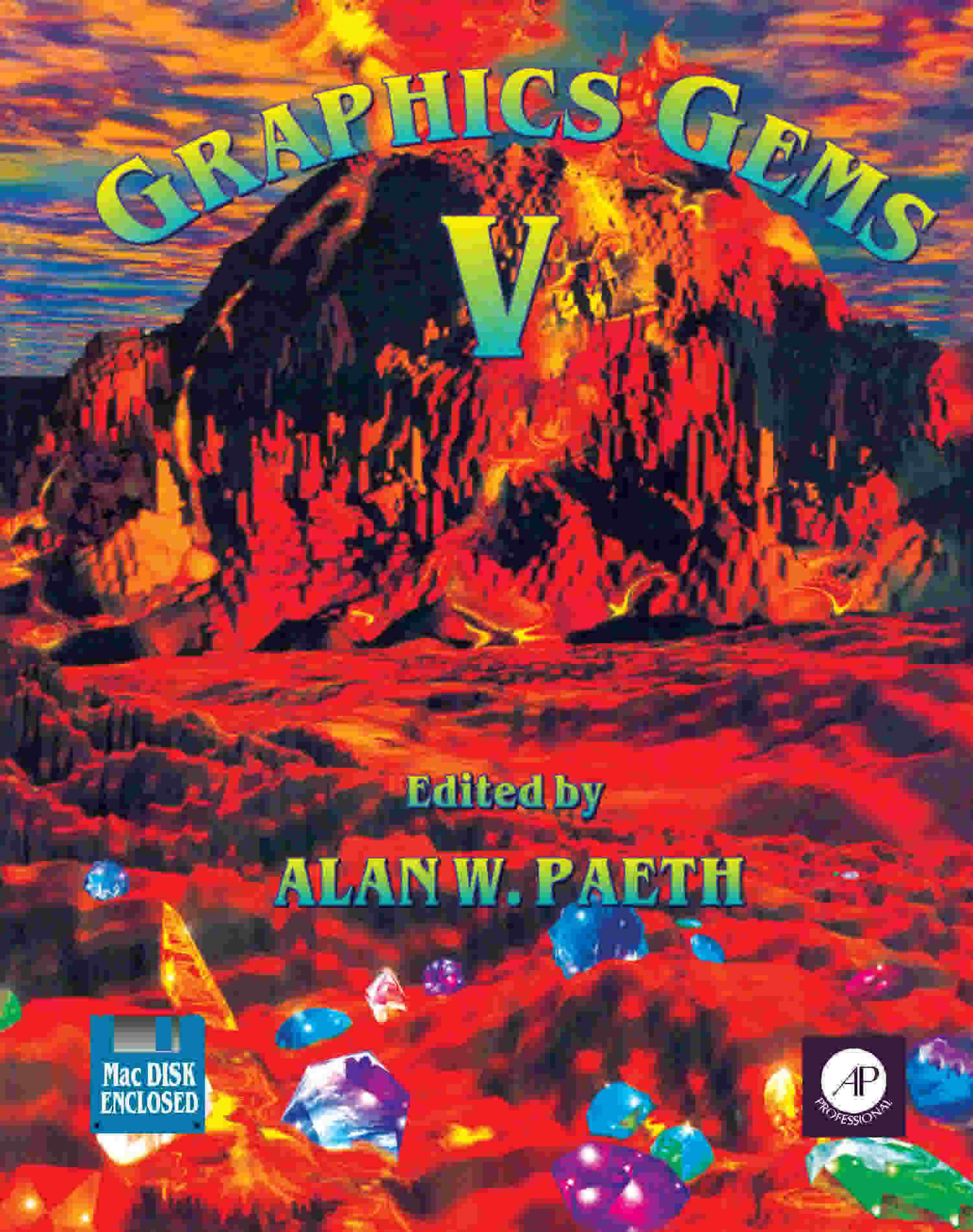
GRAPHICS GEMS V



Edited by

ALAN W. PAETH

Mac DISK ENCLOSED

AP PROFESSIONAL

GRAPHICS GEMS V

This is a volume in

The Graphics Gems Series

A Collection of Practical Techniques for the Computer Graphics Programmer

Series Editor

Andrew Glassner
Microsoft
Redmond, Washington

GRAPHICS GEMS V

Edited by Alan W. Paeth

Computer Science Department
Okanagan University College
Kelowna, British Columbia

AP PROFESSIONAL

Boston San Diego New York
London Sydney Tokyo Toronto

This book is printed on acid-free paper ♾

AP PROFESSIONAL
1300 Boylston Street, Chestnut Hill, MA 02167

An imprint of ACADEMIC PRESS, INC.
A Division of HARCOURT BRACE & COMPANY

United Kingdom Edition published by
ACADEMIC PRESS LIMITED
24–28 Oval Road, London NW1 7DX

Library of Congress Cataloging-in-Publication Data
Graphics gems V / edited by Alan W. Paeth.
p. cm. -- (The graphics gems series)
Includes bibliographical references and index.
ISBN 0-12-543455-3 (with IBM disk : alk paper)
ISBN 0-12-543457-X (with Macintosh disk : alk paper)
1. Computer graphics. I. Paeth, Alan W. II. Graphics gems 5.
III. Title: Graphics gems five. IV. Series.
T385.G6935 1995
006.6'6--dc20 93-41849
CIP

Printed in the United States of America

95 96 97 98 HM 9 8 7 6 5 4 3 2 1

Contents

Foreword

Andrew S. Glassner

Computer graphics exists because people have messages to communicate. As our tools for rendering, modeling, and animation become more sophisticated, we find it ever easier to create meaningful statements. But the tools of graphics are rarely the point of our enterprise; our goal is to enable meaningful communication of important ideas. To create meaning we must make creative choices, and this leads us to the creation of art.

There are many ways to define art, and perhaps no definition will ever work universally. For now, I will use a broad definition that includes all "technical" creations and say that any creative act can result in art, whether it produces a painting, a song, a video showing tidal forces on Saturn, or a daydream. The last example is something created purely to entertain its creator; all other forms of art are vehicles for communication. Every image we produce with computer graphics that is ultimately destined to be shown to another person contains a message: the image is simply the vehicle for communicating that underlying idea. That idea may be very simple (*e.g.*, a restful arrangement of colors), or very complex (*e.g.*, particle flow in turbulent water), but the image is always subservient to the message: without its intended message, the image has no intrinsic value.

For these reasons, I believe that as we develop our tools we must keep in mind how they help people create, refine, and present their ideas. Each new option in a paint program, each new method for interpolating 3D keyframes, and indeed every new technique, should be evaluated in terms of not just its technical performance, but also in terms of whether it improves people's ability to communicate.

The point of view that images exist to carry messages is quite far from the idea that computers should be generating their own images. The concept of computer-generated art (as opposed to computer-assisted art, which is what we have now) has been around as long as computers and science fiction have been around. Sometimes hailed as a good and sometimes couched as a warning, the idea that computers might start creating images, films, sculptures, and other artifacts in the same form as traditional media carries with it some interesting questions for those of us who create images to express our ideas and who create new tools for that purpose.

The computer is the perfect simulator and imitator, but only along one axis of the human experience: intellectual analysis. This is an essential part of what it is to be human, but not the whole thing. It is, however, the only tool at our disposal as creators of new hardware and software, because the computer is inherently a logical, rational device. We have no way of writing an intuitive or spiritual program; those ideas just don't fit into the computer model. We can force these ideas onto the Procrustean bed

of computers and try to create an algorithmic model of intuition, but I believe this does more harm than good: it means distorting the very nature of something not based on reason to codify it using the tools of reason. Perhaps someday there will be a way to emulate intuition and imagination and soul, but I see no hope of doing that with the machines and ideas that form the field of computers as we know them now.

Without these essential human characteristics, a computer by itself cannot produce art that carries anywhere near the levels of meaning that a human artist can provide. An artifact produced by a person carries within it many layers of conscious and unconscious thought, imagination, filtering, selection, phrasing, shaping, and so on. Artists struggle to find the right way to present something, to find the essential core of the message they are communicating. Even practical artists, for example, those who produce images of traffic flow on urban streets, select shapes and colors and compositions that work best, as judged by both objective and subjective criteria. We can try to codify our processes for these selections and judgments, but so many of them happen so deeply inside us that often the best we can do is create a behaviorist's paradise: a book of rules that, when obeyed, usually produces a reasonable result. Music composed by mechanically following the rules of theory is nothing like what a five-year-old makes when banging on a piano, but which has more heart? Which speaks more directly to us as people?

Returning to computer graphics, I believe that the best images and films are the ones that are made by people with something to say, and that we should address our tools to helping those people with their message. We ought not to try to place layers of computer-generated art over their message in order to make it look more sophisticated, creative, or artistic in some way, because this creates information without meaning.

Let us take as an example an imaginary lighting system (unimplemented to my knowledge) that attempts to provide lighting for scene designers. Someone creates an image or animation that appears splotchy; that is, there are some large dark regions and everything else is about evenly lit. The person invokes the lighting system, which inserts a new light to illuminate the dark regions. Is this a good thing? Consider that the new light may create new highlights if the surfaces are shiny—do those highlights draw a viewer's eye away from a region of more importance? Does the new light create shadows that change how the surface appears to move? Is it simply out of place in some way? Perhaps. The computer can't answer these questions, because they are vague and hard to define—two of the characteristics of a problem ill-suited for computerization. It is better to leave the creator of the image to define and place that light than to do it automatically. This has very little to do with expertise and experience, and everything to do with the complex job of trading off countless vague and intuitive decisions when we create anything. Whatever the person decides, it will have been a decision formed and evaluated by someone with intent, and, like the five-year-old on the piano, the message, even if imperfectly stated, is always more important than whether or not the rules were followed. To break the rules we sometimes need tools more powerful than the ones we've had in the past. And when we share those tools, the entire community gains as

we discover each other's insights. Part of the inspiration for the *Graphics Gems* series was to provide some of the small and large tools that would prove useful to creative people working on creative tasks.

So it is with great pleasure that I welcome you to *Graphics Gems V*, a volume of new and useful tools for you to apply to your work as you create images, films, and the systems that help people create them. My goal in this series has been to provide programmers with tools that have been forged by necessity, shaped by experience, and shared through a sense of community.

When I had the original idea for the first *Graphics Gems*, I was inspired by a wallet-sized card one of my college professors carried, which had the entire APL language (with examples!) printed on its two sides. I thought *Gems* would be a small paperback book that you could just carry around casually; in fact, we were uncertain that we could fill enough pages, even with large type and wide margins, to make it financially sound to print the book. The flood of high-quality submissions I received in response to the original solicitation quickly changed that premise, and now we have produced five large, densely packed volumes.

It gives me particular pleasure to note that all of the source code for all the *Gems* books is freely available to the public through many different channels. This is important to me, and I thank AP Professional for supporting this approach. You can now find much of the *Gems* source code on disk and CD-ROM, as well as through anonymous ftp, the World Wide Web, and other Internet servers.

The tools in this book are yours, to extend your reach, conserve your time, and encourage you to reach for ever-higher dreams. Enjoy!

This page intentionally left blank

Preface

As with previous volumes of the *Graphics Gems* series, this book ultimately serves a number of purposes. First, it provides a recognized, moderated forum of computer graphics dialogue, allowing emerging techniques to come to light before a large audience. Where possible, it places evolving methods within their historical context through its choice of entries and through interactions between the technical editor and each contributor. My emphasis on the latter, which took the form of providing citations lists, related articles, and copyediting for many authors, proved to be both a major undertaking and a rewarding task.

Second, the book serves as a means of dissemination and distribution of this information across a broad and secure domain. Today, the contents of this book "in any form and by any means, electronic or mechanical" is circulating in libraries lacking the benefits of Internet access. Tomorrow, it will be in libraries that will abandon that network. I regard my floppy disk from Volume III as both a landmark step in publishing and a 5 1/4″ historical keepsake. [As an electronic document, the diskette included with this book contains code from all five volumes. The original authors have in some cases revised their entries to correct bugs or to cite related work; see, for example, the code that accompanies Volume IV's "Point in Polygon Strategies." This decision in not running previous code verbatim also keeps the diskettes up to publication date with respect to their anonymous FTP mirrors at Princeton.edu (see under /pub/Graphics/GraphicsGems) and elsewhere.]

Finally, the book provides information in a medium that will never be outmoded. Good gems and good books are worthy of rereading simply on their own merit. The best implementations appearing here either transcend the C language in which they were first coded or are presently reembodied in C merely for the time being. Ultimately, this volume is not a summary of past work but a congress of ideas looking toward the electronic frontier.

Notable entries include Herbison-Evans' noniterative root solver, which opens the volume. Its code has perhaps the oldest pedigree of any gem, having begun life on an English Electric KDF9 in Algol-60 before migrating to an IBM 7040 (Fortran), thence to a PDP11/34. Other feature-length entries include the surveys. Chin's illustrative binary space partition "walk-through" is detailed to the point of a complete implementation, making it a welcome contribution for even the casual graphics programmer. Of similar value is the book's concluding survey of four extended graphics libraries. Owing to the extreme code length of these and a few other gems, only excerpts appear in print, though such gems *in toto* may truly be said to exist between the book's covers. Gems lacking code (the other extreme) are more rare; Goldman provides a remarkably concise

summary of curve and surface basis identities annotated with a valuable citations list. Finally, most of the entries in Part VI collectively describe advances in halftoning and image processing at the state of the art that beckon for further experimentation.

The editor wishes to acknowledge two who helped make this work possible: Eric Haines served as an external reviewer for four submissions and also provided editorial assistance in rewriting a portion of one contribution. Special thanks go to MIT's resident expert in communication policy, Dr. Branko Gerovak, who ran a make-shift Mass Ave *sneaker net* one late Cambridge afternoon in early Fall and to the AP PROFESSIONAL staff—Jenifer Niles, sponsoring editor, Cindy Kogut, production editor, and Jacqui Young, editorial assistant—who coordinated and managed the entire project.

◊ Afterword ◊

Five years ago a friend and fellow PARC alumnus conceived of a computer graphics text unlike any previous. A collected work, its appendices would contain full implementations—in C and placed in the public domain—of the algorithms it described. For many of us, Glassner's book offered the perfect niche for the mathematical tools and tricks accumulated over years of graphics programming, whose essential design details would fit neither a short note nor a journal article. Hitherto, our gems-in-the-rough were strewn across the backs of envelopes, among disk subdirectories, and within desk-side shoe boxes. We polished what we had, contributed liberally, then waited. The book proved a runaway success.

An evolution of volumes followed. In the second, Arvo captured many more gems not already in hardback (together, those texts total nearly fifteen hundred pages). Color plates were added. While the form and style of the book remained unchanged per se, the accompanying code was already ensconced on an Internet-based repository at Yale by the time the edition appeared in print.

The third volume retained the color plates while the FTP mirror migrated from Yale to Princeton. More important, the code was reproduced on floppy disk attached to the back cover, wherein it became a physical portion of Kirk's volume. Not coincidentally, a book leading the edge in graphics content was also pushing the envelope in methods of electronic publishing, as suggested by the four ISBN numbers that catalogue both the printed pages and IBM/Macintosh diskettes.

These advances, plus the sizable niche market of literate computer professionals, helped give rise to AP PROFESSIONAL. The fourth volume, edited by Heckbert, became a founding entry. The Internet was more widely employed for manuscript submission as well as correspondence. Accordingly, a standardized typesetting language (LaTeX) was chosen and a book style sheet provided. As a consequence, that volume—and this which follows—underwent an appendectomy in that the code listings now accompany their respective gems. In short, gems publication has became a desktop enterprise for nearly all parties involved.

This is the fifth collection of graphics gems, those practical programming essentials. The fifth volume in a series traditionally provides a summary of work to date. With this in mind, the gems were solicited (electronically) with two requests. First, that they constitute summary works. Second, that they satisfy my benchmark for a good gem: Would the author look up their own work? What came over the transom were over one hundred highly diverse submissions. Herein are four dozen shining examples from contributors who span four continents and who have widely diverse professional backgrounds. While there are only a few summary gems, each entry is unique, at times scintillating, and worth reading carefully many times over, as I have already done.

To the gems!

Alan Paeth
Kelowna, British Columbia

LIMITED WARRANTY AND DISCLAIMER OF LIABILITY

ACADEMIC PRESS, INC. ("AP") AND ANYONE ELSE WHO HAS BEEN INVOLVED IN THE CREATION OR PRODUCTION OF THE ACCOMPANYING CODE ("THE PRODUCT") CANNOT AND DO NOT WARRANT THE PERFORMANCE OR RESULTS THAT MAY BE OBTAINED BY USING THE PRODUCT. THE PRODUCT IS SOLD "AS IS" WITHOUT WARRANTY OF ANY KIND (EXCEPT AS HEREAFTER DESCRIBED), EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY WARRANTY OF PERFORMANCE OR ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. AP WARRANTS ONLY THAT THE MAGNETIC DISKETTE(S) ON WHICH THE CODE IS RECORDED IS FREE FROM DEFECTS IN MATERIAL AND FAULTY WORKMANSHIP UNDER THE NORMAL USE AND SERVICE FOR A PERIOD OF NINETY (90) DAYS FROM THE DATE THE PRODUCT IS DELIVERED. THE PURCHASER'S SOLE AND EXCLUSIVE REMEDY IN THE EVENT OF A DEFECT IS EXPRESSLY LIMITED TO EITHER REPLACEMENT OF THE DISKETTE(S) OR REFUND OF THE PURCHASE PRICE, AT AP'S SOLE DISCRETION.

IN NO EVENT, WHETHER AS A RESULT OF BREACH OF CONTRACT, WARRANTY OR TORT (INCLUDING NEGLIGENCE), WILL AP OR ANYONE WHO HAS BEEN INVOLVED IN THE CREATION OR PRODUCTION OF THE PRODUCT BE LIABLE TO PURCHASER FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PRODUCT OR ANY MODIFICATIONS THEREOF, OR DUE TO THE CONTENTS OF THE CODE, EVEN IF AP HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

Any request for replacement of a defective diskette must be postage prepaid and must be accompanied by the original defective diskette, your mailing address and telephone number, and proof of date of purchase and purchase price. Send such requests, stating the nature of the problem, to Academic Press Customer Service, 6277 Sea Harbor Drive, Orlando, FL 32887, 1-800-321-5068. APP shall have no obligation to refund the purchase price or to replace a diskette based on claims of defects in the nature or operation of the Product.

Some states do not allow limitation on how long an implied warranty lasts, nor exclusions or limitations of incidental or consequential damage, so the above limitations and exclusions may not apply to you. This Warranty gives you specific legal rights, and you may also have other rights which vary from jurisdiction to jurisdiction.

THE RE-EXPORT OF UNITED STATES ORIGIN SOFTWARE IS SUBJECT TO THE UNITED STATES LAWS UNDER THE EXPORT ADMINISTRATION ACT OF 1969 AS AMENDED. ANY FURTHER SALE OF THE PRODUCT SHALL BE IN COMPLIANCE WITH THE UNITED STATES DEPARTMENT OF COMMERCE ADMINISTRATION REGULATIONS. COMPLIANCE WITH SUCH REGULATIONS IS YOUR RESPONSIBILITY AND NOT THE RESPONSIBILITY OF AP.

Author Index

Numbers in parentheses indicate pages on which authors' gems begin.

Jaewoo Ahn (179), *Systems Engineering Research Institute, K/ST, PO Box 1, Yusong, Taejon 305-600, South Korea*

David G. Alciatore (91), *Department of Mechanical Engineering, Colorado State University, Fort Collins, Colorado 80523, dga@lance.colostate.edu*

Louis K. Arata (107), *Picker International, Ohio Imaging, Nuclear Medicine Division, 23130 Miles Road, Bedford Heights, Ohio 44128-5443, arata@nm.picker.com*

Chandrajit Bajaj (191), *Department of Computer Sciences, Purdue University, West Lafayette, Indiana 47906-1398*

Carole Blanc (139), *Laboratoire Bordelais de Recherche en Informatique, 351, cours de la Libération, 33405 Talence, France, blanc@labri.u-bordeaux.fr*

William Bouma (380), *Department of Computer Sciences, Purdue University, West Lafayette, Indiana 47906-1398*

Robert Buckley (65), *Xerox Corporation, MS 0128-27E, 800 Phillips Road, Webster, New York 14580, buckley.wbst128@xerox.com*

Paulo Cezar Pinto Carvalho (42), *Instituto de Matemática Pura e Aplicada, Universidade Federal do Rio de Janeiro, Estrada Dona Castorino, 110, 22460-320 Rio de Janeiro, Brazil, pcezar@visgraf.impa.br*

Paulo Roma Cavalcanti (42), *Instituto de Matemática Pura e Aplicada, Universidade Federal do Rio de Janeiro, Estrada Dona Castorino, 110, 22460-320 Rio de Janeiro, Brazil, proma@visgraf.impa.br*

Norman Chin (121), *Silicon Graphics, Inc., 2011 North Shoreline Boulevard, Mountain View, California 94043, nc@sgi.com*

Robert A. Cross (359), *Department of Computer Science, Indiana University, Lindley Hall 215, Bloomington, Indiana 47405, rcross@cs.indiana.edu*

Luiz Henrique de Figueiredo (173), *Instituto de Matemática Pura e Aplicada, Universidade Federal do Rio de Janeiro, Estrada Dona Castorino, 110, 22460-320 Rio de Janeiro, Brazil, lhf@visgraf.impa.br*

Jean-François Doué (323), *Gerencia Comercial, Aguos Argentinos, Av. Cordoba 1950, CP1120, Buenos Aires, Argentina*

Steven Eker (314), *Department of Computer Science, Brunel University, Uxbridge, Middlesex UB8 3PH, United Kingdom, Steven.Eker@brunel.ac.uk*

Martin Feda (290), *Institute of Computer Graphics, Technical University of Vienna, Karlsplatz 13/186, A-1040 Vienna, Austria, feda@cg.tuwien.ac.at*

Manfred Geiler (297), *Institute of Computer Graphics, Technical University of Vienna, Karlsplatz 13/186, A-1040 Vienna, Austria*

Andrew S. Glassner (50), *Microsoft, 16011 Northeast 36th Way, Redmond, Washington 98052-6399*

Ronald N. Goldman (149, 163), *Department of Computer Science, Rice University, PO Box 1892-MS 132, Houston, Texas 77251-1892, rng@cs.rice.edu*

Jens Gravesen (199), *Mathematical Institute, Technical University of Denmark, Building 303, DK-2800 Lyngby, Denmark*

Daniel Green (375), *Autodesk, Inc., Multimedia Division, 111 McInnis Parkway, San Rafael, California 94903, daniel.green@autodesk.com*

Andrew J. Hanson (55), *Department of Computer Science, Indiana University, Lindley Hall 215, Bloomington, Indiana 47405, hanson@cs.indiana.edu*

Don Hatch (375), *Silicon Graphics, Inc., 2011 North Shoreline Boulevard, Mountain View, California 94043, hatch@sgi.com*

Don Herbison-Evans (3), *Central Queensland University, Bundaberg Campus, PO Box 5424, Bundaberg West, Queensland, Australia 4670, herbisod@musgrave.cqu.edu.au*

Kenneth J. Hill (72), *Evolution Computing, 885 North Granite Reed Road #49, Scottsdale, Arizona 85257, 76667.2576@compuserve.com*

Steve Hill (98), *Computing Laboratory, University of Kent, Canterbury, Kent CT2 7NF, United Kingdom*

Siu-chi Hsu (302, 338), *Computer Science Department, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong, schsu@acm.org*

Raghu Karinthi (398), *Department of Statistics and Computer Science, West Virginia University, PO Box 6330, Knapp Hall, Morgantown, West Virginia 26506, raghu@cs.wvu.edu*

I. H. H. Lee (338), *Creature House, Ltd., Hong Kong, creature@acm.org*

Andreas Leipelt (258), *Mathematisches Seminar der Universität Hamburg, Bundesstrasse 55, D-20146 Hamburg, Germany, leipelt@GEOMAT.math.uni-hamburg.de*

Dinesh Manocha (394), *Department of Computer Science, University of North Carolina at Chapel Hill, CB# 3175, Sitterson Hall, Chapel Hill, North Carolina 27599, manocha@cs.unc.edu*

Gábor Márton (268), *Process Control Department, Technical University of Budapest, Müegyetem Rkp. 9/R, Budapest, H-1111, Hungary, marton@seeger.fsz.bme.hu*

Stephen May (400), *Department of Computer Science, The Ohio State University, Columbus, Ohio 43210, smay@cgrg.ohio-state.edu*

Robert D. Miller (111, 206), *1837 Burrwood Circle, East Lansing, Michigan 48823*

Rick Miranda (91), *Department of Mathematics, Colorado State University, Fort Collins, Colorado 80523*

Tomas Möller (242), *Lund Institute of Technology, Ulrikedalsvagen 4C:314, 224 58 Lund, Sweden, d91tm@efd.lth.se*

Atul Narkhede (394), *Department of Computer Science, University of North Carolina at Chapel Hill, CB# 3175, Sitterson Hall, Chapel Hill, North Carolina 27599, narkhede@cs.unc.edu*

Alan Wm. Paeth, editor (78, 400), *Department of Computer Science, Okanagan University College, 3333 College Way, Kelowna, British Columbia, V1V 1V7 Canada, awpaeth@okanagan.bc.ca*

Werner Purgathofer (297), *Institute of Computer Graphics, Technical University of Vienna, Karlsplatz 13/186, A-1040 Vienna, Austria*

Jonathan C. Roberts (98), *Computing Laboratory, University of Kent, Canterbury, Kent CT2 7NF, United Kingdom*

Ruben Gonzalez Rubio (323), *University of Sherbrooke, 2500 University Boulevard, Sherbrooke GIT 2RE, Quebec, Canada*

Ferdi Scheepers (400), *Department of Computer Science, The Ohio State University, Columbus, Ohio 43210, ferdi@cgrg.ohio-state.edu*

Christophe Schlick (232, 367), *Laboratoire Bordelais de Recherche en Informatique, 351, cours de la Libération, 33405 Talence, France, schlick@labri.u-bordeaux.fr*

Rajesh Sharma (355), *Indiana University, Lindley Hall 310, Bloomington, Indiana 47405, rsharma@cs.indiana.edu*

Ching-Kuang Shene (227), *Department of Math and Computer Science, Northern Michigan University, 1401 Presque Isle Avenue, Marquette, Michigan 49855, shene@nmu.edu*

Ken Shoemake (25, 210), *Computer Science Department, University of Pennsylvania, 220 S. 33rd Street, Philadelphia, Pennsylvania 19104, shoemake@graphics.cis.upenn.edu*

Gilles Subrenat (232), *Laboratoire Bordelais de Recherche en Informatique, 351, cours de la Libération, 33405 Talence, France, subrenat@labri.u-bordeaux.fr*

Robert F. Tobler (297), *Institute of Computer Graphics, Technical University of Vienna, Karlsplatz 13/186, A-1040 Vienna, Austria*

Ken Turkowski (16, 22, 168), *Apple Computer, Inc., 1 Infinite Loop, MS 301-3J, Cupertino, California 95014, turk@apple.com*

Allen Van Gelder (35), *Baskin Computer Science Center, 225 A.S., Computer and Information Sciences, University of California, Santa Cruz, California 95064*

George Vaněček, Jr. (380, 386), *Department of Computer Sciences, Purdue University, West Lafayette, Indiana 47906-1398*

Tien-tsin Wong (302), *Computer Science Department, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong, ttwong@cs.cuhk.hk*

Guoliang Xu (191), *Department of Computer Sciences, Purdue University, West Lafayette, Indiana 47906-1398*

Kurt Zimmerman (285), *Indiana University, Lindley Hall 215, Bloomington, Indiana 47405, kuzimmer@cs.indiana.edu*

◊ I ◊

Algebra and Arithmetic

The gems in this section describe general mathematical techniques having ready application to computer graphics. The methods are crafted with both efficiency and numerical stability in mind.

Herbison-Evans' root finder (I.1) offers the penultimate word in polynomial root finding for computationally closed forms. One immediate gem application generalizes the efficient 3D eigenvalue finder (gem III.2 in volume IV) onto the 4D case. Turkowski (I.2, I.3) provides two elegant and efficient (inverse) square root finders. The first is optimized for use with floating-point hardware and requires no divisions; the second is suitable for integer hardware and features a fixed binary point having adjustable position. Shoemake (I.4) discusses the utility of rational approximation and derives an implementation more stable than one based upon first principles. The availability of his code makes it a useful tool in crafting well-tuned software, as when finding the integer coefficients for the code that concludes gem II.7.

This page intentionally left blank

◊ I.1

Solving Quartics and Cubics for Graphics

Don Herbison-Evans
Central Queensland University
Bundaberg Campus
herbisod@musgrave.cqu.edu.au

◊ Introduction ◊

In principle, quartic and cubic equations can be solved without using iterative techniques. In practice, most numerical algorithms based directly upon analytic solutions of these equations are neither well-behaved nor efficient. This gem[1] derives a robust C-language implementation based upon the solutions of Neumark and Ferrari. Its superiority in controlling both round-off error and overflow is also demonstrated.

◊ Background ◊

Quartic equations need to be solved when ray tracing fourth-degree surfaces, e.g., a torus. Quartics also need to be solved in a number of problems involving quadric surfaces. Quadric surfaces (e.g., ellipsoids, paraboloids, hyperboloids, cones) are useful in computer graphics for generating objects with curved surfaces (Badler and Smoliar 1979). Fewer primitives are required than with planar surfaces to approximate a curved surface to a given accuracy (Herbison-Evans 1982b).

Bicubic surfaces may also be used for the composition of curved objects. They have the advantage of being able to incorporate recurves: lines of inflection. There is a problem, however, when drawing the outlines of bicubics in the calculation of hidden arcs. The visibility of an outline can change where its projection intersects that of another outline. The intersection can be found as the simultaneous solution of the two projected outlines. For bicubic surfaces, these outlines are cubics, and the simultaneous solution of two of these is a sextic which can be solved only by iterative techniques. For quadric surfaces, the projected outlines are quadratic. The simultaneous solution of two of these leads to a quartic equation.

[1]This gem updates a prior technical report (Herbison-Evans 1986).

IBM ISBN 0-12-543455-3
Macintosh ISBN 0-12-543457-X