

# Computational Geometry – Algorithms and Applications



Mark de Berg Marc van Kreveld  
Mark Overmars Otfried Schwarzkopf

# **Computational Geometry**

## Algorithms and Applications

Second, Revised Edition

With 370 Illustrations



Springer

Dr. Mark de Berg  
Department of Computer Science,  
TU Eindhoven,  
P.O.Box 513, 5600 MB Eindhoven,  
the Netherlands.  
email: m.t.d.berg@TUE.nl

Dr. Marc van Kreveld  
Prof. Dr. Mark Overmars  
Dr. Otfried Cheong, né Schwarzkopf  
Department of Computer Science  
Utrecht University  
P.O. Box 80.089  
3508 TB Utrecht, The Netherlands  
{marc,markov,otfried}@cs.uu.nl

Library of Congress Cataloging-in-Publication Data applied for  
Die Deutsche Bibliothek – CIP-Einheitsaufnahme

**Computational geometry: algorithms and applications / Mark de Berg...** – 2., rev. ed. –  
Berlin; Heidelberg; New York; Barcelona; Hong Kong; London; Milan; Paris; Singapore; Tokyo:  
Springer, 2000

ACM Computing Classification (1998): F.2.2, I.3.5

ISBN 978-3-662-04247-2      ISBN 978-3-662-04245-8 (eBook)  
DOI 10.1007/978-3-662-04245-8

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

© Springer-Verlag Berlin Heidelberg 1997, 2000  
Originally published by Springer-Verlag Berlin Heidelberg New York in 2000.  
Softcover reprint of the hardcover 2nd edition 2000

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Cover Design: Künkel + Lopka, Heidelberg  
Typesetting: Camera-ready by authors

Printed on acid-free paper      SPIN 11010180      45/3111 – 5 4 3

# Preface

---

Computational geometry emerged from the field of algorithms design and analysis in the late 1970s. It has grown into a recognized discipline with its own journals, conferences, and a large community of active researchers. The success of the field as a research discipline can on the one hand be explained from the beauty of the problems studied and the solutions obtained, and, on the other hand, by the many application domains—computer graphics, geographic information systems (GIS), robotics, and others—in which geometric algorithms play a fundamental role.

For many geometric problems the early algorithmic solutions were either slow or difficult to understand and implement. In recent years a number of new algorithmic techniques have been developed that improved and simplified many of the previous approaches. In this textbook we have tried to make these modern algorithmic solutions accessible to a large audience. The book has been written as a textbook for a course in computational geometry, but it can also be used for self-study.

**Structure of the book.** Each of the sixteen chapters (except the introductory chapter) starts with a problem arising in one of the application domains. This problem is then transformed into a purely geometric one, which is solved using techniques from computational geometry. The geometric problem and the concepts and techniques needed to solve it are the real topic of each chapter. The choice of the applications was guided by the topics in computational geometry we wanted to cover; they are not meant to provide a good coverage of the application domains. The purpose of the applications is to motivate the reader; the goal of the chapters is not to provide ready-to-use solutions for them. Having said this, we believe that knowledge of computational geometry is important to solve geometric problems in application areas efficiently. We hope that our book will not only raise the interest of people from the algorithms community, but also from people in the application areas.

For most geometric problems treated we give just one solution, even when a number of different solutions exist. In general we have chosen the solution that is easiest to understand and implement. This is not necessarily the most efficient solution. We also took care that the book contains a good mixture of techniques like divide-and-conquer, plane sweep, and randomized algorithms. We decided not to treat all sorts of variations to the problems; we felt it is more important to introduce all main topics in computational geometry than to give more detailed information about a smaller number of topics.

Several chapters contain one or more sections marked with a star. They contain improvements of the solution, extensions, or explain the relation between various problems. They are not essential for understanding the remainder of the book.

Every chapter concludes with a section that is entitled *Notes and Comments*. These sections indicate where the results described in the chapter originated, mention other solutions, generalizations, and improvements, and provide references. They can be skipped, but do contain useful material for those who want to know more about the topic of the chapter.

At the end of each chapter a number of exercises is provided. These range from easy tests to check whether the reader understands the material to more elaborate questions that extend the material covered. Difficult exercises and exercises about starred sections are indicated with a star.

**A course outline.** Even though the chapters in this book are largely independent, they should preferably not be treated in an arbitrary order. For instance, Chapter 2 introduces plane sweep algorithms, and it is best to read this chapter before any of the other chapters that use this technique. Similarly, Chapter 4 should be read before any other chapter that uses randomized algorithms.

For a first course on computational geometry, we advise treating Chapters 1–10 in the given order. They cover the concepts and techniques that, according to us, should be present in any course on computational geometry. When more material can be covered, a selection can be made from the remaining chapters.

**Prerequisites.** The book can be used as a textbook for a high-level undergraduate course or a low-level graduate course, depending on the rest of the curriculum. Readers are assumed to have a basic knowledge of the design and analysis of algorithms and data structures: they should be familiar with big-Oh notations and simple algorithmic techniques like sorting, binary search, and balanced search trees. No knowledge of the application domains is required, and hardly any knowledge of geometry. The analysis of the randomized algorithms uses some very elementary probability theory.

**Implementations.** The algorithms in this book are presented in a pseudo-code that, although rather high-level, is detailed enough to make it relatively easy to implement them. In particular we have tried to indicate how to handle degenerate cases, which are often a source of frustration when it comes to implementing.

We believe that it is very useful to implement one or more of the algorithms; it will give a feeling for the complexity of the algorithms in practice. Each chapter can be seen as a programming project. Depending on the amount of time available one can either just implement the plain geometric algorithms, or implement the application as well.

To implement a geometric algorithm a number of basic data types—points, lines, polygons, and so on—and basic routines that operate on them are needed.

Implementing these basic routines in a robust manner is not easy, and takes a lot of time. Although it is good to do this at least once, it is useful to have a software library available that contains the basic data types and routines. Pointers to such libraries can be found on our World Wide Web site.

**World Wide Web.** This book is accompanied by a World Wide Web site, which provides lots of additional material, like an addendum, pointers to geometric software and to an online literature database containing close to 10,000 papers written on computational geometry, and links to other sites that contain information about computational geometry. The address is

`http://www.cs.uu.nl/geobook/`

You can also use our WWW page to send us errors you found and any other comments you have about the book.

**About the second edition.** This second edition is largely the same as the first edition; most changes are corrections of small errors. In principle it is possible for students in a course to still use the first edition. In that case, however, you should be aware of the following changes. First of all, we went carefully over all the exercises, reformulating or removing some old ones and adding a number of new ones. Secondly, larger revisions have occurred in Chapter 4 (where the treatment of unbounded linear programs is different) and in Chapter 7 (where several details in the algorithm have changed).

**Acknowledgements.** Writing a textbook is a long process, even with four authors. Over the past years many people helped us by providing useful advice on what to put in the book and what not, by reading chapters and suggesting changes, and by finding and correcting errors. In particular we would like to thank Pankaj Agarwal, Helmut Alt, Marshall Bern, Jit Bose, Hazel Everett, Gerald Farin, Steve Fortune, Geert-Jan Giezeman, Mordecai Golin, Dan Halperin, Richard Karp, Matthew Katz, Klara Kedem, Nelson Max, René van Oostrum, Henry Shapiro, Sven Skyum, Jack Snoeyink, Gert Vegter, Peter Widmayer, Chee Yap, and Günther Ziegler. Preliminary versions of the book were used in courses in our and other departments. We thank all students who suffered from incomplete versions and errors, and who helped us polish the book into its current shape. We also would like to thank Springer-Verlag for their advice and support during the final stages of the creation of this book.

Finally we would like to acknowledge the support of the Netherlands Organization for Scientific Research (N.W.O.), which supported the project *Computational Geometry and its Application* during which most of this book was written.

Eindhoven,  
Utrecht (the Netherlands), September 1999

Mark de Berg  
Marc van Kreveld  
Mark Overmars  
Otfried Cheong (né Schwarzkopf)

# Contents

---

<b>1</b>	<b>Computational Geometry</b>	1
	<b>Introduction</b>	
1.1	An Example: Convex Hulls	2
1.2	Degeneracies and Robustness	8
1.3	Application Domains	10
1.4	Notes and Comments	13
1.5	Exercises	15
<b>2</b>	<b>Line Segment Intersection</b>	19
	<b>Thematic Map Overlay</b>	
2.1	Line Segment Intersection	20
2.2	The Doubly-Connected Edge List	29
2.3	Computing the Overlay of Two Subdivisions	33
2.4	Boolean Operations	39
2.5	Notes and Comments	40
2.6	Exercises	42
<b>3</b>	<b>Polygon Triangulation</b>	45
	<b>Guarding an Art Gallery</b>	
3.1	Guarding and Triangulations	46
3.2	Partitioning a Polygon into Monotone Pieces	49
3.3	Triangulating a Monotone Polygon	55
3.4	Notes and Comments	59
3.5	Exercises	60
<b>4</b>	<b>Linear Programming</b>	63
	<b>Manufacturing with Molds</b>	
4.1	The Geometry of Casting	64
4.2	Half-Plane Intersection	66
4.3	Incremental Linear Programming	71
4.4	Randomized Linear Programming	77

4.5	Unbounded Linear Programs	80
4.6*	Linear Programming in Higher Dimensions	82
4.7*	Smallest Enclosing Discs	86
4.8	Notes and Comments	90
4.9	Exercises	91
<b>5</b>	<b>Orthogonal Range Searching Querying a Database</b>	<b>95</b>
5.1	1-Dimensional Range Searching	96
5.2	Kd-Trees	99
5.3	Range Trees	105
5.4	Higher-Dimensional Range Trees	109
5.5	General Sets of Points	111
5.6*	Fractional Cascading	112
5.7	Notes and Comments	115
5.8	Exercises	117
<b>6</b>	<b>Point Location Knowing Where You Are</b>	<b>121</b>
6.1	Point Location and Trapezoidal Maps	122
6.2	A Randomized Incremental Algorithm	128
6.3	Dealing with Degenerate Cases	137
6.4*	A Tail Estimate	140
6.5	Notes and Comments	143
6.6	Exercises	144
<b>7</b>	<b>Voronoi Diagrams The Post Office Problem</b>	<b>147</b>
7.1	Definition and Basic Properties	148
7.2	Computing the Voronoi Diagram	151
7.3	Notes and Comments	160
7.4	Exercises	162
<b>8</b>	<b>Arrangements and Duality Supersampling in Ray Tracing</b>	<b>165</b>
8.1	Computing the Discrepancy	167
8.2	Duality	169
8.3	Arrangements of Lines	172
8.4	Levels and Discrepancy	177
8.5	Notes and Comments	178
8.6	Exercises	180

---

<b>9</b>	<b>Delaunay Triangulations</b>	183	CONTENTS
	<b>Height Interpolation</b>		
9.1	Triangulations of Planar Point Sets	185	
9.2	The Delaunay Triangulation	188	
9.3	Computing the Delaunay Triangulation	191	
9.4	The Analysis	197	
9.5*	A Framework for Randomized Algorithms	200	
9.6	Notes and Comments	206	
9.7	Exercises	207	
<b>10</b>	<b>More Geometric Data Structures</b>	211	
	<b>Windowing</b>		
10.1	Interval Trees	212	
10.2	Priority Search Trees	218	
10.3	Segment Trees	223	
10.4	Notes and Comments	229	
10.5	Exercises	230	
<b>11</b>	<b>Convex Hulls</b>	235	
	<b>Mixing Things</b>		
11.1	The Complexity of Convex Hulls in 3-Space	236	
11.2	Computing Convex Hulls in 3-Space	238	
11.3*	The Analysis	242	
11.4*	Convex Hulls and Half-Space Intersection	245	
11.5*	Voronoi Diagrams Revisited	247	
11.6	Notes and Comments	248	
11.7	Exercises	249	
<b>12</b>	<b>Binary Space Partitions</b>	251	
	<b>The Painter's Algorithm</b>		
12.1	The Definition of BSP Trees	253	
12.2	BSP Trees and the Painter's Algorithm	255	
12.3	Constructing a BSP Tree	256	
12.4*	The Size of BSP Trees in 3-Space	260	
12.5	Notes and Comments	263	
12.6	Exercises	264	
<b>13</b>	<b>Robot Motion Planning</b>	267	
	<b>Getting Where You Want to Be</b>		
13.1	Work Space and Configuration Space	268	

13.2	A Point Robot	270
13.3	Minkowski Sums	275
13.4	Translational Motion Planning	281
13.5*	Motion Planning with Rotations	283
13.6	Notes and Comments	287
13.7	Exercises	289
<b>14</b>	<b>Quadtrees</b>	<b>291</b>
	<b>Non-Uniform Mesh Generation</b>	
14.1	Uniform and Non-Uniform Meshes	292
14.2	Quadtrees for Point Sets	293
14.3	From Quadtrees to Meshes	300
14.4	Notes and Comments	302
14.5	Exercises	304
<b>15</b>	<b>Visibility Graphs</b>	<b>307</b>
	<b>Finding the Shortest Route</b>	
15.1	Shortest Paths for a Point Robot	308
15.2	Computing the Visibility Graph	310
15.3	Shortest Paths for a Translating Polygonal Robot	314
15.4	Notes and Comments	315
15.5	Exercises	316
<b>16</b>	<b>Simplex Range Searching</b>	<b>319</b>
	<b>Windowing Revisited</b>	
16.1	Partition Trees	320
16.2	Multi-Level Partition Trees	327
16.3	Cutting Trees	330
16.4	Notes and Comments	336
16.5	Exercises	337
	<b>Bibliography</b>	<b>341</b>
	<b>Index</b>	<b>359</b>