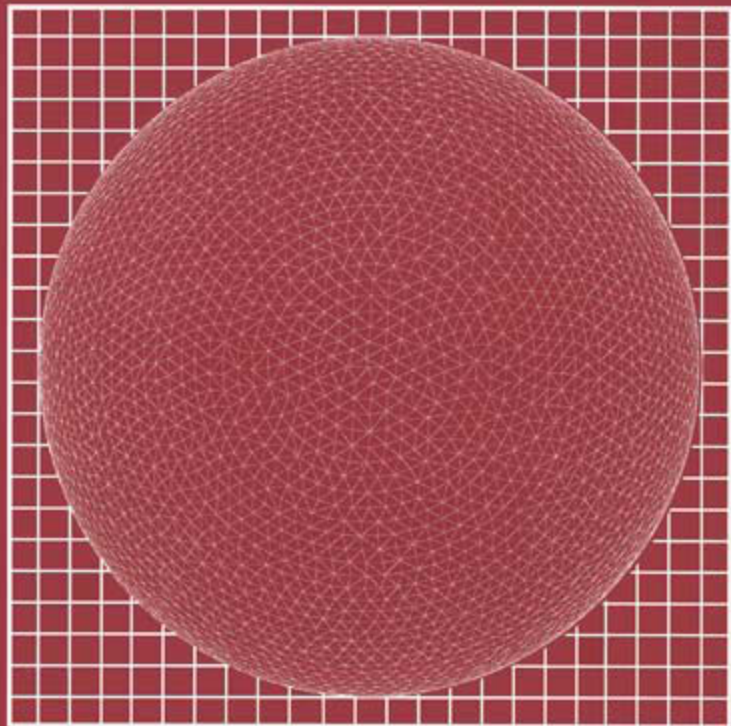


COMPUTATIONAL GEOMETRY IN C

SECOND EDITION



JOSEPH O'ROURKE

COMPUTATIONAL GEOMETRY IN C, Second Edition

This is the most suitable textbook for introducing undergraduate students and newcomers to the design of geometry algorithms, which arise in many practical areas such as computer graphics, robotics, and engineering design. The basic techniques used in computational geometry are all covered: polygon triangulations, convex hulls, Voronoi diagrams, arrangements, geometric searching, and motion planning. The self-contained treatment presumes only an elementary knowledge of mathematics, but it reaches topics on the frontier of current research. Thus professional programmers will also find it a useful tutorial.

This newly revised and expanded edition contains material on several new topics, including randomized algorithms for polygon triangulation, planar point location, and 3D convex hull construction, and intersection algorithms for ray-segment, ray-triangle, and point in polyhedron. A new “Sources” chapter pointing to supplemental literature provides an easy upgrade path to the graduate level. Numerous exercises are provided at the end of every section; a partial solutions manual is available.

A novel aspect of the book is the inclusion of working C code for many of the algorithms, with discussion of practical implementation issues. The code in this new edition is significantly improved from the first edition (more efficient and more robust), and four new routines are included. Java versions for all code are also available. The code is accessible from the book’s Web site (<http://cs.smith.edu/~orourke/>) or by anonymous ftp.

From reviews of the first edition. . .

“The book . . . conveys the feeling that computational geometry is interesting, exciting, important, and very active.” – Scot Drysdale, SIGACT News.

“The author does an excellent job in maintaining the balance between basic geometric concepts and advanced research results.” – M.-S. Kim, Zentralblatt für Mathematik.

“Teaching a course from this book must be a distinct pleasure.” – Christoph M. Hoffmann, SIAM Review.

Joseph O’Rourke is Spencer T. and Ann W. Olin Professor of Computer Science at Smith College. He received his PhD in Computer Science from the University of Pennsylvania and was Assistant then Associate Professor at The Johns Hopkins University, 1980–1988, before joining the Smith faculty in 1987. In addition to this textbook, he has published the monograph *Art Gallery Theorems and Algorithms* (Oxford, 1987) and coedited with J. E. Goodman the 1000-page *Handbook of Discrete and Computational Geometry* (CRC Press LLC, 1997). He is the author of over seventy papers in the field of computational geometry, as well as thirty-three “Computational Geometry Columns.” He is also the current `comp.graphics.algorithms` FAQ maintainer.

COMPUTATIONAL GEOMETRY IN C

SECOND EDITION

JOSEPH O'ROURKE



CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town,
Singapore, São Paulo, Delhi, Tokyo, Mexico City

Cambridge University Press
32 Avenue of the Americas, New York NY 10013-2473, USA

www.cambridge.org

Information on this title: www.cambridge.org/9780521640107

© Cambridge University Press 1994, 1998

This publication is in copyright. Subject to statutory exception
and to the provisions of relevant collective licensing agreements,
no reproduction of any part may take place without the written
permission of Cambridge University Press.

First published 1994
Second edition 1998
12th printing 2008

A catalogue record for this publication is available from the British Library.

Library of Congress Cataloging in Publication Data

O'Rourke, Joseph.

Computational geometry in C / Joseph O'Rourke, – 2nd ed.
p. cm.

Includes bibliographical references.

ISBN 0-521-64010-5 – ISBN 0-521-64976-5 (pbk.)

1. Geometry – Data processing. 2. C (Computer program language).

I. Title.

CA448.D38 1998

516'.0285'5133 – dc21

98-15363

CIP

ISBN 978-0-521-64976-6 Paperback

Cambridge University Press has no responsibility for the persistence or
accuracy of URLs for external or third-party internet websites referred to in
this publication, and does not guarantee that any content on such websites is,
or will remain, accurate or appropriate.

To Marylynn, Nell, and Russell

Contents

Preface	<i>page x</i>
1. Polygon Triangulation	1
1.1 Art Gallery Theorems	1
1.2 Triangulation: Theory	11
1.3 Area of Polygon	16
1.4 Implementation Issues	24
1.5 Segment Intersection	27
1.6 Triangulation: Implementation	32
2. Polygon Partitioning	44
2.1 Monotone Partitioning	44
2.2 Trapezoidalization	47
2.3 Partition into Monotone Mountains	57
2.4 Linear-Time Triangulation	56
2.5 Convex Partitioning	58
3. Convex Hulls in Two Dimensions	63
3.1 Definitions of Convexity and Convex Hulls	64
3.2 Naive Algorithms for Extreme Points	66
3.3 Gift Wrapping	68
3.4 QuickHull	69
3.5 Graham's Algorithm	72
3.6 Lower Bound	87
3.7 Incremental Algorithm	88
3.8 Divide and Conquer	91
3.9 Additional Exercises	96
4. Convex Hulls in Three Dimensions	101
4.1 Polyhedra	101
4.2 Hull Algorithms	109
4.3 Implementation of Incremental Algorithm	117
4.4 Polyhedral Boundary Representations	146
4.5 Randomized Incremental Algorithm	149
4.6 Higher Dimensions	150
4.7 Additional Exercises	153

5. Voronoi Diagrams	155
5.1 Applications: Preview	155
5.2 Definitions and Basic Properties	157
5.3 Delaunay Triangulations	161
5.4 Algorithms	165
5.5 Applications in Detail	169
5.6 Medial Axis	179
5.7 Connection to Convex Hulls	182
5.8 Connection to Arrangements	191
6. Arrangements	193
6.1 Introduction	193
6.2 Combinatorics of Arrangements	194
6.3 Incremental Algorithm	199
6.4 Three and Higher Dimensions	201
6.5 Duality	201
6.6 Higher-Order Voronoi Diagrams	205
6.7 Applications	209
6.8 Additional Exercises	218
7. Search and Intersection	220
7.1 Introduction	220
7.2 Segment–Segment Intersection	220
7.3 Segment–Triangle Intersection	226
7.4 Point in Polygon	239
7.5 Point in Polyhedron	245
7.6 Intersection of Convex Polygons	252
7.7 Intersection of Segments	263
7.8 Intersection of Nonconvex Polygons	266
7.9 Extreme Point of Convex Polygon	269
7.10 Extremal Polytope Queries	272
7.11 Planar Point Location	285
8. Motion Planning	294
8.1 Introduction	294
8.2 Shortest Paths	295
8.3 Moving a Disk	300
8.4 Translating a Convex Polygon	302
8.5 Moving a Ladder	313
8.6 Robot Arm Motion	322
8.7 Separability	339
9. Sources	347
9.1 Bibliographies and FAQs	347
9.2 Textbooks	347
9.3 Book Collections	348

Contents

ix

9.4	Monographs	349
9.5	Journals	349
9.6	Conference Proceedings	350
9.7	Software	350
Bibliography		351
Index		361

Preface

Computational geometry broadly construed is the study of algorithms for solving geometric problems on a computer. The emphasis in this text is on the design of such algorithms, with somewhat less attention paid to analysis of performance. I have in several cases carried out the design to the level of working C programs, which are discussed in detail.

There are many brands of geometry, and what has become known as “computational geometry,” covered in this book, is primarily discrete and combinatorial geometry. Thus polygons play a much larger role in this book than do regions with curved boundaries. Much of the work on continuous curves and surfaces falls under the rubrics of “geometric modeling” or “solid modeling,” a field with its own conferences and texts,¹ distinct from computational geometry. Of course there is substantial overlap, and there is no fundamental reason for the fields to be partitioned this way; indeed they seem to be merging to some extent.

The field of computational geometry is a mere twenty years old as of this writing, if one takes M. I. Shamos’s thesis (Shamos 1978) as its inception. Now there are annual conferences, journals, texts, and a thriving community of researchers with common interests.

Topics Covered

I consider the “core” concerns of computational geometry to be polygon partitioning (including triangulation), convex hulls, Voronoi diagrams, arrangements of lines, geometric searching, and motion planning. These topics form the chapters of this book. The field is not so settled that this list can be considered a consensus; other researchers would define the core differently.

Many textbooks include far more material than can be covered in one semester. This is not such a text. I usually cover about 80% of the text with undergraduates in one 40 class-hour semester and all of the text with graduate students. In order to touch on each of the core topics, I find it necessary to oscillate the level of detail, only sketching some algorithms while detailing others. Which ones are sketched and which detailed is a personal choice that I can only justify by my classroom experiences.

Prerequisites

The material in this text should be accessible to students with only minimal preparation. Discrete mathematics, calculus, and linear algebra suffice for mathematics. In fact very

¹E.g., Hoffmann (1989) and Mortenson (1990).

little calculus or linear algebra is used in the text, and the enterprising student can learn the little needed on the fly. In computer science, a course in programming and exposure to data structures is enough (Computer Science I and II at many schools). I do not presume a course in algorithms, only familiarity with the “big- O ” notation. I teach this material to college juniors and seniors, mostly computer science and mathematics majors.

I hasten to add that the book can be fruitfully studied by those who have no programming experience at all, simply by skipping all the implementation sections. Those who know some programming language, but not C, can easily appreciate the implementation discussions even if they cannot read the code. All code is available in Java as well as C, although only C is discussed in the body of the text.

When teaching this material to both computer science and mathematics majors, I offer them a choice of projects that permits those with programming skills to write code and those with theoretical inclinations to avoid programming.

Although written to be accessible to undergraduates, my experience is that the material can form the basis of a challenging graduate course as well. I have tried to mix elementary explanations with references to the latest literature. Footnotes provide technical details and citations. A number of the exercises pose open problems. It is not difficult to supplement the text with research articles drawn from the 300 bibliographic references, effectively upgrading the material to the graduate level.

Implementations

Not all algorithms discussed in the book are provided with implementations. Full code for twelve algorithms is included:²

- Area of a polygon.
- Triangulating a polygon.
- Convex hull in two dimensions.
- Convex hull in three dimensions.
- Delaunay triangulation.
- Segment/ray–segment intersection.
- Segment/ray–triangle intersection.
- Point in polygon.
- Point in polyhedron.
- Intersecting convex polygons.
- Minkowski convolution with a convex polygon.
- Multilink robot arm reachability.

Researchers in industry coming to this book for working code for their favorite algorithms may be disappointed: They may seek an algorithm to find the minimum spanning circle for a set of points and find it as an exercise.³ The presented code should be viewed as samples of geometry programs. I hope I have chosen a representative set of algorithms to implement; much room is left for student projects.

²The distribution also includes code to generate random points in a cube (Figure 4.14), random points on a sphere (Figure 4.15), and uniformly distributed points on a sphere (the book cover image).

³Exercise 5.5.6[12].

All the C code in the book is available by anonymous ftp from `cs.smith.edu` (131.229.222.23), in the directory `/pub/compgeom`.⁴ I regularly update the files in this directory to correct errors and incorporate improvements. The Java versions of all programs are in the same directory.

Exercises

There are approximately 250 exercises sprinkled throughout the text. They range from easy extensions of the text to quite difficult problems to “open” problems. These latter are an exciting feature of such a fresh field; students can reach the frontier of knowledge so quickly that even undergraduates can hope to solve problems that no one has managed to crack yet. Indeed I have written several papers with undergraduates as a result of their work on homework problems I posed.⁵ Not all open problems are necessarily difficult; some are simply awaiting the requisite attention.

Exercises are sporadically marked “[easy]” or “[difficult],” but the lack of these notations should not be read to imply that neither apply. Those marked “[programming]” require programming skills, and those marked “[open]” are unsolved problems as far as I know at this writing. I have tried to credit authors of individual problems where appropriate. Instructors may contact me for a partial solutions manual.

Second Edition Improvements

It is a law of nature that second editions are longer than first editions, and this book is no exception: It is about fifty pages longer, with fifty new exercises, thirty new figures, and eighty additional bibliographic references. All the code from the first edition is significantly improved: All programs now produce Postscript output, all have been translated to Java, many are simpler and/or logically cleaner, most are more robust in the face of degeneracies and numerical error, and most run (sometimes) significantly faster. Both the polygon triangulation code and the Delaunay triangulation code are now $O(n^2)$.

Four new programs have been included: for computing Delaunay triangulation from the three-dimensional convex hull (Section 5.7.4), for intersecting a ray with a triangle in 3-space (Section 7.3), for deciding if a point is inside a polyhedron (Section 7.5), for computing the convolution (Minkowski sum) of a convex polygon with a general polygon (Section 8.4.4), as well as the point generation code that produced the cover image.

New sections are included on partitioning into monotone mountains (Section 2.3), randomized triangulation (Section 2.4.1), the ultimate(?) planar convex hull algorithm (Section 3.8.4), randomized convex hull in three dimensions (Section 4.5), the twin edge data structure (Section 4.4), intersection of a segment and triangle (Section 7.3), the point-in-polyhedron problem (Section 7.5), the Bentley–Ottmann algorithm for intersecting segments (Section 7.7), computing Boolean operations between two polygons (Section 7.8), randomized trapezoidal decomposition for point location (Section 7.11.4), Minkowski convolution computation (Section 8.4.4), and a list of sources for further reading (Chapter 9).

⁴Connect with `ftp cs.smith.edu` and use the name `anonymous`. Or access the files via `http://cs.smith.edu/~orourke`.

⁵The material from one paper is incorporated into Section 7.6.

Other sections are greatly improved, including those on QuickHull (Section 3.4), Graham's algorithm (Section 3.5.5), volume overflow (Section 4.3.5), Delaunay triangulation via the paraboloid transformation (Section 5.7.4), the point-in-polygon problem (Section 7.4), intersecting two segments (Section 7.2), and the implementation of convex polygon intersection (Section 7.6.1).

Acknowledgments

I have received over six hundred e-mail messages from readers of the first edition of this book, and I despair of accurately apportioning credit to their many specific contributions to this edition. I deeply appreciate the suggestions of the following people, many of whom are my professional colleagues, twenty-nine whom are my former students, but most of whom I have met only electronically: Pankaj Agarwal, Kristy Anderson, Bill Baldwin, Michael Baldwin, Brad Barber, Pierre Beauchemin, Ed Bolson, Helen Cameron, Joanne Cannon, Roy Chien, Satyan Coorg, Glenn Davis, Adlai DePano, Matthew Diaz, Tamala Dirlam, David Dobkin, Susan Dorward, Scot Drysdale, Herbert Edelsbrunner, John Ellis, William Flis, Steve Fortune, Robert Fraczkiwicz, Reinaldo Garcia, Sharmilli Ghosh, Carole Gitlin, Jeff Glickman, Jacob E. Goodman, Michael Goodrich, Thom Goodsell, Horst Greiner, Suleyman Guleyupoglu, Eric Haines, Daniel Halperin, Eszter Hargittai, Paul Heckbert, Claudio Heckler, Paul Heffernan, Kevin Hemsteter, Christoph Hoffmann, Rob Hoffmann, Chun-Hsiung Huang, Knut Hunstad, Ferran Hurtado, Joan Hutchinson, Andrei Iones, Chris Johnston, Martin Jones, Amy Josefczyk, Martin Kerscher, Ed Knorr, Nick Korneenko, John Kutcher, Eugene Lee, David Lubinsky, Joe Malkevitch, Michelle Maurer, Michael McKenna, Thomas Meier, Walter Meyer, Simon Michael, Jessica Miller, Andy Mirzaian, Joseph Mitchell, Adelene Ng, Steve Oudot, Seongbin Park, Irena Pashchenko, Octavia Petrovici, Madhav Ponamgi, Don Power, Ari Rappoport, Jennifer Rippel, Christopher Saunders, Catherine Schevon, Christian Schmidt, Peter Schorn, Vadim Shapiro, Thomas Shermer, Paul Short, Saul Simhon, Steve Skiena, Kenneth Sloan, Stephen Smeulders, Evan Smyth, Sharon Solms, Ted Stern, Ileana Streinu, Vinita Subramanian, Dan Sunday, J.W.H. Tangelder, Yi Tao, Seth Teller, Godfried Toussaint, Christopher Van Wyk, Gert Vetger, Jim Ward, Susan Weller, Wendy Welsh, Rephael Wenger, Gerry Wiener, Bob Williamson, Lang Withers, Stacia Wyman, Min Xu, Dianna Xu, Chee Yap, Amy Yee, Wei Yinong, Lilla Zollei, and the Faculty Advancement in Mathematics 1992 Workshop participants. My apologies for the inevitable omissions.

Lauren Cowles at Cambridge has been the ideal editor. I have received generous support from the National Science Foundation for my research in computational geometry, most recently under grant CCR-9421670.

Joseph O'Rourke
orourke@cs.smith.edu
<http://cs.smith.edu/~orourke>
Smith College, Massachusetts
December 23, 1997

Note on the Cover:

The cover image shows the convex hull of 5,000 points distributed on a spiral curve on the surface of a sphere. It was generated by running the `spiral.c` and `chull.c` code distributed with this book: `spiral 5000 -r1000 | chull`.

Polygon Triangulation

1.1. ART GALLERY THEOREMS

1.1.1. Polygons

Much of computational geometry performs its computations on geometrical objects known as polygons. Polygons are a convenient representation for many real-world objects; convenient both in that an abstract polygon is often an accurate model of real objects and in that it is easily manipulated computationally. Examples of their use include representing shapes of individual letters for automatic character recognition, of an obstacle to be avoided in a robot's environment, or of a piece of a solid object to be displayed on a graphics screen. But polygons can be rather complicated objects, and often a need arises to view them as composed of simpler pieces. This leads to the topic of this and the next chapter: partitioning polygons.

Definition of a Polygon

A *polygon* is the region of a plane bounded by a finite collection of line segments¹ forming a simple closed curve. Pinning down a precise meaning for the phrase “simple closed curve” is unfortunately a bit difficult. A topologist would say that it is the homeomorphic image of a circle,² meaning that it is a certain deformation of a circle. We will avoid topology for now and approach a definition in a more pedestrian manner, as follows.

Let $v_0, v_1, v_2, \dots, v_{n-1}$ be n points in the plane. Here and throughout the book, all index arithmetic will be mod n , implying a cyclic ordering of the points, with v_0 following v_{n-1} , since $(n-1) + 1 \equiv 0 \pmod{n}$. Let $e_0 = v_0v_1, e_1 = v_1v_2, \dots, e_i = v_iv_{i+1}, \dots, e_{n-1} = v_{n-1}v_0$ be n segments connecting the points. Then these segments bound a polygon iff³

1. The intersection of each pair of segments adjacent in the cyclic ordering is the single point shared between them: $e_i \cap e_{i+1} = v_{i+1}$, for all $i = 0, \dots, n-1$.
2. Nonadjacent segments do not intersect: $e_i \cap e_j = \emptyset$, for all $j \neq i+1$.

¹A *line segment* ab is a closed subset of a line contained between two points a and b , which are called its *endpoints*. The subset is closed in the sense that it includes the endpoints. (Many authors use \overline{ab} to indicate this segment.)

²A *circle* is a one-dimensional set of points. We reserve the term *disk* to mean the two-dimensional region bounded by a circle.

³“Iff” means “if and only if,” a convenient abbreviation popularized by Halmos (1985, p. 403).

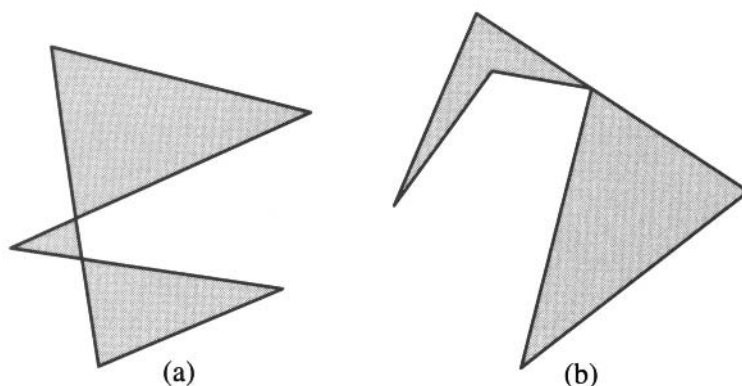


FIGURE 1.1 Nonsimple polygons.

The reason these segments define a *curve* is that they are connected end to end; the reason the curve is *closed* is that they form a cycle; the reason the closed curve is *simple* is that nonadjacent segments do not intersect.

The points v_i are called the *vertices* of the polygon, and the segments e_i are called its *edges*. Note that a polygon of n vertices has n edges.

An important theorem of topology is the Jordan Curve Theorem:

Theorem 1.1.1 (Jordan Curve Theorem). *Every simple closed plane curve divides the plane into two components.*

This strikes most as so obvious as not to require a proof; but in fact a precise proof is quite difficult.⁴ We will take it as given. The two parts of the plane are called the *interior* and *exterior* of the curve. The exterior is unbounded, whereas the interior is bounded. This justifies our definition of a polygon as the region bounded by the collection of segments. Note that we define a polygon P as a closed region of the plane. Often a polygon is considered to be just the segments bounding the region, and not the region itself. We will use the notation ∂P to mean the boundary of P ; this is notation borrowed from topology.⁵ By our definition, $\partial P \subseteq P$.

Figure 1.1 illustrates two nonsimple polygons. For both objects in the figure, the segments satisfy condition (1) above (adjacent segments share a common point), but not condition (2): nonadjacent segments intersect. Such objects are often called polygons, with those polygons satisfying (2) called *simple polygons*. As we will have little use for nonsimple polygons in this book, we will drop the redundant modifier.

We will follow the convention of listing the vertices of a polygon in counterclockwise order, so that if you walked along the boundary visiting the vertices in that order (a *boundary traversal*), the interior of the polygon would be always to your left.

⁴See, e.g., Henle (1979, pp. 100–3). The theorem dates back to 1877.

⁵There is a sense in which the boundary of a region is like a derivative, so it makes sense to use the partial derivative symbol ∂ .

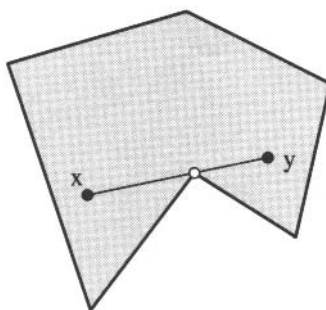


FIGURE 1.2 Grazing contact of line of sight.

1.1.2. The Art Gallery Theorem

Problem Definition

We will study a fascinating problem posed by Klee⁶ that will lead us naturally into the issue of triangulation, the most important polygon partitioning. Imagine an art gallery room whose floor plan can be modeled by a polygon of n vertices. Klee asked: How many stationary guards are needed to guard the room? Each guard is considered a fixed point that can see in every direction, that is, has a 2π range of visibility.⁷ Of course a guard cannot see through a wall of the room. An equivalent formulation is to ask how many point lights are needed to fully illuminate the room. We will make Klee's problem rigorous before attempting an answer.

Visibility

To make the notion of visibility precise, we say that point x can *see* point y (or y is *visible* to x) iff the closed segment xy is nowhere exterior to the polygon P : $xy \subseteq P$. Note that this definition permits the line-of-sight to have grazing contact with a vertex, as shown in Figure 1.2. An alternative, equally reasonable definition would say that a vertex can block vision; say that x has *clear visibility* to y if $xy \subseteq P$ and $xy \cap \partial P \subseteq \{x, y\}$. We will occasionally use this alternative definition in exercises (Exercises 1.1.4[2] and [3]).

A guard is a point. A set of guards is said to *cover* a polygon if every point in the polygon is visible to some guard. Guards themselves do not block each other's visibility. Note that we could require the guards to see only points of ∂P , for presumably that is where the paintings are! This is an interesting variant, explored in Exercise 1.1.4[1].

Max over Min Formulation

We have now made most of Klee's problem precise, except for the phrase "How many." Succinctly put, the problem is to find the maximum over all polygons of n vertices, of the minimum number of guards needed to cover the polygon. This max-over-min formulation is confusing to novices, but it is used quite frequently in mathematics, so we will take time to explain it carefully.

⁶Posed in 1973, as reported by Honsberger (1976). The material in this section (and more on the topic) may be found in O'Rourke (1987).

⁷We will use radians throughout to represent angles. π radians = 180° .

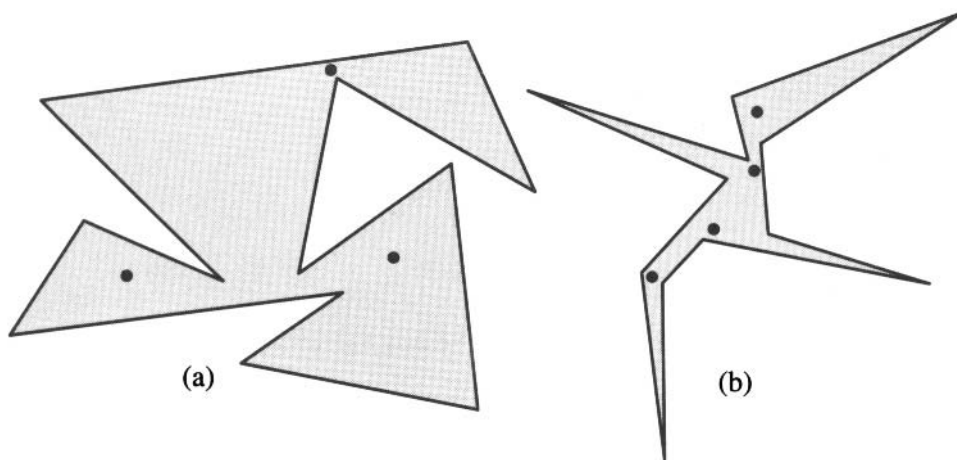


FIGURE 1.3 Two polygons of $n = 12$ vertices: (a) requires 3 guards; (b) requires 4 guards.

For any given fixed polygon, there is some minimum number of guards that are *necessary* for complete coverage. Thus in Figure 1.3(a), it is clear that three guards are needed to cover this polygon of twelve vertices, although there is considerable freedom in the location of the three guards. But is three the most that is ever needed for all possible polygons of twelve vertices? No: the polygon in Figure 1.3(b), also with twelve vertices, requires four guards. What is the largest number of guards that any polygon of twelve vertices needs? We will show eventually that four guards always *suffice* for any polygon of twelve vertices. This is what Klee's question seeks: Express as a function of n , the smallest number of guards that suffice to cover any polygon of n vertices. Sometimes this number of guards is said to be *necessary and sufficient* for coverage: necessary in that at least that many are needed for *some* polygons, and sufficient in that that many always suffice for *any* polygon.

We formalize the problem before exploring it further. Let $g(P)$ be the smallest number of guards needed to cover polygon P : $g(P) = \min_S |\{S : S \text{ covers } P\}|$, where S is a set of points, and $|S|$ is the cardinality⁸ of S . Let P_n be a polygon of n vertices. $G(n)$ is the maximum of $g(P_n)$ over all polygons of n vertices: $G(n) = \max_{P_n} g(P_n)$. Klee's problem is to determine the function $G(n)$. It may not be immediately evident that $G(n)$ is defined for each n : It is at least conceivable that for some polygon, no finite number of guards suffice. Fortunately, $G(n)$ is finite for all n , as we will see. But whether it can be expressed as a simple formula, or must be represented by an infinite table of values, is less clear.

Empirical Exploration

Sufficiency of n . Certainly at least one guard is always necessary. In terms of our notation, this provides a lower bound on $G(n)$: $1 \leq G(n)$. It seems obvious that n guards suffice for any polygon: stationing a guard at every vertex will certainly cover the polygon. This

⁸The *cardinality* of a set is its number of elements.

provides an upper bound: $G(n) \leq n$. But it is not even so clear that n guards suffice. At the least it demands a proof. It turns out to be true, justifying intuition, but this success of intuition is tempered by the fact that the same intuition fails in three dimensions: Guards placed at every vertex of a polyhedron do not necessarily cover the polyhedron! (See Exercise 1.1.4[6].)

There are many art-gallery-like problems, and for most it is easiest to first establish a lower bound on $G(n)$ by finding generic examples showing that a large number of guards are sometimes necessary. When it seems that no amount of ingenuity can increase the number necessary, then it is time to turn to proving that that number is also sufficient. This is how we will proceed.

Necessity for Small n . For small values of n , it is possible to guess the value of $G(n)$ with a little exploration. Clearly every triangle requires just one guard, so $G(3) = 1$.

Quadrilaterals may be divided into two groups: convex quadrilaterals and quadrilaterals with a reflex vertex. Intuitively a polygon is convex if it has no dents. This important concept will be explored in detail in Chapter 3. A vertex is called *reflex*⁹ if its internal angle is strictly greater than π ; otherwise a vertex is called *convex*.¹⁰ A convex quadrilateral has four convex vertices. A quadrilateral can have at most one reflex vertex, for reasons that will become apparent in Section 1.2. As Figure 1.4(a) makes evident, even quadrilaterals with a reflex vertex can be covered by a single guard placed near that vertex. Thus $G(4) = 1$.

For pentagons the situation is less clear. Certainly a convex pentagon needs just one guard, and a pentagon with one reflex vertex needs only one guard for the same reason as in a quadrilateral. A pentagon can have two reflex vertices. They may be either adjacent or separated by a convex vertex, as in Figures 1.4(c) and (d); in each case one guard suffices. Therefore $G(5) = 1$.

Hexagons may require two guards, as shown in Figure 1.4(e) and (f). A little experimentation can lead to a conviction that no more than two are ever needed, so that $G(6) = 2$.

Necessity of $\lfloor n/3 \rfloor$

At this point the reader might be able to leap to a generalization of Figure 1.4(f) for larger values of n . Figure 1.5 illustrates the design for $n = 12$; note the relation to Figure 1.4(f). This “comb” shape consists of k prongs, with each prong composed of two edges, and adjacent prongs separated by an edge. Associating each prong with the separating edge to its right, and the bottom edge with the rightmost prong, we see that a comb of k prongs has $n = 3k$ edges (and therefore vertices). Because each prong requires its own guard, we establish with this one example that $n/3 \leq G(n)$ for $n = 3k$. This is what I meant earlier by saying that a generic example can be used to establish a lower bound on $G(n)$.

⁹Often this is called *concave*, but the similarity of “concave” and “convex” invites confusion, so I will use “reflex.”

¹⁰Some authors use “convex” to indicate what I’ll call *strict convexity*, an interior angle strictly less than π .

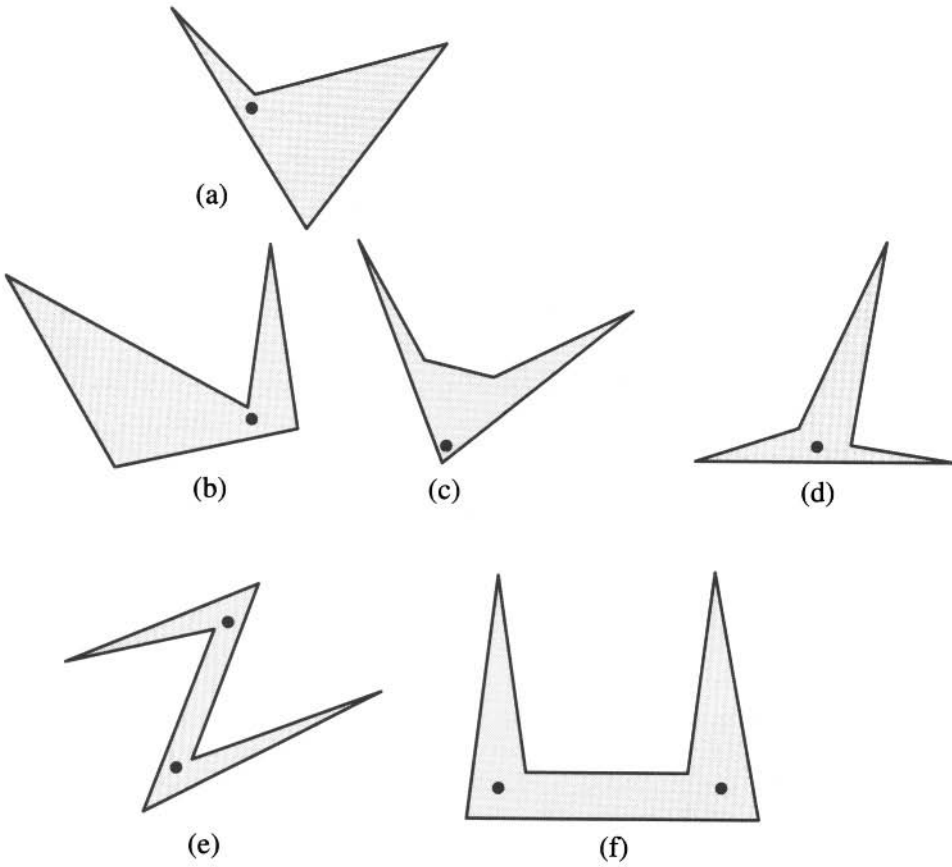


FIGURE 1.4 Polygons of $n = 4, 5, 6$ vertices.

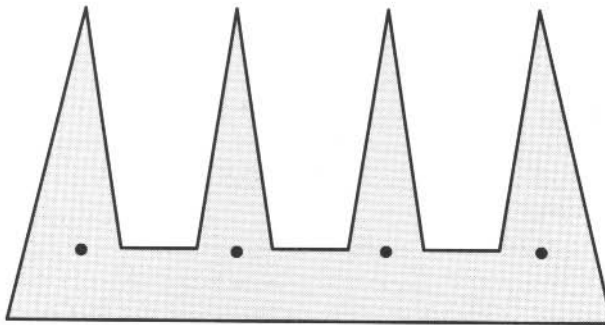


FIGURE 1.5 Chvátal's comb for $n = 12$.

Noticing that $G(3) = G(4) = G(5)$ might lead one to conjecture that $G(n) = \lfloor n/3 \rfloor$,¹¹ and in fact this conjecture turns out to be true. This is the usual way that such mathematical questions are answered: First the answer is conjectured after empirical exploration,

¹¹ $\lfloor x \rfloor$ is the *floor* of x : the largest integer less than or equal to x . The floor function has the effect of discarding the fractional portion of a positive real number.

and only then, with a definite goal in mind, is the result proven. We now turn to a proof.

1.1.3. Fisk's Proof of Sufficiency

The first proof that $G(n) = \lfloor n/3 \rfloor$ was due to Chvátal (1975). His proof is by induction: Assuming that $\lfloor n/3 \rfloor$ guards are needed for all $n < N$, he proves the same formula for $n = N$ by carefully removing part of the polygon so that its number of vertices is reduced, applying the induction hypothesis, and then reattaching the removed portion. The proof splinters into a number of cases and is quite delicate.

Three years later Fisk found a very simple proof, occupying just a single journal page (Fisk 1978). We will present Fisk's proof here.

Diagonals and Triangulation

Fisk's proof depends crucially on partitioning a polygon into triangles with diagonals. A *diagonal* of a polygon P is a line segment between two of its vertices a and b that are clearly visible to one another. Recall that this means the intersection of the closed segment ab with ∂P is exactly the set $\{a, b\}$. Another way to say this is that the open segment from a to b does not intersect ∂P ; thus a diagonal cannot make grazing contact with the boundary.

Let us call two diagonals *noncrossing* if their intersection is a subset of their endpoints: They share no interior points. If we add as many noncrossing diagonals to a polygon as possible, the interior is partitioned into triangles. Such a partition is called a *triangulation* of a polygon. The diagonals may be added in arbitrary order, as long as they are legal diagonals and noncrossing. In general there are many ways to triangulate a given polygon. Figure 1.6 shows two triangulations of a polygon of $n = 14$ vertices.

We will defer a proof that every polygon can be triangulated to Section 1.2, and for now we just assume the existence of a triangulation.

Three Coloring

To prove sufficiency of $\lfloor n/3 \rfloor$ guards for *any* polygon, the proof must work for an arbitrary polygon. So assume an arbitrary polygon P of n vertices is given. The first step of Fisk's proof is to triangulate P . The second step is to "recall" that the resulting graph may be 3-colored. We need to explain what this graph is, and what 3-coloring means.

Let G be a graph associated with a triangulation, whose arcs are the edges of the polygon and the diagonals of the triangulation, and whose nodes are the vertices of the polygon. This is the graph used by Fisk. A *k-coloring* of a graph is an assignment of k colors to the nodes of the graph, such that no two nodes connected by an arc are assigned the same color. Fisk claims that every triangulation graph may be 3-colored. We will again defer a proof of this claim, but a little experimentation should make it plausible. Three-colorings of the triangulations in Figure 1.6 are shown in Figure 1.7. Starting at, say, the vertex indicated by the arrow, and coloring its triangle arbitrarily with three colors, the remainder of the coloring is completely forced: There are no other free choices. Roughly, the reason this always works is that the forced choices never double back on an earlier choice; and the reason this never happens is that the underlying figure is a polygon (with no holes, by definition).

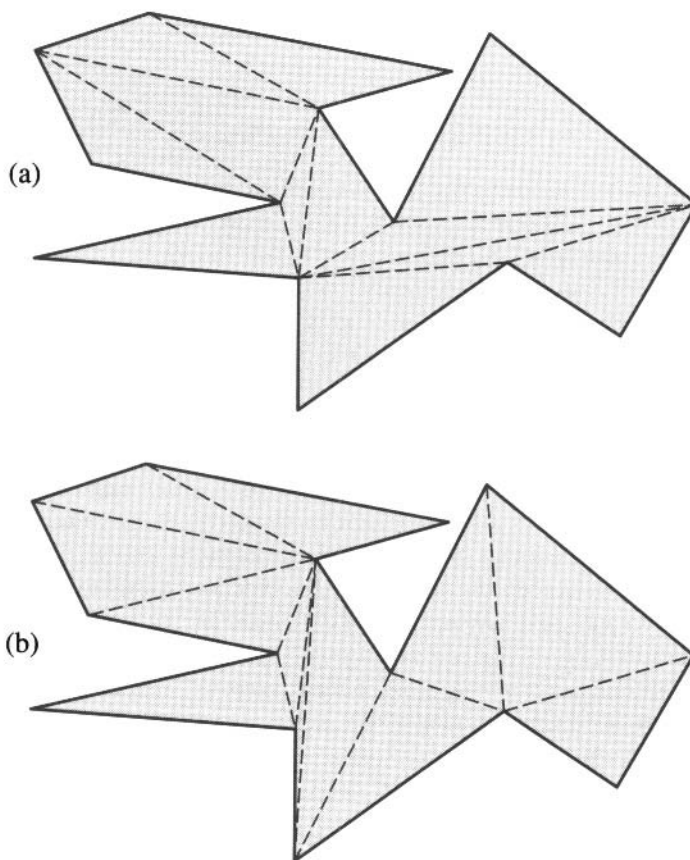


FIGURE 1.6 Two triangulations of a polygon of $n = 14$ vertices.

The third step of Fisk's proof is the observation that placing guards at all the vertices assigned one color guarantees visibility coverage of the polygon. His reasoning is as follows. Let red, green, and blue be the colors used in the 3-coloring. Each triangle must have each of the three colors at its three corners. Thus every triangle has a red node at one corner. Suppose guards are placed at every red node. Then every triangle has a guard in one corner. Clearly a triangle is covered by a guard at one of its corners. Thus every triangle is covered. Finally, the collection of triangles in a triangulation completely covers the polygon. Thus the entire polygon is covered if guards are placed at red nodes. Similarly, the entire polygon is covered if guards are placed at green nodes or at blue nodes.

The fourth and final step of Fisk's proof applies the "pigeon-hole principle": If n objects are placed into k pigeon holes, then at least one hole must contain no more than n/k objects. For if each one of the k holes contained more than n/k objects, the total number of objects would exceed n . In our case, the n objects are the n nodes of the triangulation graph, and the k holes are the 3 colors. The principle says that one color must be used no more than $n/3$ times. Since n is an integer, we can conclude that one color is used no more than $\lfloor n/3 \rfloor$ times. We now have our sufficiency proof: Just place

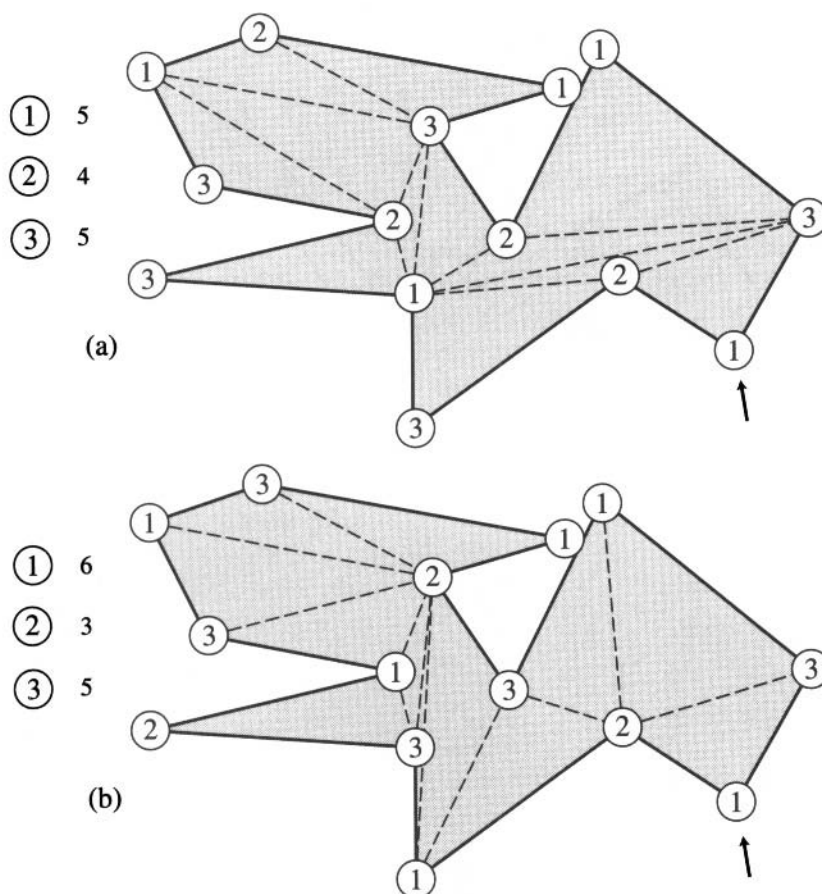


FIGURE 1.7 Two 3-colorings of a polygon of $n = 14$ vertices, based on the triangulations shown in Figure 1.6.

guards at nodes colored with the least-frequently used color in the 3-coloring. We are guaranteed that this will cover the polygon with no more than $G(n) = \lfloor n/3 \rfloor$ colors.

If you don't find this argument beautiful (or at least charming), then you will not enjoy much in this book!

In Figure 1.7, $n = 14$, so $\lfloor n/3 \rfloor = 4$. In (a) of the figure color 2 is used four times; in (b), the same color is used only three times. Note that the 3-coloring argument does not always lead to the most efficient use of guards.

1.1.4. Exercises

1. *Guarding the walls.* Construct a polygon P and a placement of guards such that the guards see every point of ∂P , but there is at least one point interior to P not seen by any guard.
2. *Clear visibility, point guards.* What is the answer to Klee's question for clear visibility (Section 1.1.2)? More specifically, let $G'(n)$ be the smallest number of *point guards* that suffice to clearly see every point in any polygon of n vertices. Point guards are guards who may stand at any point of P ; these are distinguished from *vertex guards* who may be stationed only at vertices.

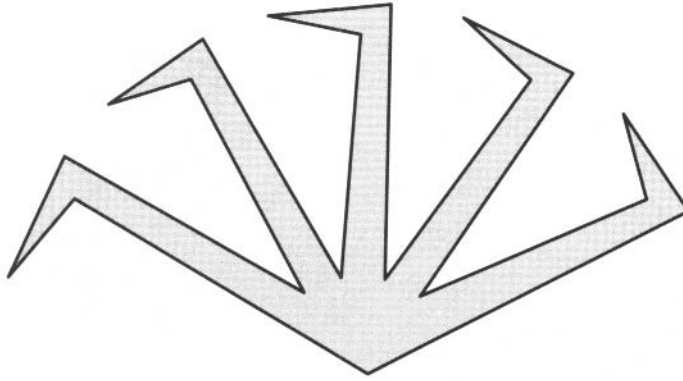


FIGURE 1.8 $\lfloor n/4 \rfloor$ edge guards are necessary (Toussaint).

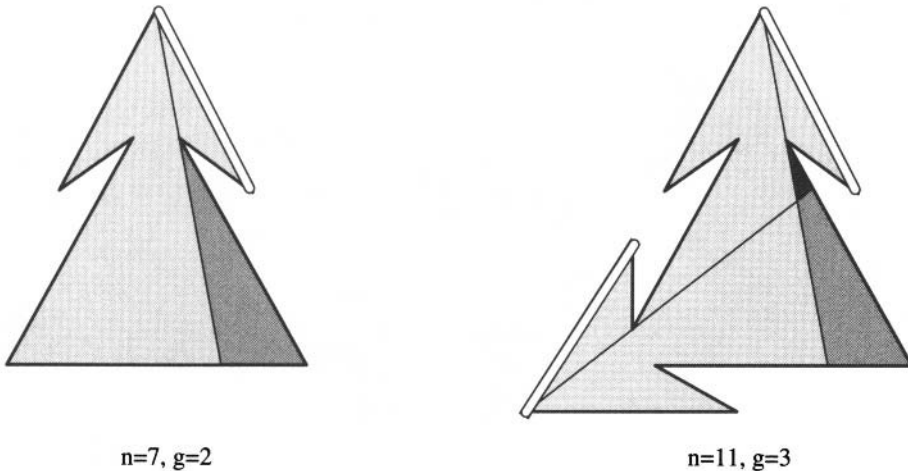


FIGURE 1.9 Two polygons that require $\lfloor (n + 1)/4 \rfloor$ edge guards.

Are clearly seeing guards stronger or weaker than usual guards? What relationship between $G'(n)$ and $G(n)$ follows from their relative strength? ($G(n)$ is defined in Section 1.1.2) Does Fisk's proof establish $\lfloor n/3 \rfloor$ sufficiency for clear visibility? Try to determine $G'(n)$ exactly.

3. *Clear visibility, vertex guards* (Thomas Shermer). Answer question 2, but for *vertex* guards: guards restricted to vertices.
4. *Edge guards* [open]. An *edge guard* is a guard who may patrol one edge e of a polygon. A point $y \in P$ is covered by the guard if there is some point $x \in e$ such that x can see y . Another way to view this is to imagine a fluorescent light whose extent matches e . The portion of P that is illuminated by this light is the set of points covered by the edge guard.

Toussaint showed that $\lfloor n/4 \rfloor$ edge guards are sometimes necessary, as demonstrated by the "half-swastika" polygon shown in Figure 1.8 (O'Rourke 1987, p. 83). He conjectured that $\lfloor n/4 \rfloor$ suffice except for a few small values of n . This odd exception is necessitated by the two "arrowhead" polygons shown in Figure 1.9, which do not seem to generalize. These examples are taken from Shermer (1992).

Prove or disprove Toussaint's conjecture.

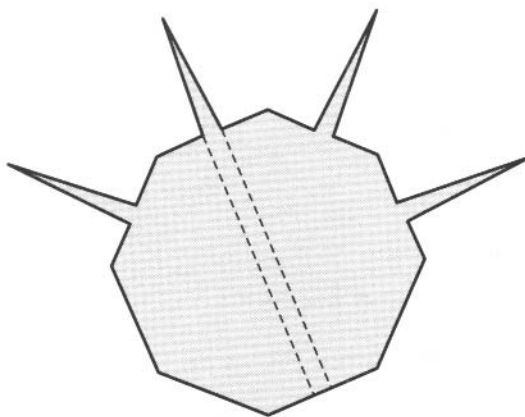


FIGURE 1.10 $\lfloor n/5 \rfloor$ edge guards are necessary (Toussaint).

5. *Edge guards in star polygons* [open]. A *star polygon* is one that can be covered by a single (point) guard. Toussaint proved that $\lfloor n/5 \rfloor$ edge guards are sometimes necessary to cover a star polygon with the example shown in Figure 1.10 (O'Rourke 1987, p. 119). The conjecture that $\lfloor n/5 \rfloor$ always suffice was shown to be false for $n = 14$ (Subramaniyam & Diwan 1991), but otherwise little is known. Prove or disprove that $n/5 + c$ suffice for some constant $c > 0$.
6. *Guards in polyhedra*. Design a polyhedron such that guards placed at every vertex fail to completely cover the interior. A polyhedron is a three-dimensional version of a polygon, composed of polygonal faces, and enclosing a volume. A precise definition is offered in Chapter 4 (Section 4.1). *Hint*: See O'Rourke (1987, Sec. 10.2.2).

1.2. TRIANGULATION: THEORY

In this section we prove that every polygon has a triangulation, and we establish some basic properties of triangulations. In later sections (1.4–1.6.5) we will discuss algorithms for constructing triangulations.

A natural reaction on being presented with the question, “Must every polygon have a triangulation?” is to respond with another question: “How could a polygon *not* have a triangulation?” Indeed it cannot not have one! But if you feel this is too obvious for a proof, consider the equivalent question in three dimensions: There the natural generalization is false! See O'Rourke (1987, p. 253–4).

1.2.1. Existence of a Diagonal

The key to proving the existence of a triangulation is proving the existence of a diagonal. Once we have that, the rest will follow easily. For the proof, we need one other even more obvious fact: Every polygon must have at least one strictly convex vertex.¹²

Lemma 1.2.1. *Every polygon must have at least one strictly convex vertex.*

¹²Recall that a (nonstrict) convex vertex could be collinear with its adjacent vertices.

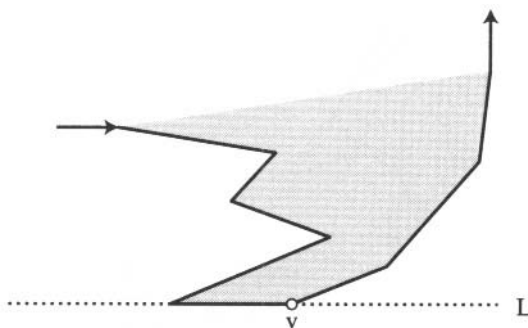


FIGURE 1.11 The rightmost lowest vertex must be strictly convex.

Proof. If the edges of a polygon are oriented so that their direction indicates a counterclockwise traversal, then a strictly convex vertex is a left turn for someone walking around the boundary, and a reflex vertex is a right turn. The interior of the polygon is always to the left of this hypothetical walker. Let L be a line through a lowest vertex v of P , lowest in having minimum y coordinate with respect to a coordinate system; if there are several lowest vertices, let v be the rightmost. The interior of P must be above L . The edge following v must lie above L . See Figure 1.11. Together these conditions imply that the walker makes a left turn at v and therefore that v is a strictly convex vertex. \square

This proof can be used to construct an efficient test for the orientation of a polygon (Exercise 1.3.9[3]).

Lemma 1.2.2 (Meisters). *Every polygon of $n \geq 4$ vertices has a diagonal.*

Proof. Let v be a strictly convex vertex, whose existence is guaranteed by Lemma 1.2.1. Let a and b be the vertices adjacent to v . If ab is a diagonal, we are finished. So suppose ab is not a diagonal. Then either ab is exterior to P , or it intersects ∂P . In either case, since $n > 3$, the closed triangle Δavb contains at least one vertex of P other than a, v, b . Let x be the vertex of P in Δavb that is closest to v , where distance is measured orthogonal to the line through ab . Thus x is the first vertex in Δavb hit by a line L parallel to ab moving from v to ab . See Figure 1.12.

Now we claim that vx is a diagonal of P . For it is clear that the interior of Δavb intersected with the halfplane bounded by L that includes v (the shaded region in the figure) is empty of points of ∂P . Therefore vx cannot intersect ∂P except at v and x , and so it is a diagonal. \square

Theorem 1.2.3 (Triangulation). *Every polygon P of n vertices may be partitioned into triangles by the addition of (zero or more) diagonals.*

Proof. The proof is by induction. If $n = 3$, the polygon is a triangle, and the theorem holds trivially.

Let $n \geq 4$. Let $d = ab$ be a diagonal of P , as guaranteed by Lemma 1.2.2. Because d by definition only intersects ∂P at its endpoints, it partitions P into two polygons,

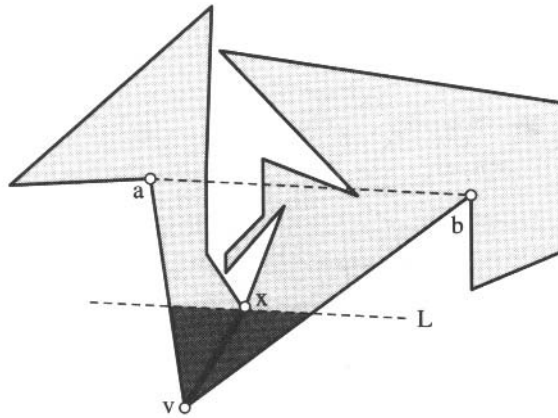


FIGURE 1.12 vx must be a diagonal.

each using d as an edge, and each of fewer than n vertices; see Figure 1.13. The reason each has fewer vertices is that no vertices are added by this process, and clearly there is at least one vertex in each part in addition to a and b . Applying the induction hypothesis to the two subpolygons completes the proof. \square

1.2.2. Properties of Triangulations

Although in general there can be a large number of different ways to triangulate a given polygon (Exercise 1.2.5[4]), they all have the same number of diagonals and triangles, as is easily established by the same argument as used in Theorem 1.2.3:

Lemma 1.2.4 (Number of Diagonals). *Every triangulation of a polygon P of n vertices uses $n - 3$ diagonals and consists of $n - 2$ triangles.*

Proof. The proof is by induction. Both claims are trivially true for $n = 3$.

Let $n \geq 4$. Partition P into two polygons P_1 and P_2 with a diagonal $d = ab$. Let the two polygons have n_1 and n_2 vertices respectively. We have that $n_1 + n_2 = n + 2$, since a and b are counted in both n_1 and n_2 . Applying the induction hypothesis to

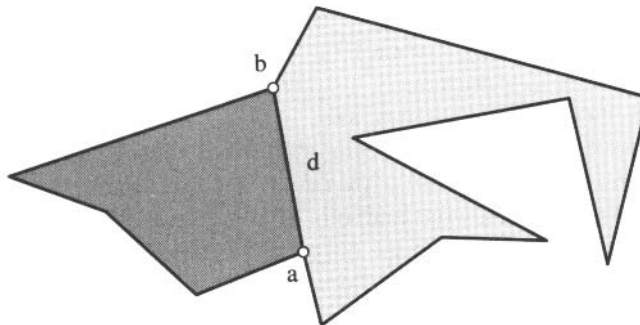


FIGURE 1.13 A diagonal partitions a polygon into two smaller polygons.

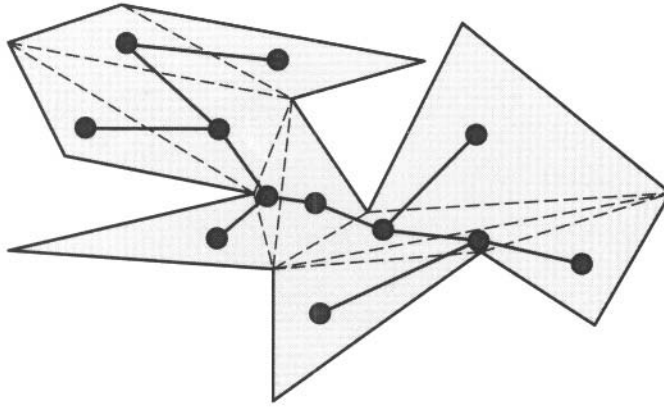


FIGURE 1.14 Triangulation dual.

the subpolygons, we see that altogether there are $(n_1 - 3) + (n_2 - 3) + 1 = n - 3$ diagonals, with the final $+1$ term counting d . And there are $(n_1 - 2) + (n_2 - 2) = n - 2$ triangles. \square

Corollary 1.2.5 (Sum of Angles). *The sum of the internal angles of a polygon of n vertices is $(n - 2)\pi$.*

Proof. There are $n - 2$ triangles by Lemma 1.2.4, and each contributes π to the internal angles. \square

1.2.3. Triangulation Dual

An important concept in graph theory is the “dual” of a graph. We will not need this concept in its full generality, but rather we will define specific dual graphs as the need arises. In particular, studying the triangulation dual reveals useful structure in the triangulation.

The *dual* T of a triangulation of a polygon is a graph with a node associated with each triangle and an arc between two nodes iff their triangles share a diagonal. See Figure 1.14.

Lemma 1.2.6. *The dual T of a triangulation is a tree,¹³ with each node of degree at most three.*

Proof. That each node has degree at most three is immediate from the fact that a triangle has at most three sides to share.

Suppose T is not a tree. Then it must have a cycle C . If this cycle is drawn as a path π in the plane, connecting with straight segments the midpoints of the diagonals shared by the triangles whose nodes comprise C (to make the path specific), then it must enclose some polygon vertices: namely one endpoint of each diagonal crossed by π . But then π must also enclose points exterior to the polygon, for these enclosed vertices are on ∂P . This contradicts the simplicity of the polygon. \square

¹³A *tree* is a connected graph with no cycles.

The nodes of degree one are leaves of T ; nodes of degree two lie on paths of the tree; nodes of degree three are branch points. Note that T is a binary tree when rooted at any node of degree one or two! Given the ubiquity of binary trees in computer science, this correspondence between triangulation duals and binary trees is fortunate and may often be exploited (Exercise 1.2.5[7]).

Lemma 1.2.6 leads to an easy proof of Meisters's "Two Ears Theorem" (Meisters 1975), which, although simple, is quite useful. Three consecutive vertices of a polygon a, b, c form an *ear* of the polygon if ac is a diagonal; b is the ear *tip*. Two ears are *nonoverlapping* if their triangle interiors are disjoint.

Theorem 1.2.7 (Meisters's Two Ears Theorem). *Every polygon of $n \geq 4$ vertices has at least two nonoverlapping ears.*

Proof. A leaf node in a triangulation dual corresponds to an ear. A tree of two or more nodes (by Lemma 1.2.4 the tree has $(n - 2) \geq 2$ nodes) must have at least two leaves. \square

1.2.4. 3-Coloring Proof

This theorem in turn leads to an easy proof of the 3-colorability of triangulation graphs. The idea is to remove an ear for induction, which, because it only "interfaces" at its one diagonal, can be colored consistently.

Theorem 1.2.8 (3-coloring). *The triangulation graph of a polygon P may be 3-colored.*

Proof. The proof is by induction on the number of vertices n . Clearly a triangle can be 3-colored.

Assume therefore that $n \geq 4$. By Theorem 1.2.7, P has an ear $\triangle abc$, with ear tip b . Form a new polygon P' by cutting off the ear: That is, replace the sequence abc in ∂P with ac in $\partial P'$. P' has $n - 1$ vertices: It is missing only b . Apply the induction hypothesis to 3-color P' . Now put the ear back, coloring b with the color not used at a and c . This is a 3-coloring of P . \square

1.2.5. Exercises

1. *Exterior angles* [easy]. What is the sum of the exterior angles of a polygon of n vertices?
2. *Realization of triangulations*. Prove or disprove: Every binary tree is realizable as a triangulation dual of a polygon.
3. *Extreme triangulations*. Which polygons have the fewest number of distinct triangulations? Can polygons have unique triangulations? Which polygons have the largest number of distinct triangulations?
4. *Number of triangulations* [difficult]. How many distinct triangulations are there of a convex polygon of n vertices?
5. *Quad-ears*. An *orthogonal polygon* is one composed entirely of edges that meet orthogonally (e.g., horizontal and vertical edges). Define a notion of a "quad-ear" of an orthogonal polygon, a four-sided version of an ear, and answer the question of whether every orthogonal polygon has a quad-ear under your definition.

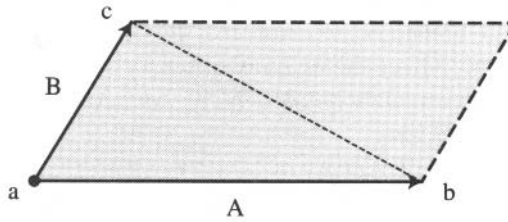


FIGURE 1.15 Cross product parallelogram.

6. *Do nonconvex polygons have mouths?* (Pierre Beauchemin). Define three consecutive vertices a, b, c of a polygon to form a *mouth* if b is reflex and the closed $\triangle abc$ does not contain any vertices other than its three corners. Prove or disprove: Every nonconvex polygon has a mouth.
7. *Tree rotations.* For those who know tree rotations used to balance binary trees:¹⁴ Interpret tree rotations in terms of polygon triangulations.
8. *Diagonals \Rightarrow triangulation.* Given a list of diagonals of a polygon forming a triangulation, with each diagonal specified by counterclockwise indices of the endpoints, design an algorithm to build the triangulation dual tree. [difficult]: Achieve $O(n)$ time at the expense of $O(n^2)$ space.

1.3. AREA OF POLYGON

In this section we will explore the question of how to compute the area of a polygon. Although this is an interesting question in its own right, our objective is to prepare the way for calculation of containment in halfplanes, the intersection between line segments, visibility relations, and ultimately to lead to a triangulation algorithm in Section 1.6.5.

1.3.1. Area of a Triangle

The area of a triangle is one half the base times the altitude. However, this formula is not directly useful if we want the area of a triangle T whose three vertices are arbitrary points a, b, c . Let us denote this area as $\mathcal{A}(T)$. The base is easy: $|a - b|$,¹⁵ but the altitude is not so immediately available from the coordinates, unless the triangle happens to be oriented with one side parallel to one of the axes.

1.3.2. Cross Product

From linear algebra we know that the magnitude of the cross product of two vectors is the area of the parallelogram they determine: If A and B are vectors, then $|A \times B|$ is the area of the parallelogram with sides A and B , as shown in Figure 1.15. Since any triangle can be viewed as half of a parallelogram, this gives an immediate method of computing the area from coordinates. Just let $A = b - a$ and $B = c - a$. Then the area is half the length of $A \times B$. The cross product can be computed from the

¹⁴See, e.g., Cormen, Leiserson & Rivest (1990, pp. 265–7).

¹⁵ $|a - b|$ is the length of the vector $a - b$, sometimes written $\|a - b\|$.

following determinant, where \hat{i} , \hat{j} , and \hat{k} are unit vectors in the x , y , and z directions respectively:

$$\begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ A_0 & A_1 & A_2 \\ B_0 & B_1 & B_2 \end{vmatrix} = (A_1 B_2 - A_2 B_1) \hat{i} + (A_2 B_0 - A_0 B_2) \hat{j} + (A_0 B_1 - A_1 B_0) \hat{k}. \quad (1.1)$$

For two-dimensional vectors, $A_2 = B_2 = 0$, so the above calculation reduces to $(A_0 B_1 - A_1 B_0) \hat{k}$: The cross product is a vector normal (perpendicular) to the plane of the triangle. Thus the area is given by

$$\mathcal{A}(T) = \frac{1}{2}(A_0 B_1 - A_1 B_0).$$

Substitution of $A = b - a$ and $B = c - a$ yields

$$2\mathcal{A}(T) = a_0 b_1 - a_1 b_0 + a_1 c_0 - a_0 c_1 + b_0 c_1 - c_0 b_1 \quad (1.2)$$

$$= (b_0 - a_0)(c_1 - a_1) - (c_0 - a_0)(b_1 - a_1). \quad (1.3)$$

This achieves our immediate goal: an expression for the area of the triangle as a function of the coordinates of its vertices.

1.3.3. Determinant Form

There is another way to represent the calculation of the cross product that is formally identical but generalizes more easily to higher dimensions.¹⁶

The expression obtained above (Equation 1.3), is the value of the 3×3 determinant of the three point coordinates, with the third coordinate replaced by 1:¹⁷

$$\begin{vmatrix} a_0 & a_1 & 1 \\ b_0 & b_1 & 1 \\ c_0 & c_1 & 1 \end{vmatrix} = (b_0 - a_0)(c_1 - a_1) - (c_0 - a_0)(b_1 - a_1) = 2\mathcal{A}(T). \quad (1.4)$$

This determinant is explored in Exercise 1.6.8[1]. We summarize in a lemma.

Lemma 1.3.1. *Twice the area of a triangle $T = (a, b, c)$ is given by*

$$2\mathcal{A}(T) = \begin{vmatrix} a_0 & a_1 & 1 \\ b_0 & b_1 & 1 \\ c_0 & c_1 & 1 \end{vmatrix} = (b_0 - a_0)(c_1 - a_1) - (c_0 - a_0)(b_1 - a_1). \quad (1.5)$$

¹⁶Note that the operation of cross product is restricted to three-dimensional vectors (or two-dimensional vectors with a zero third coordinate). It is more accurate to view the cross product as an exterior product producing, not another vector, but a “bivector.” See, e.g., Koenderink (1990).

¹⁷One can view each row as a point in “homogenous coordinates,” with the third coordinate normalized to 1.

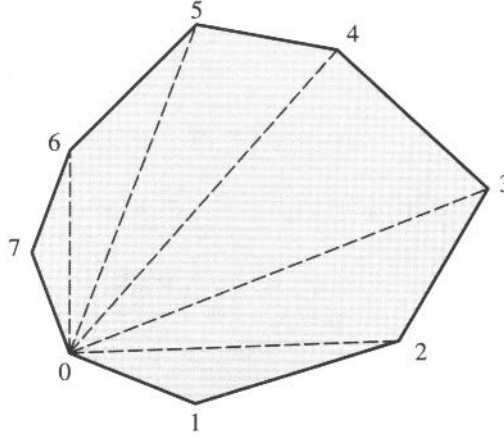


FIGURE 1.16 Triangulation of a convex polygon. The fan center is at 0.

1.3.4. Area of a Convex Polygon

Now that we have an expression for the area of a triangle, it is easy to find the area of any polygon by first triangulating it, and then summing the triangle areas. But it would be pleasing to avoid the rather complex step of triangulation, and indeed this is possible. Before turning to that issue, we consider convex polygons, where triangulation is trivial.

Every convex polygon may be triangulated as a “fan,” with all diagonals incident to a common vertex; and this may be done with any vertex serving as the fan “center.” See Figure 1.16. Therefore the area of a polygon with vertices v_0, v_1, \dots, v_{n-1} labeled counterclockwise can be calculated as

$$A(P) = A(v_0, v_1, v_2) + A(v_0, v_2, v_3) + \cdots + A(v_0, v_{n-2}, v_{n-1}). \quad (1.6)$$

Here v_0 is the fan center.

We will warm up to the result we will prove in Theorem 1.3.3 below by examining convex and nonconvex quadrilaterals, where the relevant relationships are obvious.

1.3.5. Area of a Convex Quadrilateral

The area of a convex quadrilateral $Q = (a, b, c, d)$ may be written in two ways, depending on the two different triangulations (see Figure 1.17):

$$A(Q) = A(a, b, c) + A(a, c, d) = A(d, a, b) + A(d, b, c). \quad (1.7)$$

Writing out the expressions for the areas using Equation (1.2) for the two terms of the first triangulation, we get

$$\begin{aligned} 2A(Q) &= a_0b_1 - a_1b_0 + a_1c_0 - a_0c_1 + b_0c_1 - c_0b_1 \\ &\quad + a_0c_1 - a_1c_0 + a_1d_0 - a_0d_1 + c_0d_1 - d_0c_1. \end{aligned} \quad (1.8)$$

Note that the terms $a_1c_0 - a_0c_1$ appear in $A(a, b, c)$ and in $A(a, c, d)$ with opposite signs, and so they cancel. Thus the terms “corresponding” to the diagonal ac cancel;

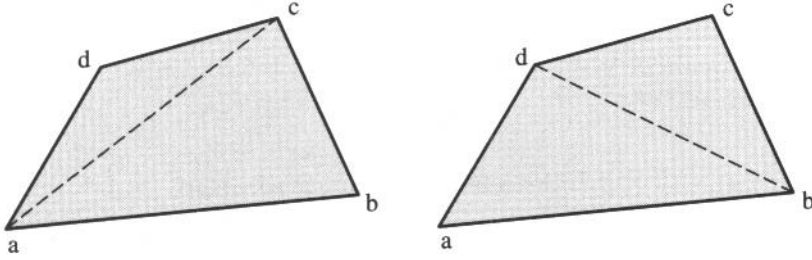


FIGURE 1.17 The two triangulations of a convex quadrilateral.

similarly the terms corresponding to the diagonal db in the second triangulation cancel. And thus we arrive at the exact same expression independent of the triangulation, as of course we must.

Generalizing, we see we get two terms per polygon edge, and none for internal diagonals. So if the coordinates of vertex v_i are x_i and y_i , twice the area of a convex polygon is given by

$$2\mathcal{A}(P) = \sum_{i=0}^{n-1} (x_i y_{i+1} - y_i x_{i+1}). \quad (1.9)$$

We will soon see that this equation holds for nonconvex polygons as well.

1.3.6. Area of a Nonconvex Quadrilateral

Now suppose we have a nonconvex quadrilateral $Q = (a, b, c, d)$ as shown in Figure 1.18. Then there is only one triangulation, using the diagonal db . But we just showed that the algebraic expression obtained is independent of the diagonal chosen, so it must be the case that the equation

$$\mathcal{A}(Q) = \mathcal{A}(a, b, c) + \mathcal{A}(a, c, d)$$

is still true, even though the diagonal ac is external to Q . This equation has an obvious interpretation: $\mathcal{A}(a, c, d)$ is negative, and it is therefore subtracted from the surrounding

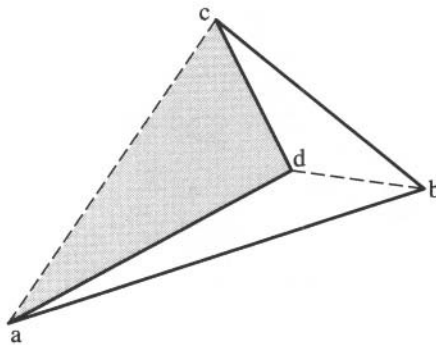


FIGURE 1.18 Triangulation of a nonconvex quadrilateral. The shaded area $\mathcal{A}(a, d, c)$ is negative.

triangle $\triangle abc$. And indeed, note that (a, c, d) is a clockwise path, so the cross product formulation shows that the area will be negative.

The phenomenon observed with a nonconvex quadrilateral is general, as we now proceed to demonstrate.

1.3.7. Area from an Arbitrary Center

We now formalize the observations in the preceding paragraphs, which we will then use to obtain the area of general nonconvex polygons.

Let us generalize the method of summing the areas of the triangles in a triangulation to summing areas based on an arbitrary, perhaps external, point p . Let $T = \triangle abc$ be a triangle, with the vertices oriented counterclockwise, and let p be any point in the plane. Then we claim that

$$\mathcal{A}(T) = \mathcal{A}(p, a, b) + \mathcal{A}(p, b, c) + \mathcal{A}(p, c, a). \quad (1.10)$$

Consider Figure 1.19. With $p = p_1$, the first term of Equation (1.10), $\mathcal{A}(p_1, a, b)$, is negative because the vertices are clockwise, whereas the remaining two terms are positive because the vertices are counterclockwise. Now note that $\mathcal{A}(p_1, a, b)$ subtracts exactly that portion of the quadrilateral (p_1, b, c, a) that lies outside T , leaving the total sum precisely $\mathcal{A}(T)$ as claimed.

Similarly, from $p = p_2$, both $\mathcal{A}(p_2, a, b)$ and $\mathcal{A}(p_2, b, c)$ are negative because the vertices are clockwise, and they remove from $\mathcal{A}(p_2, c, a)$, which is positive, precisely the amount needed to leave $\mathcal{A}(T)$.

All other positions for p in the plane not internal to T are equivalent to either p_1 or p_2 by symmetry; and of course the equation holds when p is internal, as we argued in Section 1.3.4. Therefore we have established the following lemma:

Lemma 1.3.2. *If $T = \triangle abc$ is a triangle, with vertices oriented counterclockwise, and p is any point in the plane, then*

$$\mathcal{A}(T) = \mathcal{A}(p, a, b) + \mathcal{A}(p, b, c) + \mathcal{A}(p, c, a). \quad (1.11)$$

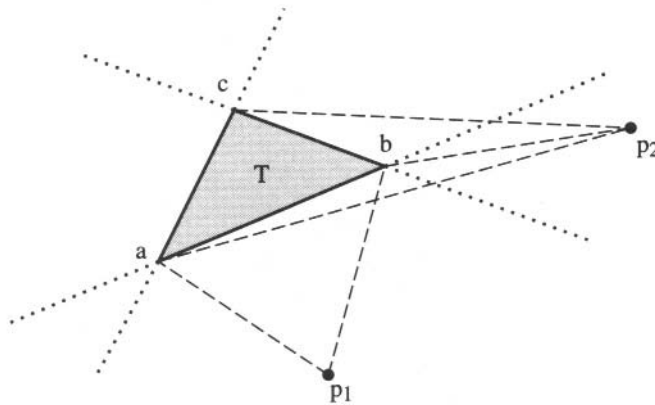


FIGURE 1.19 Area of T based on various external points p_1, p_2 .

We may now generalize the preceding lemma to establish the same equation (generalized) for arbitrary polygons.

Theorem 1.3.3 (Area of Polygon).¹⁸ *Let a polygon (convex or nonconvex) P have vertices v_0, v_1, \dots, v_{n-1} labeled counterclockwise, and let p be any point in the plane. Then*

$$\begin{aligned} \mathcal{A}(P) = & \mathcal{A}(p, v_0, v_1) + \mathcal{A}(p, v_1, v_2) + \mathcal{A}(p, v_2, v_3) + \cdots \\ & + \mathcal{A}(p, v_{n-2}, v_{n-1}) + \mathcal{A}(p, v_{n-1}, v_0). \end{aligned} \quad (1.12)$$

If $v_i = (x_i, y_i)$, this expression is equivalent to the equations

$$2\mathcal{A}(P) = \sum_{i=0}^{n-1} (x_i y_{i+1} - y_i x_{i+1}) \quad (1.13)$$

$$= \sum_{i=0}^{n-1} (x_i + x_{i+1})(y_{i+1} - y_i). \quad (1.14)$$

Proof. We prove the area sum equation by induction on the number of vertices n of P . The base case, $n = 3$, is established by Lemma 1.3.2.

Suppose then that Equation (1.12) is true for all polygons with $n - 1$ vertices, and let P be a polygon of n vertices. By Theorem 1.2.7, P has an “ear.” Renumber the vertices of P so that $E = (v_{n-2}, v_{n-1}, v_0)$ is an ear. Let P_{n-1} be the polygon obtained by removing E . By the induction hypothesis,

$$\mathcal{A}(P_{n-1}) = \mathcal{A}(p, v_0, v_1) + \cdots + \mathcal{A}(p, v_{n-3}, v_{n-2}) + \mathcal{A}(p, v_{n-2}, v_0).$$

By Lemma 1.3.2,

$$\mathcal{A}(E) = \mathcal{A}(p, v_{n-2}, v_{n-1}) + \mathcal{A}(p, v_{n-1}, v_0) + \mathcal{A}(p, v_0, v_{n-2}).$$

Since $\mathcal{A}(P) = \mathcal{A}(P_{n-1}) + \mathcal{A}(E)$, we have

$$\begin{aligned} \mathcal{A}(P) = & \mathcal{A}(p, v_0, v_1) + \cdots + \mathcal{A}(p, v_{n-3}, v_{n-2}) + \mathcal{A}(p, v_{n-2}, v_0) \\ & + \mathcal{A}(p, v_{n-2}, v_{n-1}) + \mathcal{A}(p, v_{n-1}, v_0) + \mathcal{A}(p, v_0, v_{n-2}). \end{aligned}$$

But note that $\mathcal{A}(p, v_0, v_{n-2}) = -\mathcal{A}(p, v_{n-2}, v_0)$. Canceling these terms leads to the claimed equation.

Equation (1.13) is obtained by expansion of the determinants and canceling terms, as explained in Section 1.3.5. Equation (1.14) can be seen as equivalent by multiplying out and again canceling terms. \square

¹⁸This theorem can be viewed as a discrete version of Green’s theorem, which relates an integral around the boundary of a region with an integral over the interior of the region: $\int_{\partial P} \omega = \iint_P d\omega$, where ω is a “1-form” (see, e.g., Buck & Buck (1965, p. 406) or Koenderink (1990, p. 99)).

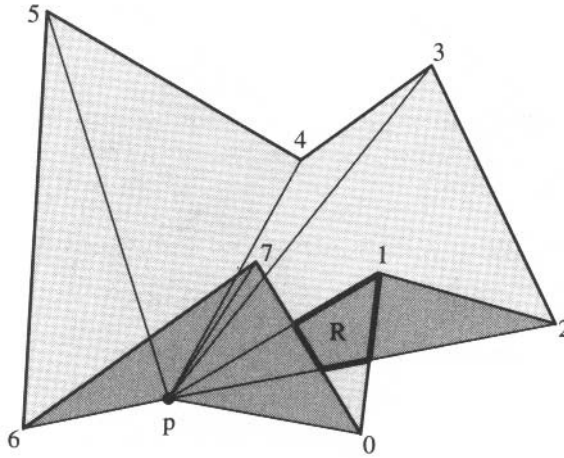


FIGURE 1.20 Computation of the area of a nonconvex polygon from point p . The darker triangles are oriented clockwise, and thus they have negative area.

Equation (1.14) can be computed with one multiplication and two additions per term, whereas Equation (1.13) uses two multiplications and one addition. The second form is therefore more efficient in most implementations.

In Figure 1.20, the triangles $\Delta p12$, $\Delta p67$, and $\Delta p70$ are oriented clockwise, and the remainder are counterclockwise. One can think of the counterclockwise triangles as attaching to each point they cover a $+1$ charge, whereas the clockwise triangles attach a -1 charge. Then the points R of $\Delta p12$ that falls inside the polygon (labeled in the figure) are given a -1 charge by this clockwise triangle; but R is also covered by two counterclockwise triangles, $\Delta p01$ and $\Delta p23$. So R has net $+1$ charge. Similarly every point inside P is assigned a net $+1$ charge, and every point outside is assigned a net 0 charge.

1.3.8. Volume in Three and Higher Dimensions

One of the benefits of the determinant formulation of the area of a triangle in Lemma 1.3.1 is that it extends directly into higher dimensions. In three dimensions, the volume of a tetrahedron T with vertices a, b, c, d is

$$6V(T) = \begin{vmatrix} a_0 & a_1 & a_2 & 1 \\ b_0 & b_1 & b_2 & 1 \\ c_0 & c_1 & c_2 & 1 \\ d_0 & d_1 & d_2 & 1 \end{vmatrix} \quad (1.15)$$

$$\begin{aligned} &= -(a_2 - d_2)(b_1 - d_1)(c_0 - d_0) + (a_1 - d_1)(b_2 - d_2)(c_0 - d_0) \\ &\quad + (a_2 - d_2)(b_0 - d_0)(c_1 - d_1) - (a_0 - d_0)(b_2 - d_2)(c_1 - d_1) \\ &\quad - (a_1 - d_1)(b_0 - d_0)(c_2 - d_2) + (a_0 - d_0)(b_1 - d_1)(c_2 - d_2). \end{aligned} \quad (1.16)$$

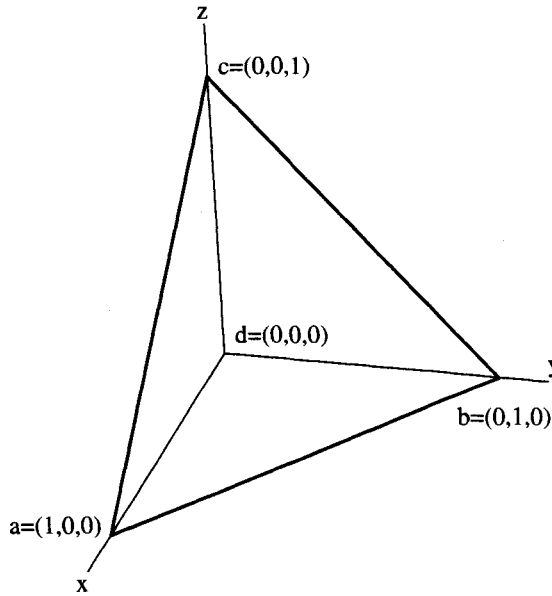


FIGURE 1.21 Tetrahedron at the origin.

This volume is signed; it is positive if (a, b, c) form a counterclockwise circuit when viewed from the side away from d , so that the face normal determined by the right-hand rule points toward the outside. For example, let $a = (1, 0, 0)$, $b = (0, 1, 0)$, $c = (0, 0, 1)$, and $d = (0, 0, 0)$. Then (a, b, c) is counterclockwise from outside; see Figure 1.21. Substitution into Equation (1.15) yields a determinant of 1, so $\mathcal{V}(T) = \frac{1}{6}$. This accords with the $\frac{1}{3}$ base area times height rule: $\frac{1}{3} \cdot \frac{1}{2} \cdot 1$. We will make use of this volume formula later to compute the “convex hull” of points in three dimensions (Chapter 4).

Remarkably, Theorem 1.3.3 generalizes directly also: The volume of a polyhedron may be computed by summing the (signed) volumes of tetrahedra formed by an arbitrary point and each triangular face of the polyhedron (Exercise 4.7[7].) Here all the faces must be oriented counterclockwise from outside.

Moreover, Equation (1.15) generalizes to higher dimensions d , yielding the volume of the d -dimensional “simplex” (the generalization of a tetrahedron to higher dimensions) times the constant $d!$.

1.3.9. Exercises

1. *Triple product.* Interpret the determinant expression (Equation (1.4)) for the area of a triangle in terms of the triple vector product.

If A , B , and C are three-dimensional vectors, then their triple product is $A \cdot (B \times C)$. This is a scalar with value equal to the volume of the parallelepiped determined by the three vectors, determined in the same sense that two vectors determine a parallelogram. The value is the same as that of the determinant

$$A \cdot (B \times C) = \begin{vmatrix} A_0 & A_1 & A_2 \\ B_0 & B_1 & B_2 \\ C_0 & C_1 & C_2 \end{vmatrix}.$$

Assuming this determinant is the parallelepiped volume, argue that Equation (1.4) is twice the area of the indicated triangle.

2. *Orientation of a polygon: from area [easy].* Given a list of vertices of a simple polygon in boundary traversal order, how can its orientation (clockwise versus counterclockwise) be determined using Theorem 1.3.3?
3. *Orientation of a polygon.* Use the proof of Lemma 1.2.1 to design a more efficient algorithm for determining the orientation of a polygon.
4. *Volume of a cube.* Compute the volume of a unit cube (side length 1) with the analog of Equation (1.12), using one vertex as p .

1.4. IMPLEMENTATION ISSUES

The remainder of the chapter takes a rather long “digression” into implementation issues. The goal is to present code to compute a triangulation. This hinges on detecting intersection between two segments, a seemingly trivial task that often is implemented incorrectly. We will approach segment intersection using the computation of areas from Section 1.3. We start with a few representation issues.

1.4.1. Representation of a Point

Arrays versus Records

All points will be represented by arrays of the appropriate number of coordinates. It is common practice to represent a point by a record with fields named x and y , but this precludes the use of for-loops to iterate over the coordinates.¹⁹ There may seem little need to write a for-loop to iterate over only two indices, but I find it easier to understand, and it certainly generalizes to higher dimensions more easily.

Integers versus Reals

We will represent the coordinates with integers rather than with floating-point numbers wherever possible. This will permit us to avoid the issue of floating-point round-off error and allow us to write code that is verifiably correct within a range of coordinate values. Numerical error is an important topic and will be discussed at various points throughout the book (e.g., Sections 4.3.5 and 7.2). Obviously this habit of using integers will have to be relaxed when we compute, for example, the point of intersection between two line segments. The type definitions will be isolated so that modification of the code to handle different varieties of coordinate datatypes can be made in one location.

Point Type Definition

All type identifiers will begin with lowercase t . All defined constants will appear entirely in uppercase. The suffixes i and d indicate *integer* and *double* types respectively. See Code 1.1. In mathematical expressions, we will write p_0 and p_1 for $p[0]$ and $p[1]$.

¹⁹That is, precludes it in most programming languages.

```

#define X 0
#define Y 1
typedef enum {FALSE, TRUE } bool;

#define DIM 2                /* Dimension of points */
typedef int tPointi[DIM];    /* Type integer point */

```

Code 1.1 Point type.**1.4.2. Representation of a Polygon**

The main options here are whether to use an array or a list, and if the latter, whether singly or doubly linked, and whether linear or circular.

Arrays are attractive for code clarity: The structure of loops and index increments are somewhat clearer with arrays than with lists. However, insertion and deletion of points is clumsy with arrays. As the triangulation code we develop will clip off ears, we will sacrifice simplicity to gain ease of deletion. In any case, we will need to use identical structures for the convex hull code in Chapters 3 and 4, so the investment here will reward us later. With an eye toward that generality, we opt to use a doubly linked circular list to represent a polygon. The basic cell of the data structure represents a single vertex, `tVertexStructure`, whose primary data field is `tPoint`. Pointers `next` and `prev` are provided to link each vertex to its adjacent vertices. See Code 1.2. An integer index `vnum` is included for printout, and other fields (such as `bool ear`) will be added as necessary.

```

typedef struct tVertexStructure tsVertex;    /* Used only in NEW(). */
typedef tsVertex *tVertex;
struct tVertexStructure {
    int      vnum;                /* Index */
    tPointi  v;                  /* Coordinates */
    bool     ear;                 /* TRUE iff an ear */
    tVertex  next, prev;
};
tVertex vertices = NULL;         /* "Head" of circular list. */

```

Code 1.2 Vertex structure.

At all times, a global variable `vertices` is maintained that points to some vertex cell. This will serve as the "head" of the list during iterative processing. Loops over all vertices will take the form shown in Code 1.3. Care must be exercised if the processing in the loop deletes the cell to which vertex points.