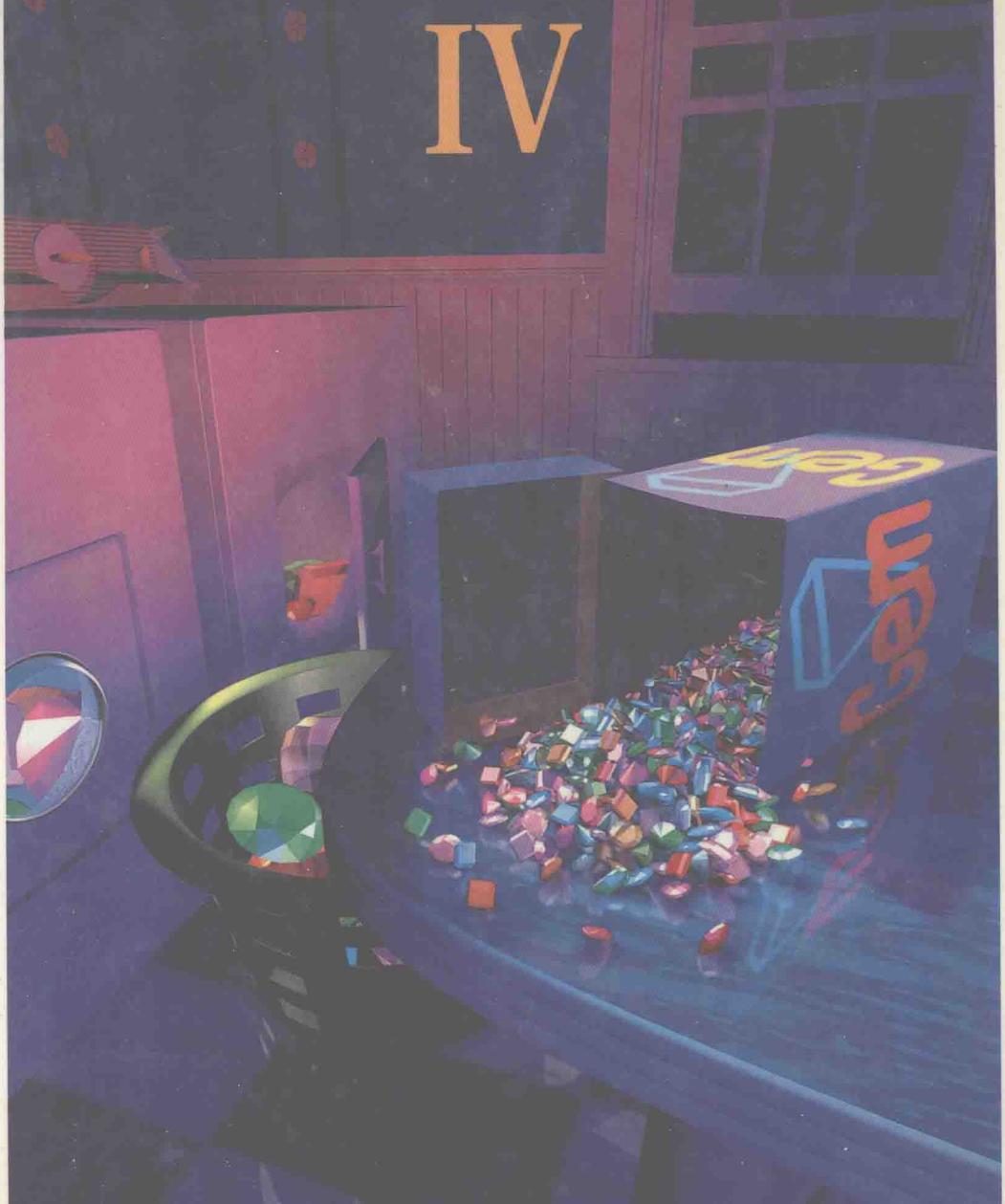
GRAPHICS GEMS

EDITED BY PAUL S. HECKBERT



AP PROFESSIONAL

MAC DISK ENCLOSED

GRAPHICS GEMS IV

Edited by Paul S. Heckbert

Computer Science Department
Carnegie Mellon University
Pittsburgh, Pennsylvania

AP PROFESSIONAL

Boston San Diego New York
London Sydney Tokyo Toronto

This book is printed on acid-free paper ♾

AP PROFESSIONAL
955 Massachusetts Avenue, Cambridge, MA 02139

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 IV / edited by Paul S. Heckbert.
p. cm. -- (The Graphics gems series)
Includes bibliographical references and index.
ISBN 0-12-336156-7 (with Macintosh disk). —ISBN 0-12-336155-9 (with IBM disk)
1. Computer graphics. I. Heckbert, Paul S., 1958– .
II. Title: Graphics gems 4. III. Title: Graphics gems four.
IV. Series.
T385.G6974 1994
006.6'6--dc20 93-46995
CIP

Printed in the United States of America

94 95 96 97 MV 9 8 7 6 5 4 3 2 1

Author Index

Format: *author, institution, chapter number:* p. *start page.*
Author's full address is listed on the first page of each chapter.

Steve Hill, University of Kent, Canterbury, UK, X.1: p. 521.
R. Victor Klassen, Xerox Webster Research Center, Webster, NY, USA, IV.4: p. 261.
Manfred Kopp, Technical University of Vienna, Vienna, Austria, VII.1: p. 413.
Dani Lischinski, Cornell University, Ithaca, NY, USA, I.5: p. 47, IV.5: p. 278.
Joe Marks, Digital Equipment Corporation, Cambridge, MA, USA, IX.1: p. 497.
Henry Massalin, Microunity Corporation, Sunnyvale, CA, USA, VIII.3: p. 447.
Robert D. Miller, E. Lansing, MI, USA II.4: p. 132.
Yoshikazu Ohashi, Cognex, Needham, MA, USA, II.2: p. 120.
Alan W. Paeth, Okanagan University College, Kelowna, British Columbia, Canada, VIII.6: p. 486.
John W. Peterson, Taligent, Inc., Cupertino, CA, USA, IV.6: p. 286.
Rich Rabbitz, Martin Marietta, Moorestown, NJ, USA, I.8: p. 83.
John Schlag, Industrial Light and Magic, San Rafael, CA, USA, VIII.1: p. 433.
Christophe Schlick, Laboratoire Bordelais de Recherche en Informatique, Talence, France, VI.1: p. 385, VI.3: p. 401, VII.3: p. 422.
Peter Schorn, ETH, Zürich, Switzerland, I.2: p. 7.
Ching-Kuang Shene, Northern Michigan University, Marquette, MI, USA, IV.7: p. 321, V.1: p. 353.
Stuart Shieber, Harvard University, Cambridge, MA, USA, IX.1: p. 497.
Peter Shirley, Indiana University, Bloomington, IN, USA, V.4: p. 370.
Ken Shoemake, University of Pennsylvania, Philadelphia, PA, USA, III.1: p. 175, III.4: p. 207, III.5: p. 222, III.6: p. 230.
László Szirmay-Kalos, Technical University of Budapest, Budapest, Hungary, IX.2: p. 505.
Tim Van Hook, Silicon Graphics, Mountain View, CA, USA, II.3: p. 125.
Warren N. Waggenspack, Jr., Louisiana State University, Baton Rouge, LA, USA, V.2: p. 356.
Changyaw Wang, Indiana University, Bloomington, IN, USA, V.4: p. 370.
Greg Ward, Lawrence Berkeley Laboratory, Berkeley, CA, USA, VII.2: p. 415.
Kevin Weiler, Autodesk Inc., Sausalito, CA, USA, I.3: p. 16.
George Wolberg, City College of New York/CUNY, New York, NY, USA, VIII.3: p. 447.
Andrew Woo, Alias Research, Inc., Toronto, Ontario, Canada, VI.2: p. 388.
Kevin Wu, SunSoft, Mountain View, CA, USA, III.3: p. 199.
Guoliang Xu, Purdue University, West Lafayette, IN, USA, IV.3: p. 256.
Karel Zuiderveld, Utrecht University, Utrecht, The Netherlands, VIII.5: p. 474.

Foreword

Andrew S. Glassner

We make images to communicate. The ultimate measure of the quality of our images is how well they communicate information and ideas from the creator's mind to the perceiver's mind. The efficiency of this communication, and the quality of our image, depends on both what we want to say and to whom we intend to say it.

I believe that computer-generated images are used today in two distinct ways, characterized by whether the intended receiver of the work is a person or machine. Images in these two categories have quite different reasons for creation, and need to satisfy different criteria in order to be successful.

Consider first an image made for a machine. For example, an architect planning a garden next to a house may wish to know how much light the garden will typically receive per day during the summer months. To determine this illumination, the architect might build a 3D model of the house and garden, and then use computer graphics to simulate the illumination on the ground at different times of day in a variety of seasons. The images generated by the rendering program would be a by-product, and perhaps never even looked at; they were only generated in order to compute illumination. The only criterion for judgment for such images is an appropriate measure of *accuracy.*

Nobody will pass judgment on the *aesthetics* of these pictures, since no person with an aesthetic sense will ever see them. Accuracy does not require beauty. For example, a simulation may not produce images that are individually correct, but instead average to the correct answer. The light emitted by the sun may be modeled as small, discrete chunks, causing irregular blobs of illumination on the garden. When these blobs are averaged together over many hours and days, the estimates approach the correct value for the received sunlight. No one of these pictures is accurate individually, and probably none of them would be very attractive.

When we make images for people, we have a different set of demands. We almost always require that our images be *attractive* in some way. In this context, attractive does not necessarily mean beautiful, but it means that there must be an aesthetic component influenced by composition, color, weight, and so on. Even when we intend to act as analytic and dispassionate observers, humans have an innate sense of beauty that cannot be denied. This is the source of all ornament in art, music, and literature: we always desire something beyond the purely functional. Even the most utilitarian objects, such as hammers and pencils, are *designed* to provide grace and beauty to our eyes and offer comfort to our hands. When we weave together beauty and utility, we create elegance. People are more interested in beautiful things than neutral things, because they stimulate our senses and our feelings.

So even the most utilitarian image intended to communicate something to another person must be designed with that person in mind: the picture must be composed so that it is balanced in terms of form and space, the colors must harmonize, the shapes must not jar. It is by occasionally violating these principles that we can make one part of the image stand out with respect to the background; ignoring them produces images that have no focus and no balance, and thus do not capture and hold our interest. Their ability to communicate is reduced. Every successful creator of business charts, wallpaper designs, and scientific visualizations knows these rules and works with them.

So images intended for people must be attractive. Only then can we further address the idea of accuracy. What does it mean for an image intended for a person to be "accurate"?

Sometimes "accuracy" is interpreted to mean that the energy of the visible light calculated to form the image exactly matches the energy that would be measured if the modeled scene (including light sources) really existed, and were photographed; this idea is described in computer graphics by the term *photorealism.* This would certainly be desirable, under some circumstances, if the image were intended for a machine's analysis, but the human perceptual apparatus responds differently than a flatbed scanner. People are not very good at determining absolute levels of light, and we are easily fooled into thinking that the brightest and least chromatic part of an image is "white."

Again we return to the question of what we're trying to communicate. If the point of an image is that a garden is well-lit and that there is uniform illumination over its entire surface, then we do not care about the radiometric accuracy of the image as much as the fact that it conveys that information; the whole picture could be too bright or too dark by some constant factor and this message will still be carried without distortion. In the garden image, we expect a certain variation due to the variety of soil, rocks, plants, and other geometry in the scene. Very few people could spot the error in a good but imprecise approximation of such seemingly random fluctuation. In this type of situation, if you can't see the error, you don't care about it. So not only can the illumination be off by a constant factor, it can vary from the "true" value quite a bit from point to point and we won't notice, or if we do notice, we won't mind.

If we want to convey the sense of a scene viewed at night, then we need to take into account the entire observer of a night scene. The human visual system *adapts* to different light levels, which changes how it perceives different ranges of light. If we look at a room lit by a single 25-watt light bulb, and then look at it again when we use a 1000-watt bulb, the overall illumination has changed by a constant factor, but our perception of the room changes in a non-linear way. The room lit by the 25-watt bulb appears dark and shadowy, while the room lit by the 1000-watt bulb is stark and bright. If we display both on a CRT using the same intensity range, even though the underlying radiance values were computed with precision, both images will appear the same. Is this either *accurate* or *photorealistic*?

Sometimes some parts of an image intended for a person must be accurate, depending

on what that image is intended to communicate. If the picture shows a new object intended for possible manufacture, the precise shape may be important, or the way it reflects light may be critical. In these applications we are treating the person as a machine; we are inviting the person to analyze one or more characteristics of the image as a predictor of a real object or scene. When we are making an image of a smooth and glossy object prior to manufacture in order to evaluate its appearance, the shading must match that of the final object as accurately as possible. If we are only rendering the shape in order to make sure it will fit into some packing material, the shading only needs to give us information about the shape of the object; this shading may be arbitrarily inaccurate as long as we still get the right perception of shape. A silver candlestick might be rendered as though it were made of concrete, for example, if including the highlights and caustics would interfere with judging its shape. In this case our definition of "accuracy" involves our ability to judge the structure of shapes from their images, and does not include the optical properties of the shape.

My point is that images made for machines should be judged by very different criteria than images made for people. This can help us evaluate the applicability of different types of images with different objective accuracies. Consider the picture generated for an architect's client, with the purpose of getting an early opinion from the client regarding whether there are enough trees in the yard. The accuracy of this image doesn't matter as long as it looks good and is roughly correct in terms of geometry and shading. Too much precision in every part of the image may lead to too much distraction; because of its perceived realism and implied finality, the client may start thinking about whether a small shed in the image is placed just right, when it hasn't even been decided that there will be a shed at all. Precision implies a statement; vagueness implies a suggestion.

Consider the situation where someone is evaluating a new design for a crystal drinking glass; the precision of the geometry and the rendering will matter a great deal, since the reflections and sparkling colors are very important in this situation. But still, the numerical accuracy of the energy simulation need not be right, as long as the relative accuracy of the image is correct. Then there's the image made as a simulation for analysis by a machine. In this case the image must be accurate with respect to whatever criteria will be measured and whatever choice of measurement is used.

Images are for communication, and the success of an image should be measured only by how well it communicates. Sometimes too little objective accuracy can distort the message; sometimes too much accuracy can detract from the message. The reason for making a picture is to communicate something that must be said; the image should support that message and not dominate it. The medium must be chosen to fit the message.

To make effective images we need effective tools, and that is what this book is intended to provide. Every profession has its rules of thumb and tricks of the trade; in computer graphics, these bits of wisdom are described in words, equations, and programs. The

Graphics Gems series is like a general store; it's fun to drop in every once in a while and browse, uncovering unusual items with which you were unfamiliar, and seeing new applications for old ideas. When you're faced with a sticky problem, you may remember seeing just the right tool on display. Happily, our stock is in limitless supply, and as near as your bookshelf or library.

Preface

This book is a cookbook for computer graphics programmers, a kind of "Numerical Recipes" for graphics. It contains practical techniques that can help you do 2D and 3D modeling, animation, rendering, and image processing. The 52 articles, written by 54 authors worldwide, have been selected for their usefulness, novelty, and simplicity. Each article, or "Gem," presents a technique in words and formulas, and also, for most of the articles, in C or C++ code as well. The code is available in electronic form on the IBM or Macintosh floppy disk in the back pocket of the book, and is available on the Internet via FTP (see address below). The floppy disk also contains all of the code from the previous volumes: *Graphics Gems I, II,* and *III.* You are free to use and modify this code in any way you like.

A few of the Gems in this book deserve special mention because they provide implementations of particularly useful, but non-trivial algorithms. Gems IV.6 and IV.8 give very general, modular code to polygonize parametric and implicit surfaces, respectively. With these two and a polygon renderer, you could probably display 95% of all computer graphics models! Gem I.5 finds 2D Voronoi diagrams or Delaunay triangulations. These data structures are very widely used for mesh generation and other geometric operations. In the area of interaction, Gem III.1 provides code for control of orientation in 3D. This could be used in interactive 3D modelers. Finally, Gem I.8 gives code to find collisions of polyhedra, an important task in physically based modeling and animation.

This book, like the previous three volumes in the *Graphics Gems* series, lies somewhere between the media of textbook, journal, and computer bulletin board. Textbooks explain algorithms very well, but if you are doing computer graphics programming, then they may not provide what you need: an implementation. Similarly, technical journals seldom present implementations, and they are often much more theoretical than a programmer cares for. The third alternative, computer bulletin boards such as the USENET news group *comp.graphics.algorithms*, occasionally contains good code, but because they are unmoderated and unedited, they are so flooded with queries that it is tedious to find useful information. The *Graphics Gems* series is an attempt at a middle ground, where programmers worldwide can contribute graphics techniques that they have found useful, and the best of these get published. Most of the articles are written by the inventors of the techniques, so you will learn their motivations and see their programming techniques firsthand. Also, the implementations have been selected for their portability; they are not limited to UNIX, IBM, or Macintosh systems. Most of them will compile and run, perhaps with minor modifications, on any computer with a C or C++ compiler.

Assembling this book has been a collaborative process involving many people. In the Spring of 1993, a call for contributions was distributed worldwide via electronic mail and word of mouth. Submissions arrived in the Summer of 1993. These were read by me and many were also read by one or more of my outside reviewers: Eric Haines, Andrew Glassner, Chandrajit Bajaj, Tom Duff, Ron Goldman, Tom Sederberg, David Baraff, Jules Bloomenthal, Ken Shoemake, Mike Kass, Don Mitchell, and Greg Ward. Of the 155 articles submitted, 52 were accepted for publication. These were revised and, in most cases, formatted into LaTeX by the authors. Coordinating the project at Academic Press in Cambridge, Massachusetts, were Jenifer Niles and Brian Miller. Book composition was done by Rena Wells at Rosenlaui Publishing Services in Houston, Texas, and the cover image was made by Eben Ostby of Pixar, in Richmond, California. I am very thankful to all of these people and to the others who worked on this book for helping to make it a reality. Great thanks also to the *Graphics Gems* series editor, Andrew Glassner, for inviting me to be editor for this volume, and to my wife, Bridget Johnson-Heckbert, for her patience.

There are a few differences between this book and the previous volumes of the series. Organizationally, the code and bibliographies are not collected at the back of the book, but appear with the text of the corresponding article. These changes make each Gem more self-contained. The book also differs in emphasis. Relative to the previous volumes, I have probably stressed novelty more, and simplicity less, preferring an implementation of a complex computer graphics algorithm over formulas from analytic geometry, for example.

In addition to the *Graphics Gems* series, there are several other good sources for practical computer graphics techniques. One of these is the column "Jim Blinn's Corner" that appears in the journal *IEEE Computer Graphics and Applications*. Another is the book *A Programmer's Geometry*, by Adrian Bowyer and John Woodwark (Butterworth's, London, 1983), which is full of analytic geometry formulas. A mix of analytic geometry and basic computer graphics formulas is contained in the book *Computer Graphics Handbook: Geometry and Mathematics* by Michael E. Mortensen (Industrial Press, New York, 1990). Another excellent source is, of course, graphics textbooks.

Code in this book is available on the Internet by anonymous FTP from `princeton.edu` (`128.112.128.1`) in the directory `pub/GraphicsGems/GemsIV`. The code for other *Graphics Gems* books is also available nearby. Bug reports should be submitted as described in the `README` file there.

Paul Heckbert, March 1994

About the Cover

The cover: "Washday Miracle" by Eben Ostby. Copyright © 1994 Pixar.

When series editor Andrew Glassner called me to ask if I could help with a cover image for *Graphics Gems IV*, there were four requirements: the image needed to tell a story; it needed to have gems in it; it should be a computer-generated image; and it should look good. To these parameters, I added one of my own: it should tell a story that is different from the previous covers. Those stories were usually mystical or magical; accordingly, I decided to take the mundane as my inspiration.

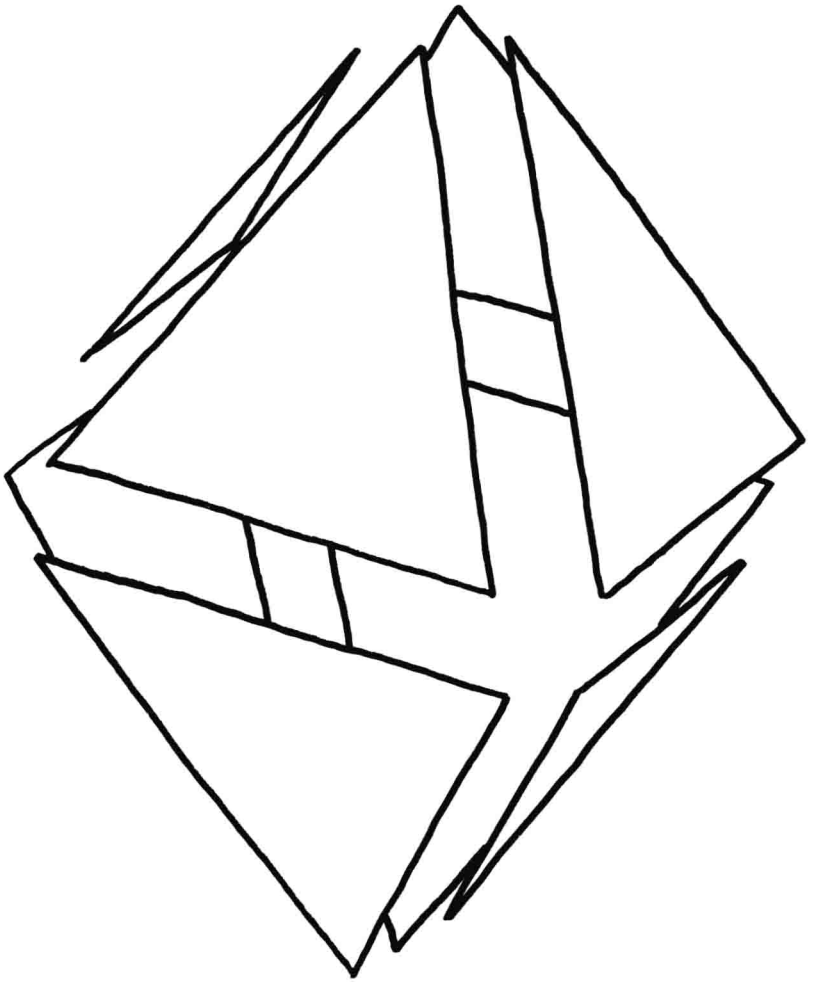
The image was created using a variety of tools, including Alias Studio; Menv, our own internal animation system; and Photorealistic RenderMan. The appliances, table, and basket were built in Alias. The gems were placed by a stochastic "gem-placer" running under Menv. The house set was built in Menv. Surface descriptions were written in the RenderMan shading language and include both procedural and painted textures.

For the number-conscious, this image was rendered at a resolution of 2048 by 2695 and contains the following:

16 lights
643 gems
30,529 lines or 2,389,896 bytes of model information
4 cycles: regular, delicate, Perma-Press, and Air Fluff

Galyn Susman did the lighting design. Andrew Glassner reviewed and critiqued, and made the image far better as a result. Matt Martin made prepress proofs. Pixar (in corpora Karen Robert Jackson and Ralph Guggenheim) permitted me time to do this.

Eben Ostby
Pixar



Contents

Polygons and Polyhedra

This part of the book contains five Gems on polygons and three on polyhedra. Polygons and polyhedra are the most basic and popular geometric building blocks in computer graphics.

I.1. Centroid of a Polygon, *by Gerard Bashein and Paul R. Detmer.*
Gives formulas and code to find the centroid (center of mass) of a polygon. This is useful when simulating Newtonian dynamics. Page 3.

I.2. Testing the Convexity of a Polygon, *by Peter Schorn and Frederick Fisher.*
Gives an algorithm and code to determine if a polygon is convex, non-convex (concave but not convex), or non-simple (self-intersecting). For many polygon operations, faster algorithms can be used if the polygon is known to be convex. This is true when scan converting a polygon and when determining if a point is inside a polygon, for instance. Page 7.

I.3. An Incremental Angle Point in Polygon Test, *by Kevin Weiler.*

I.4. Point in Polygon Strategies, *by Eric Haines.*
Provide algorithms for testing if a point is inside a polygon, a task known as point inclusion testing in computational geometry. Point-in-polygon testing is a basic task when ray tracing polygonal models, so these methods are useful for 3D as well as 2D graphics. Weiler presents a single algorithm for testing if a point lies in a concave polygon, while Haines surveys a number of algorithms for point inclusion testing in both convex and concave polygons, with empirical speed tests and practical optimizations. Pages 16 and 24.