendering and parallel rendering.

Programs, primarily from the first part of the book, are included in Appendix A. They are also available online (see Support Materials). Appendices B and C contain a review of the background mathematics. Appendix D contains a synopsis of the OpenGL functions used in the book.

Changes from the Fifth Edition

The reaction of readers to the first five editions of this book was overwhelmingly positive, especially to the use of OpenGL and the top-down approach. Although each edition added material to keep up with what was going on in the field, the fifth edition made a major change by introducing programmable shaders and the OpenGL Shading Language. This material was somewhat optional because the existing versions of OpenGL were backward compatible. Most instructors chose to focus on the first six chapters and didn't get to programmable shaders.

As we pointed out at the beginning of this preface, with modern OpenGL, every application must provide shaders. Most of the basic functions from earlier versions, including those that specified geometry, transformations, and lighting parameters, have been deprecated. Consequently, programmable shaders and GLSL need to be introduced in Chapter 2. Many of the examples produce the same output as in previous editions but the code is very different.

We decided to incorporate the core material on interactivity in Chapter 2 and eliminate the separate chapter on input and interactivity. Thus, Chapter 2 became a little longer, but compared to previous editions, we feel that the added material on programmable shaders will only slightly delay the assignment of a first programming exercise.

Programmable shaders give the application programmer a choice of where to carry out most of the core graphics functionality. We have reorganized some of the material so as to be able to show the options together for a particular topic. For example, carrying out the lighting calculation in the application, in a vertex shader, and in a fragment shader are all in Chapter 5.

Given the positive feedback we've received on the core material from Chapters 1–6 in previous editions, we've tried to keep the changes to those chapters (now Chapters 1–5) to a minimum. We still see Chapters 1–5 as the core of any introductory course in computer graphics. Chapters 6–11 can be used in almost any order, either as a survey in a one-semester course or as the basis of a two-semester sequence.

Support Materials

The support for the book is on the Web, both through the author's Web site www.cs.unm.edu/~angel and Addison-Wesley's site www.aw.com/cssupport. Support material that is available to all readers of this book includes

- Sources of information on OpenGL
- Instructions on how to get started with OpenGL on the most popular systems
- Additional material on writing more robust OpenGL applications
- Program code
- Solutions to selected exercises
- PowerPoint lectures
- Figures from the book

Additional support materials, including solutions to all the nonprogramming exercises, are available only to instructors adopting this textbook for classroom use. Please contact your school's Addison-Wesley representative for information on obtaining access to this material.

Acknowledgments

Ed has been fortunate over the past few years to have worked with wonderful students at UNM. They were the first to get him interested in OpenGL, and he has learned much from them. They include Hue (Bumgarner-Kirby) Walker, Ye Cong, Pat Crossno, Tommie Daniel, Chris Davis, Lisa Desjarlais, Kim Edlund, Lee Ann Fisk, Maria Gallegos, Brian Jones, Christopher Jordan, Max Hazelrigg, Sheryl Hurley, Thomas Keller, Ge Li, Pat McCormick, Al McPherson, Ken Moreland, Martin Muller, David Munich, Jim Pinkerton, Jim Prewett, Dave Rogers, Hal Smyer, Takeshi Hakamata, Dave Vick, Brian Wylie, and Jin Xiong. Many of the examples in the color plates were created by these students.

The first edition of this book was written during Ed's sabbatical; various parts were written in five different countries. The task would not have been accomplished without the help of a number of people and institutions that made their facilities available to him. He is greatly indebted to Jonas Montilva and Chris Birkbeck of the Universidad de los Andes (Venezuela), to Rodrigo Gallegos and Aristides Novoa of the Universidad Tecnologica Equinoccial (Ecuador), to Long Wen Chang of the National Tsing Hua University (Taiwan), and to Kin Hong Wong and Pheng Ann Heng of the Chinese University of Hong Kong. Ramiro Jordan of ISTEC and the University of New Mexico made possible many of these visits. John Brayer and Jason Stewart at the University of New Mexico and Helen Goldstein at Addison-Wesley somehow managed to get a variety of items to him wherever he happened to be. His Web site contains a description of his adventures writing the first edition.

David Kirk and Mark Kilgard at NVIDIA were kind enough to provide cards for testing many of the algorithms. A number of other people provided significant help. He thanks Ben Bederson, Gonzalo Cartagenova, Tom Caudell, Kathi Collins, Kath-

leen Danielson, Roger Ehrich, Robert Geist, Chuck Hansen, Mark Henne, Bernard Moret, Dick Nordhaus, Helena Saona, Dave Shreiner, Vicki Shreiner, Gwen Sylvan, and Mason Woo. Mark Kilgard, Brian Paul, and Nate Robins are owed a great debt by the OpenGL community for creating software that enables OpenGL code to be developed over a variety of platforms.

At the University of New Mexico, the Art, Research, Technology, and Science Laboratory (ARTS Lab) and the Center for High Performance Computing have provided support for many of Ed's projects. The Computer Science Department, the Arts Technology Center in the College of Fine Arts, the National Science Foundation, Sandia National Laboratories, and Los Alamos National Laboratory have supported many of Ed's students and research projects that led to parts of this book. David Beining formerly with the Lodestar Astronomy Center and now at the ARTS Lab has provided tremendous support for the Fulldome Project. Sheryl Hurley, Christopher Jordan, Laurel Ladwig, Jon Strawn and Hue (Bumgarner-Kirby) Walker provided some of the images in the color plates through Fulldome projects. Hue Walker has done the wonderful covers for the previous three editions and her images are the basis of the cover for this edition.

Dave would like to first thank Ed for asking him to participate in this project. We've exchanged ideas on OpenGL and how to teach it for many years, and it's exciting to advance those concepts to new audiences. Dave would first like to thank his colleagues who worked at Silicon Graphics Computer Systems and who created OpenGL, and the OpenGL Working Group of the Khronos Group who continue to evolve the API. As Ed mentioned, SIGGRAPH has featured prominently in the development of these materials, and is definitely owed a debt of gratitude for providing access to enthusiastic test subjects for exploring our ideas.

Reviewers of the manuscript drafts provided a variety of viewpoints on what we should include and what level of presentation we should use. These reviewers for previous editions include Gur Saran Adhar (University of North Carolina at Wilmington), Mario Agrular (Jacksonville State University), Michael Anderson (University of Hartford), C. S. Bauer (University of Central Florida), Marty Barrett (East Tennessee State University), Kabekode V. Bhat (The Pennsylvania State University), Eric Brown, Robert P. Burton (Brigham Young University), Sam Buss (University of California, San Diego), Kai H. Chang (Auburn University), Ron DiNapoli (Cornell University), Eric Alan Durant (Milwaukee School of Engineering), David S. Ebert (Purdue University), Richard R. Eckert (Binghamton University), Jianchao (Jack) Han (California State University, Dominguez Hills), Chenyi Hu (University of Central Arkansas), Mark Kilgard (NVIDIA Corporation), Lisa B. Lancor (Southern Connecticut State University), Chung Lee (CA Polytechnic University, Pomona), John L. Lowther (Michigan Technological University), R. Marshall (Boston University and Bridgewater State College), Hugh C. Masterman (University of Massachusetts, Lowell), Bruce A. Maxwell (Swathmore College), James R. Miller (University of Kansas), Rodrigo Obando (Columbus State University), Andrea Salgian (The College of New Jersey), Lori L. Scarlatos (Brooklyn College, CUNY), Han-Wei Shen (The Ohio State University), Oliver Staadt (University of California, Davis), Stephen L. Stepoway (Southern Methodist University), Bill Toll (Taylor University), Michael

Wainer (Southern Illinois University, Carbondale), Yang Wang (Southern Methodist State University), Steve Warren (Kansas State University), Mike Way (Florida Southern College), George Wolberg (City College of New York), Xiaoyu Zhang (California State University San Marcos), Ye Zhao (Kent State University). and Ying Zhu (Georgia State University). Although the final decisions may not reflect their views—which often differed considerably from one another—each reviewer forced us to reflect on every page of the manuscript.

The reviewers for this edition were particularly supportive. They include Norman I. Badler (University of Pennsylvania), Mike Bailey(Oregon State University), Bedrich Benes (Purdue University), Isabelle Bichindaritz (University of Washington, Tacoma), Cory D. Boatright, (University of Pennsylvania), Eric Brown, James Cremer (University of Iowa), John David N. Dionisio (Loyola Marymount University), W Randolph Franklin (Rensselaer Polytechnic Institute), Natacha Gueorguieva, (City University of New York/College of Staten Island), George Kamberov (Stevens Institute of Technology), Hugh Masterman (University of Massachusetts, Lowell), Tim McGraw (West Virginia University), Jon A. Preston (Southern Polytechnic State University), and Bill Toll (Taylor University). They were asked to review material that was not in their own courses and if adapted would change their courses significantly. Each one recognized the importance of our approach and indicated a willingness to adopt it.

We would also like to acknowledge the entire production team at Addison-Wesley. Ed's editors, Peter Gordon, Maite Suarez-Rivas, and Matt Goldstein, have been a pleasure to work with through six editions of this book and the OpenGL primer. Through six editions, Paul Anagnostopoulos at Windfall Software has always been more than helpful in assisting with TEX problems. Ed is especially grateful to Lyn Dupré. If the readers could see the original draft of the first edition, they would understand the wonders that Lyn does with a manuscript.

Ed wants to particularly recognize his wife, Rose Mary Molnar, who did the figures for his first graphics book, many of which form the basis for the figures in this book. Probably only other authors can fully appreciate the effort that goes into the book production process and the many contributions and sacrifices our partners make to that effort. The dedication to the book is a sincere but inadequate recognition of all of Rose Mary's contributions to Ed's work.

Dave would like to recognize the support and encouragement of Vicki, his wife, without whom creating works like this would never occur. Not only does she provide warmth and companionship, but also provides invaluable feedback on our presentation and materials. She's been a valuable, unrecognized partner in all of Dave's OpenGL endeavors.

Ed Angel
Dave Shreiner