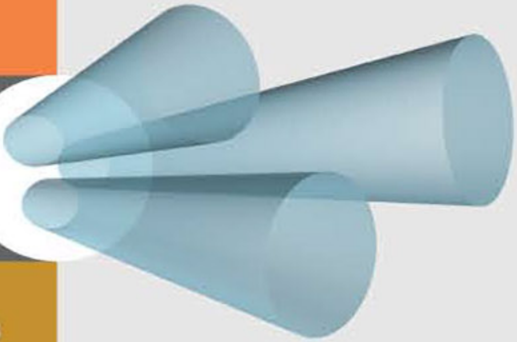
DAVID H. EBERLY

SERIES IN INTERACTIVE 3D TECHNOLOGY

3D GAME ENGINE DESIGN

A PRACTICAL APPROACH TO REAL-TIME COMPUTER GRAPHICS

SECOND EDITION



MORGAN KAUFMANN

3D Game Engine Design

A Practical Approach to Real-Time Computer Graphics

3D Game Engine Design

A Practical Approach to Real-Time Computer Graphics

Second Edition

David H. Eberly

Geometric Tools, Inc.

ELSEVIER

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Morgan Kaufmann is an imprint of Elsevier

MORGAN KAUFMANN PUBLISHERS

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Version Date: 20150223

International Standard Book Number-13: 978-1-4822-6730-3 (eBook - PDF)

TRADEMARKS

About the Author

Dave Eberly is the president of Geometric Tools, Inc. (*www.geometrictools.com*), a company that specializes in software development for computer graphics, image analysis, and numerical methods. Previously, he was the director of engineering at Numerical Design Ltd. (NDL), the company responsible for the real-time 3D game engine, NetImmerse. He also worked for NDL on Gamebryo, which was the next-generation engine after NetImmerse. His background includes a BA degree in mathematics from Bloomsburg University, MS and PhD degrees in mathematics from the University of Colorado at Boulder, and MS and PhD degrees in computer science from the University of North Carolina at Chapel Hill. He is the author of *Game Physics* (2004) and *3D Game Engine Architecture* (2005) and coauthor with Philip Schneider of *Geometric Tools for Computer Graphics* (2003), all published by Morgan Kaufmann. As a mathematician, Dave did research in the mathematics of combustion, signal and image processing, and length-biased distributions in statistics. He was an associate professor at the University of Texas at San Antonio with an adjunct appointment in radiology at the U.T. Health Science Center at San Antonio. In 1991, he gave up his tenured position to retrain in computer science at the University of North Carolina. After graduating in 1994, he remained for one year as a research associate professor in computer science with a joint appointment in the Department of Neurosurgery, working in medical image analysis. His next stop was the SAS Institute, working for a year on SAS/Insight, a statistical graphics package. Finally, deciding that computer graphics and geometry were his real calling, Dave went to work for NDL (which is now Emergent Game Technologies), then to Magic Software, Inc., which later became Geometric Tools, Inc. Dave's participation in the newsgroup *comp.graphics.algorithms* and his desire to make 3D graphics technology available to all are what has led to the creation of his company's Web site and his books.

Contents

Preface

The first edition of *3D Game Engine Design* appeared in print over six years ago (September 2000). At that time, shader programming did not exist on consumer graphics hardware. All rendering was performed using the fixed-function pipeline, which consisted of setting render states in order to control how the geometric data was affected by the drawing pass.

The first edition contained a CDROM with the source code for Wild Magic Version 0.1, which included 1,015 source files and 17 sample applications, for a total of 101,293 lines of code. The distribution contained support only for computers running the Microsoft Windows operating system; the renderer was built on top of OpenGL; and project files were provided for Micrsoft Visual C++ 6. Over the years, the source code evolved to Wild Magic Version 3.9, which contained additional support for Linux and Macintosh platforms, had OpenGL and Direct3D renderers, and included some support for shader programming. However, the design of the engine was still based on a fixed-function pipeline. The distribution also included support for multiple versions of Microsoft's compilers, support for other compilers on the various platforms, and contained some tools such as importers and exporters for processing of art assets.

This is the second edition of *3D Game Engine Design*. It is much enhanced, describing the foundations for shader programming and how an engine can support it. The second edition is about twice the size of the first. The majority of the increase is due to a more detailed description of all aspects of the graphics system, particularly about how shaders fit into the geometric pipeline. The material on scene graphs and their management is also greatly expanded. The second edition has more figures and less emphasis on the mathematical aspects of an engine.

The second edition contains a CDROM with the source code for Wild Magic Version 4.0, which includes 1,587 source files and 105 sample applications, for a total of 249,860 lines of code. The Windows, Linux, and Macintosh platforms are still supported, using OpenGL renderers. The Windows platform also has a Direct3D renderer whose performance is comparable to that of the OpenGL renderer. Multiple versions of Microsoft's C++ compilers are supported—versions 6, 7.0, 7.1, and 8.0 (Professional and Express Editions). The MINGW compiler and MSYS environment are also supported on the Windows platform. The Linux platform uses the g++ compiler, and the Macintosh platform uses Apple's Xcode tools.

The graphics system of Wild Magic Version 4.0 is fully based on shader programming and relies on NVIDIA's Cg programming language. The precompiled shader programs were created using the `arbvp1` and `arbfp1` profiles for OpenGL and using the `vs_2_0` and `ps_2_0` profiles for Direct3D, so your graphics hardware must support these in order to run the sample applications. If your graphics hardware supports only lesser profiles such as `vs_1_1` and `ps_1_1`, you must recompile the shader programs with these profiles and use the outputs instead of what is shipped on the CDROM. The distribution also contains a fully featured, shader-based software renderer to illustrate all aspects of the geometric pipeline, not just the vertex and pixel shader components.

The replacement of the fixed-function approach by a shader-based approach has made Wild Magic Version 4 a much more powerful graphics engine for use in all graphics applications, not just in games. Much effort went into making the engine easier to use and to extend, and into improving the performance of the renderers. I hope you enjoy this new manifestation of Wild Magic!

A book is never just the product of the author alone. Many people were involved in making this book as good as it possibly can be. Thanks to the reviewers for providing valuable and insightful feedback about the first edition regarding how to improve it for a second edition. A special thanks goes to Marc Olano (University of Maryland, Baltimore County) for taking the time to provide me with detailed comments based on his experience using the first edition as a textbook. Thanks to Elisabeth Beller, the production editor and project manager for all of my Morgan Kaufmann Publisher books, for assembling yet another fine group of people who have the superb ability to take my unattractive book drafts and make them look really good. And, as always, thanks to my editor Tim Cox for his patience and help in producing yet another book for Morgan Kaufmann Publishers.

CHAPTER 1

Introduction

I have no fault to find with those who teach geometry. That science is the only one which has not produced sects; it is founded on analysis and on synthesis and on the calculus; it does not occupy itself with probable truth; moreover it has the same method in every country.

— Frederick the Great

1.1 The Evolution of Graphics Hardware and Games

The first edition of *3D Game Engine Design* was written in the late 1990s when 3dfx Voodoo cards were in style and the NVIDIA Riva TNT cards had just arrived. The book was written based on the state of graphics at that time. Six years have passed between that edition and this, the second edition. Graphics hardware has changed dramatically. So have games. The hardware has extremely powerful graphics processing units (GPUs), lots of video memory, and the option of programming it via shader programs. (These did not exist on consumer cards when I wrote the first edition.) Games have evolved also, having much richer (and much more) content and using more than graphics. We now have physics engines and more recently physics processors (PhysX from Ageia).

The Sony Playstation 2 was not quite released when I started writing the first edition. We've also seen Microsoft's Xbox arrive on the scene, as well as the Nintendo GameCube. These days we have Microsoft's Xbox 360 with multiple processors, and the Sony Playstation 3 is soon to follow with the Cell architecture. Smaller game-playing devices are available. Mobile phones with video screens are also quite popular.

With all this evolution, the first edition of the book has shown its age regarding the discussion of real-time graphics. The time is right for the second edition, so here it is.

1.2 The Evolution of This Book and Its Software

In the late 1990s when I conceived the idea of writing a book on real-time graphics as used in games, I was employed by Numerical Design, Ltd. (now Emergent Game Technologies) designing and developing NetImmerse (now Gamebryo). At that time the term *game engine* really did refer to the graphics portion of the system. Companies considering using NetImmerse wanted the real-time graphics in order to free up the computer processing unit (CPU) for use by other systems they themselves were used to building: the game logic, the game artificial intelligence (AI), rudimentary collision and physics, networking, and other components. The first edition of *3D Game Engine Design* is effectively a detailed summary of what went into building NetImmerse.

Over the years I have received some criticism for using "game engine" in the title when the book is mainly about graphics. Since that time, the term *game engine* has come to mean a collection of engines—for graphics, physics, AI, networking, scripting, and you name it. It is not feasible to write a book in a reasonable amount of time with sufficient depth to cover all these topics, nor do I intend to write such a massive tome. To address the criticism about the book title, I could have changed the title itself. However, I have chosen to keep the original title—the book is known now by its name, for better or for worse. The second edition includes some discussion about physics and some discussion about an application layer and how the engines must interact, but probably this is not enough to discourage the criticism about the title. So be it.

The first edition appeared in print in September 2000. It is now six years later and the book remains popular in many circles. The algorithmic aspects are still relevant, the scene graph management still applies, but the material on rendering is clearly out of date. That material was essentially about the *fixed-function pipeline* view of a graphics system. The evolution of graphics hardware to support a *shader-based pipeline* has been rapid, now allowing us to concentrate on the special effects themselves (via shader programming) rather than trying to figure out how to call the correct set of state-enabling functions in the fixed-function pipeline to obtain a desired effect.

The second edition of the book now focuses on the design of the scene graph managment system and its associated rendering layer. Most of the algorithmic concepts have not changed regarding specialized scene graph classes such as the controller classes, the sorting classes, or level-of-detail classes. Core classes such as the spatial, geometry, and node classes have changed to meet the needs of a shader-based system. The current scene graph management system is much more powerful, flexible, and efficient than its predecessors. The shader effect system is integrated with the scene graph management so that you may do single-pass drawing, multipass drawing with a single effect, or even drawing with multiple effects. I have paid much attention to hiding as many details as possible from the application developer, relying on well-designed and automated subsystems to do the work that earlier versions of my scene graph management system forced the developer to do.

One characteristic of my books that I believe sets them apart from other technical books is the inclusion of large source code libraries, a lot of sample applications that actually compile and run, and support for multiple platforms (PC, Mac, Linux/Unix, and various compilers on each platform). What you have purchased is a book *and* a software product to illustrate what is described in the book. The sample source code that ships with many books is not carefully planned, lacks quality control, is not multiplatform, and usually is not maintained by the book authors. I am interested in carefully designed and planned code. I believe in quality source code. I maintain the source code on a regular basis, so I encourage people to send email about problems they encounter, both with the source code and in the book material. The Geometric Tools website lists all the updates to the software, including bug fixes as well as new features, and the site has pages for book corrections.

The first edition of this book shipped with Wild Magic version 0.1. The book had two additional printings in its first edition, one shipping with Wild Magic version 0.2 and one shipping with Wild Magic version 0.4. The second edition ships with Wild Magic version 4.0, which when compared to version 0.1 looks very little like it. I believe the quality of the Wild Magic source code is a significant feature that has attracted many users. The latest version represents a significant rewrite to the rendering layer that has led to easier use of the engine and better performance by the renderers. The rewrite represents three months of dedicated time, something most authors would never consider investing time in, and it includes implementing a shader-based software renderer just to illustrate the book concepts in detail. I hope you enjoy using Wild Magic for your leisure projects!

1.3 A Summary of the Chapters

The book is partitioned into six parts.

Graphics. Chapter 2 discusses the details of a rendering system, including transformations, camera models, culling and clipping, rasterizing, and issues regarding software versus hardware rendering and about specific graphics application programmer interfaces (graphics APIs) in use these days. Chapter 3 is about rendering from the perspective of actually writing all the subsystems for a software renderer. The chapter includes what I consider a reasonable abstraction of the interface for a shader-based rendering system. This interface essentially shows that the renderer spends most of its time doing resource management. Chapter 3 also includes details about shader programs—not about writing them but about dealing with data management issues. Here I address such things as matching geometric vertex data to vertex program inputs, matching vertex program outputs to pixel program inputs, and ensuring that the infrastructure is set so that all resources are in the right place at the right time, hooked up, and ready to use for real-time rendering.

Scene Graph Management. Chapter 4 is about the essentials of organizing your data as a scene graph. This system is designed to be high level to allow ease of use by application programmers, to be an efficient system to feed a renderer, and to be naturally extensible. The chapter also includes a section that talks about scene graph compiling—converting scene graphs to more optimized forms for target platforms. Chapters 5, 6, and 7 are about specially designed nodes and subsystems of the scene graph management system. These include subsystems to support animation, spatial sorting, and level of detail.

Collision Detection and Physics. Some general concepts you see in attempting to have physical realism in a three-dimensional (3D) application are discussed in Chapters 8 and 9. A generic approach to collision detection is presented, one that I have successfully implemented in a real environment. I also discuss picking operations and briefly talk about automatic pathfinding as a means for collision avoidance. The chapter on physics is a brief discussion of some concepts you will see often when working with physical simulations, but it does not include a discussion about the black-box-style physics you see in a commercial physics engine, such as Havok. That type of physics requires a lot more discussion than space allows in this book. Such physics is heavily mathematical and requires attention to issues of numerical round-off errors when using floating-point arithmetic.

Mathematical Topics. Chapters 10 through 17 include a lot of the mathematical detail for much of the source code you will find in Wild Magic. These chapters include a discussion of standard objects encountered in geometric manipulation and queries, including curves and surfaces covered in Chapters 11 and 12. You will also find material on queries regarding distance, containment, and intersection. Chapter 16 presents some common numerical methods that are useful in graphics and physics applications. The final chapter in this partition is about the topic of rotation, including basic properties of rotation matrices and how quaternions are related to matrices.

Software Engineering. Chapter 18 is a brief summary of basic principles of object-oriented design and programming. Various base-level support for large libraries is important and includes topics such as run-time type information, shared objects and reference counting, streaming of data (to/from disk, across a network), and initialization and termination for disciplined object creation and destruction in an application. Chapter 19 is about memory management. This is of particular importance when you want to write your own memory managers in order to build a system that is handed a fixed-side memory block and told it may only use memory from that block. In particular, this approach is used on game consoles where each engine (graphics, physics, sound, and so on) is given its memory "budget." This is important for having predictable behavior of the engines. The last thing you want to happen in your game is one system consuming so much memory from a global heap that another system fails because it cannot successfully allocate what it needs.

Special Effects Using Shaders. Chapter 20 shows a handful of sample shaders and the applications that use them. This is just to give you an idea of what you can do with shaders and how Wild Magic handles them. The appendix describes how you can add new shader effects to Wild Magic. The process is not difficult (by design).

I believe the organization here is an improvement over that of the first edition of the book. A number of valid criticisms of the first edition were about the amount of mathematics interleaved in the discussions. Sorry, but I remain a firm believer that you need a lot of mathematics when building sophisticated graphics and physics engines. That said, I have made an attempt to discuss first the general concepts for graphics, factoring the finer detail into the chapters in the mathematics section that occurs late in the book. For example, it is possible to talk about culling of bounding volumes against frustum planes without immediately providing the details of the algorithm for specific bounding volumes. When discussing distance-based collision detection, it is possible to motivate the concepts without the specific distance algorithms for pairs of objects. The mathematics is still here, but factored to the end of the book rather than interleaved through the entire book.

Another criticism of the first edition of the book was its lack of figures. I believe I have remedied this, adding quite a few more figures to the second edition. That said, there may be places in the book where someone might feel the need for a figure where I thought the concept did not require one. My apologies if this happens. Send me feedback by email if you believe certain parts of the book can be improved by the addition of figures.

Finally, I have included some exercises in the book. Creating a large set of well-crafted exercises is a full-time job. In fact, I recall meeting a person who worked for Addison-Wesley (back in the early 1980s). His full-time job was managing the exercises for the calculus textbook by George Thomas and Ross Finney (seventh edition at that time). As much as I would like to have included more exercises here, my time budget for writing the book, writing the Wild Magic source code to go with the book, and making a living doing contract programming already exceeded 24 hours per day. I hope the exercises I have included will support the use of the book as a textbook in a graphics course. Most of them are programming exercises, requests to modify the source code to do something different or to do something in addition to what it does. I can imagine some of these taking quite some time to do. But I also believe they will make students think—the point of exercises!

This page intentionally left blank

References

[AJ97] E. Andres and M. A. Jacob. The discrete analytical hyperspheres. *IEEE Transactions on Visualization and Computer Graphics*, 3(1):75–86, 1997.

[AM01] Timo Aila and Ville Miettinen. *SurRender Umbra Reference Manual*, vol. 2.3, 2001.

[And94] E. Andres. Discrete circles, rings and spheres. *Computer and Graphics*, 18(5):695–706, 1994.

[Arv91] James Arvo, ed. *Graphics Gems II*. Academic Press, San Diego, CA, 1991.

[AS65] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover, New York, 1965.

[Bar01] David Baraff. Physically Based Modeling: Rigid Body Simulation. *www.pixar.com/companyinfo/research/pbm2001/notesg.pdf*, 2001, 68 pages.

[BF01] Richard L. Burden and J. Douglas Faires. *Numerical Analysis*, 7th ed., Brooks/Cole, Belmont, CA, 2001.

[Bli77] James F. Blinn. Models of light reflection for computer synthesized pictures. *ACM Computer Graphics (Proceedings of SIGGRAPH 1977)*, July 1977, pp. 192–198.

[Boo87] Grady Booch. *Software Components with Ada: Structures, Tools, and Subsystems*. Pearson Benjamin Cummings, San Francisco, CA, 1987.

[Bre65] J. E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965.

[Cam97] Stephen Cameron. Enhancing GJK: Computing minimum and penetration distances between convex polyhedra. In *Proceedings of the IEEE Int'l Conference on Robotics and Automation*, pp. 3112–3117, 1997.

[CLR90] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.

[COM98] Jonathan D. Cohen, Marc Olano, and Dinesh Manocha. Appearance-preserving simplifications. In *Proceedings of SIGGRAPH*, 1998, pp. 115–122.

[Cor] Microsoft Corporation. The DirectX Software Development Kit. *www.microsoft.com*.

[CRE01] Elaine Cohen, Richard F. Riesenfeld, and Gershon Elber. *Geometric Modeling with Splines: An Introduction*. A. K. Peters, Natick, MA, 2001.

[CVM$^+$96] J. D. Cohen, A. Varshney, D. Manocha, G. Turk, et al. Simplification envelopes. In *Proceedings of SIGGRAPH*, 1996, pp. 119–128.

[DK90] D. P. Dobkin and D. G. Kirkpatrick. Determining the separation of preprocessed polyhedra—a unified approach. *Proceedings of the 17th Int'l Colloq. Automata Lang. Program, Lecture Notes in Computer Science*, 43:400–413, 1990.

[DSS88] H. Das, J-J. E. Slotine, and T. B. Sheridan. Inverse kinematic algorithms for redundant systems. *IEEE International Conference on Robotics and Automation*, 1988, pp. 43–48.

[DWS$^+$97] Mark Duchaineau, Murray Wolinsky, David E. Sigeti, Mark C. Miller, et al. Roaming terrain: Real-time optimally adapting meshes. *IEEE Visualization 1997*, pp. 81–88.

[Ebe03] David Eberly. *Game Physics*. Morgan Kaufmann, San Francisco, CA, 2003.

[Ede87] H. Edelsbrunner. *Algorithms in Computational Geometry*. Springer-Verlag, Heidelberg, Germany, 1987.

[EM85] H. Edelsbrunner and H. A. Maurer. Finding extreme points in three dimensions and solving the post-office problem in the plane. *Inform. Process. Lett.*, 21:39–47, 1985.

[Eng02] Wolfgang F. Engel, ed. *Direct3D ShaderX: Vertex and Pixel Shader Tips and Tricks*. Wordware, Plano, TX, 2002.

[Eng03] Wolfgang F. Engel, ed. *ShaderX2: Shader Programming Tips & Tricks*. Wordware, Plano, TX, 2003.

[Eng04] Wolfgang F. Engel, ed. *ShaderX3: Advanced Rendering with DirectX and OpenGL*. Charles River Media, Hingham, MA, 2004.

[Eng06] Wolfgang F. Engel, ed. *ShaderX4: Lighting and Rendering*. Charles River Media, Hingham, MA, 2006.

[Eri04] Christer Ericson. *Real-Time Collision Detection*. Morgan Kaufmann, San Francisco, CA, 2004.

[Far90] Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design*. Academic Press, San Diego, CA, 1990.

[Far99] Gerald Farin. *NURBS: From Projective Geometry to Practical Use*. A. K. Peters, Natick, MA, 1999.

[Fer04] Randima Fernando, ed. *GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics*. Addison-Wesley, Boston, MA, 2004.

[FG03] Kaspar Fischer and Bernd Gärtner. The smallest enclosing ball of balls: Combinatorial structure and algorithms. In *Annual Symposium on Computational Geometry: Proceedings of the Nineteenth Conference on Computational Geometry*, 2003, pp. 292–301.

[FKN79] Henry Fuchs, Zvi Kedem, and Bruce Naylor. Predetermining visibility priority in 3D scenes. In *Proceedings of SIGGRAPH*, 1979, pp. 175–181.

[FKN80] Henry Fuchs, Zvi Kedem, and Bruce Naylor. On visible surface generation by a priori tree structures. In *Proceedings of SIGGRAPH*, 1980, pp. 124–133.

[FvDFH90] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer Graphics: Principles and Practice*, 2nd ed. Addison-Wesley, Reading, MA, 1990.

[GF90] E. G. Gilbert and C-P. Foo. Computing the distance between general convex objects in three-dimensional space. *IEEE Transactions on Robotics and Automation*, 6(1):53–61, 1990.

[GH97] Michael Garland and Paul Heckbert. Surface simplification using quadric error metrics. In *Proceedings of SIGGRAPH*, 1997, pp. 209–216.

[GH98] Michael Garland and Paul Heckbert. Simplifying surfaces with color and texture using quadric error metrics. *IEEE Visualization*, 1998, pp. 263–269.

[GJK88] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between objects in three-dimensional space. *IEEE Journal Robotics and Automation*, vol. 4:193–203, 1988.

[GL93] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*, 2nd ed. Johns Hopkins University Press, Baltimore, MD, 1993.

[Gla90] Andrew S. Glassner, ed. *Graphics Gems I*. Academic Press, San Diego, CA, 1990.

[GLM96] Stefan Gottschalk, Ming Lin, and Dinesh Manocha. OBBTree: A hierarchical structure for rapid interference detection. In *Proceedings of SIGGRAPH*, 1996, pp. 171–180.

[Hec94] Paul Heckbert, ed. *Graphics Gems IV*. Academic Press, San Diego, CA, 1994.

[HJ85] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, UK, 1985.

[Hop96a] Hugues Hoppe. Progressive meshes. In *Proceedings of SIGGRAPH*, 1996, pp. 99–108.

[Hop96b] Hugues Hoppe. View-dependent refinement of progressive meshes. In *Proceedings of SIGGRAPH*, 1996, pp. 189–198.

[Kal] Michael Kallay. Computing moment of inertia. Private correspondence, November 29, 2004.

[KB86] Doris H. U. Kochanek and Richard H. Bartels. Interpolating splines with local tension, continuity, and bias control. *ACM SIGGRAPH Course Notes 22, Advanced Computer Animation*, 1986.

[Kir83] D. G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal on Computing*, 12:28–35, 1983.

[Kir92] David Kirk, ed. *Graphics Gems III*. Academic Press, San Diego, CA, 1992.

[Knu73] Donald E. Knuth. *The Art of Computer Programming, Volumes 1–3*. Addison-Wesley, Boston, MA, 1973.

[Lan98] Jeff Lander. Oh my god, i inverted kine. *Game Developer Magazine*, 1998, pp. 9–14.

[LB84] Y-D. Liang and B. A. Barsky. A new concept and method for line clipping. *ACM Transactions on Graphics*, 3(1):1–22, 1984.

[LE97] D. Luebke and C. Erikson. View-dependent simplification of arbitrary polygonal environments. In *Proceedings of ACM SIGGRAPH 1997*, pp. 199–208.

[Lev00] Ron Levine. Collision of Moving Objects. *www.sourceforce.net* (on the game developer algorithms list), November 2000.

[LH04] F. Losasso and H. Hoppe. Geometry clipmaps: Terrain rendering using nested regular grids. In *Proceedings of ACM SIGGRAPH 2004*, pp. 769–776.

[Lib] Boost C++ Libraries. Computing Moment of Inertia. *www.boost.org/*.

[Lim01] Ravenbrook Limited. The Memory Management Reference. *www.memory-management.org*, 2001.

[LKR$^+$96] Peter Lindstrom, David Koller, William Ribarsky, Larry F. Hodges, et al. Real-time, continuous level of detail rendering of height fields. In *Proceedings of SIGGRAPH*, 1996, pp. 109–118.

[LLM05] Stanley B. Lippman, Jósee Lajoie, and Barbara E. Moo. *C++ Primer*, 4th ed. Addison-Wesley, Reading, MA, 2005.

[Lom03] Chris Lomont. Fast Inverse Square Root. *www.math.purdue.edu/~clomont /Math/Papers/2003/InvSqrt.pdf*, 2003.

[LRC$^+$03] David Luebke, Martin Reddy, Jonathan D. Cohen, Amitabh Varshney, et al. *Level of Detail for 3D Graphics*. Morgan Kaufmann, San Francisco, CA, 2003.

[LT98] Peter Lindstrom and Greg Turk. Fast and memory efficient polygonal simplification. *IEEE Visualization*, 1998, pp. 279–286.

[MB05] Tom McReynolds and David Blythe. *Advanced Graphics Programming Using OpenGL*. Morgan Kaufmann, San Francisco, CA, 2005.

[Mey88] Bertrand Meyer. *Object-Oriented Software Construction*. International Series in Computer Science, C. A. R. Hoare, ed., Prentice Hall, Englewood Cliffs, NJ, 1988.

[MH02] Tomas Möller and Eric Haines. *Real-Time Rendering*, 2nd ed. A. K. Peters Ltd., Natick, MA, 2002.

[Mir96] Brian Mirtich. Fast and accurate computation of polyhedral mass properties. *Journal of Graphics Tools*, 1(2):31–50, 1996.

[Möl97] Tomas Möller. A fast triangle-triangle intersection test. *Journal of Graphics Tools*, 2(2):25–30, 1997.

[MT97] Tomas Möller and Ben Trumbore. Fast, minimum storage ray-triangle intersection. *Journal of Graphics Tools*, 2(1):21–28, 1997.

[NN90] Zoran R. Novaković and Bojan Nemec. A solution of the inverse kinematics problem using the sliding mode. *IEEE Transactions on Robotics and Automation*, 6(2):247–252, 1990.

[O'R85] J. O'Rourke. Finding minimal enclosing boxes. *Int'l Journal of Comput. Inform. Sci.*, 14:183–199, June 1985.

[O'R98] Joseph O'Rourke. *Computational Geometry in C*, 2nd ed. Cambridge University Press, Cambridge, UK, 1998.

[Pae95] Alan Paeth, ed. *Graphics Gems V*. Academic Press, San Diego, CA, 1995.

[Pau06] Brian Paul. The Mesa 3D Graphics Library. *www.mesa3d.org*, 2006.

[PFTV88] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Cambridge, UK, 1988.

[Pha05] Matt Pharr, ed. *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*. Addison-Wesley, Boston, MA, 2005.

[Pho00] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):335–342, July 2000.

[Pro06] Virtual Terrain Project. *www.vterrain.org/LOD/Papers/*, 2006.

[PS85] Franco P. Preparata and Michael I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, Heidelberg, Germany, 1985.

[PZB90] C. Phillips, J. Zhao, and N. Badler. Interactive real-time articulated figure manipulation using multiple kinematic constraints. *SIGGRAPH I3D Symposium*, 24(2):245–250, 1990.

[RB93] Jared Rossignac and Paul Borrel. Multiresolution 3D approximations for rendering complex scenes. In *Modeling in Computer Graphics: Methods and Applications*, B. Falcidieno and T. Kunii, eds. Springer-Verlag, Heidelberg, Germany, 1993, pp. 455–463.

[RM03] Andrew Rollings and Dave Morris. *Game Architecture and Design: A New Edition*. Peachpit Press, Berkeley, CA, 2003.

[Rog01] David F. Rogers. *An Introduction to NURBS with Historical Perspective*. Morgan Kaufmann, San Francisco, CA, 2001.

[RWPD05] Erik Reinhard, Greg Ward, Sumanta Pattanaik, and Paul Debevec. *High Dynamic Range Imaging: Acquisition, Display, and Image-Based Lighting*. Morgan Kaufmann, San Francisco, CA, 2005.

[SE02] Philip J. Schneider and David Eberly. *Geometric Tools for Computer Graphics*. Morgan Kaufmann, San Francisco, CA, 2002.

[Sha99] Brian Sharp. Optimizing curved surface geometry. *Game Developer Magazine*, 1999, pp. 40–48.

[Spe06] Henry Spencer. *getopt* command-line parser. *www.lysator.liu.se/c/henry/l*, 2006.

[SS87] Lorenzo Sciavicco and Bruno Siciliano. A dynamic solution to the inverse kinematic problem for redundant manipulators. In *IEEE Int'l Conference on Robotics and Automation*, 1987, pp. 1081–1086.

[Str00] Bjarne Stroustrup. *The C++ Programming Language*, 3rd ed. Addison-Wesley, Reading, MA, 2000.

[SWND05] Dave Shreiner, Mason Woo, Jackie Neider, and Tom Davis. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 2*. Addison-Wesley, Boston, MA, 2005.

[SZL92] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. In *Proceedings of SIGGRAPH*, 1992, pp. 65–70.

[Tou84] G. T. Toussaint. An optimal algorithm for computing the minimum vertex distance between two crossing convex polygons. *Proceedings of IEEE Int'l Conference on Pattern Recogn.*, 1984, pp. 465–467.

[vdB99] Gino van den Bergen. A fast and robust GJK implementation for collision detection of convex objects. *Journal of Graphics Tools*, 4(2):7–25, 1999.

[vdB03] Gino van den Bergen. *Collision Detection in Interactive 3D Environments*. Morgan Kaufmann, San Francisco, CA, 2003.

[Wad] Bretton Wade. BSP Tree Frequently Asked Questions. *www.faqs.org/faqs /graphics/bsptree-faq/*.

[WC91] Li-Chun Wang and Chih Cheng Chen. A combined optimization method for solving the inverse kinematics problem of mechanical manipulators. *IEEE Transactions on Robotics and Applications*, 7(4):489–499, 1991.

[Wel91] Emo Welzl. Smallest enclosing disks (balls and ellipsoids). *Lecture Notes in Computer Science*, 555:359–370, 1991.

[Wel93] Chris Welman. "Inverse Kinematics and Geometric Constraints for Articulated Figures." Master's thesis, Simon Frasier University, Burnaby, Canada, 1993.

[WG95a] Chionh Eng Wee and Ronald N. Goldman. Elimination and resultants, Part 1: Elimination and bivariate resultants. *IEEE Computer Graphics and Applications*, 1995, pp. 69–77.

[WG95b] Chionh Eng Wee and Ronald N. Goldman. Elimination and resultants, Part 2: Multivariate resultants. *IEEE Computer Graphics and Applications*, 1995, pp. 60–69.

[Wil83] Lance Williams. Pyramidal parametrics. *Computer Graphics*, 7(3):1–11, 1983.

[WW92] Alan Watt and Mark Watt. *Animation and Rendering Techniques: Theory and Practice*. ACM Press, New York, 1992.

[ZB94] Jianmin Zhao and Norman I. Badler. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics*, 13(4):313–336, 1994.

This page intentionally left blank