Preface

This book grew out of a conversation between two of the authors. We were discussing the fact that many of our students needed a set of competencies, which they could not really learn in any course that we offered at the Technical University of Denmark. The specific competencies were at the junction of computer vision and computer graphics, and they all had something to do with "how to deal with" discrete 3D shapes (often simply denoted "geometry").

The tiresome fact was that many of our students at graduate level had to pick up things like registration of surfaces, smoothing of surfaces, reconstruction from point clouds, implicit surface polygonization, etc. on their own. Somehow these topics did not quite fit in a graphics course or a computer vision course. In fact, just a few years before our conversation, topics such as these had begun to crystallize out of computer graphics and vision forming the field of geometry processing. Consequently, we created a course in computational geometry processing and started writing a set of course notes, which have been improved over the course of a few years, and now, after some additional polishing and editing, form the present book.

Of course, the question remains: why was the course an important missing piece in our curriculum, and, by extension, why should anyone bother about this book?

The answer is that optical scanning is becoming ubiquitous. In principle, any technically minded person can create a laser scanner using just a laser pointer, a web cam, and a computer together with a few other paraphernalia. Such a device would not be at the 20 micron precision which an industrial laser scanner touts these days, but it goes to show that the principles are fairly simple. The result is that a number of organizations now have easy access to optical acquisition devices. In fact, many individuals have too—since the Microsoft Kinect contains a depth sensing camera. Geometry also comes from other sources. For instance, medical CT, MR and 3D ultrasound scanners provide us with huge volumetric images from which we can extract surfaces.

However, often we cannot directly use this acquired geometry for its intended purpose. Any measurement is fraught with error, so we need to be able to filter the geometry to reduce noise, and usually acquired geometry is also very verbose and simplification is called for. Often we need to convert between various representations, or we need to put together several partial models into one big model. In other words, raw acquired geometry needs to be processed before it is useful for some envisioned purpose, and this book is precisely about algorithms for such processing of geometry as is needed in order to make geometric data useful.

Overview and Goals

Geometry processing can loosely be defined as the field which is concerned with how geometric objects (points, lines, polygons, polyhedra, smooth curves, or smooth surfaces) are worked upon by a computer. Thus, we are mostly concerned with algorithms that work on a (large) set of data. Often, but not necessarily, we have data that have been acquired by scanning some real object. Dealing with laser scanned data is a good example of what this book is about, but it is by no means the only example.

We could have approached the topic by surveying the literature within the topics covered by the book. That would have led to a book giving an overview of the topics, and it would have allowed us to cover more methods than we actually do. Instead, since we believe that we have a relatively broad practical experience in the areas, we have chosen to focus on methods we actually use, cf. Chap. 1. Therefore, with very few exceptions, the methods covered in this book have been implemented by one or more of the authors. This strategy has allowed us to put emphasis on what we believe to be the core tools of the subject, allowing the reader to gain a deeper understanding of these, and, hopefully, made the text more accessible. We believe that our strategy makes this book very suitable for teaching, because students are able to implement much of the material in this book without needing to consult other texts.

We had a few other concerns too. One is that we had no desire to write a book which was tied to a specific programming library or even a specific programming language, since that tends to make some of the information in a book less general. On the other hand, in our geometry processing course, we use C++ for the exercises in conjunction with a library called GEL[1] which contains many algorithms and functions for geometry processing. In this book, we rarely mention GEL except in the exercises, where we sometimes make a note that some particular problem can be solved in a particular way using the GEL library.

In many ways this is a practical book, but we aim to show the connections to the mathematical underpinnings: Most of the methods rely on theory which it is our desire to explain in as much detail as it takes for a graduate student to not only implement a given method but also to understand the ideas behind it, its limitations and its advantages.

Organization and Features

A problem confronting any author is how to delimit the subject. In this book, we cover a range of topics that almost anyone intending to do work in geometry processing will need to be familiar with. However, we choose not to go into concrete

[1] C++ library developed by some of the authors of this book and freely available. URL provided at the end of this preface.

applications of geometry processing. For instance, we do not discuss animation, deformation, 3D printing of prototypes, or topics pertaining to (computer graphics) rendering of geometric data. In the following, we give a brief overview of the contents of the book.

Chapter 1 contains a brief overview of techniques for acquisition of 3D geometry and applications of 3D geometry.

Chapters 2–4 are about mathematical theory which is used throughout the rest of the book. Specifically, these chapters cover vector spaces, metric space, affine spaces, differential geometry, and finite difference methods for computing derivatives and solving differential equations. For many readers these chapters will not be necessary on a first reading, but they may serve as useful points of reference when something in a later chapter is hard to understand.

Chapters 5–7 are about geometry representations. Specifically, these chapters cover polygonal meshes, splines, and subdivision surfaces.

Chapter 8 is about computing curvature from polygonal meshes. This is something often needed either for analysis or for the processing algorithms described in later chapters.

Chapters 9–11 describe algorithms for mesh smoothing, mesh parametrization, and mesh optimization and simplification—operations very often needed in order to be able to use acquired geometry for the intended purpose.

Chapters 12–13 cover point location databases and convex hulls of point sets. Point databases (in particular kD trees) are essential to many geometry processing algorithms, for instance registration. Convex hulls are also needed in numerous contexts such as collision detection.

Chapters 14–18 are about a variety of topics that pertain to the reconstruction of triangle meshes from point clouds: Delaunay triangulation, registration of point clouds (or meshes), surface reconstruction using scattered data interpolation (with radial basis functions), volumetric methods for surface reconstruction and the level set method, and finally isosurface extraction. Together, these chapters should provide a fairly complete overview of the algorithms needed to go from a raw set of scanned points to a final mesh. For further processing of the mesh, the algorithms in Chaps. 9–11 are likely to be useful.

Target Audience

The intended reader of this book is a professional or a graduate student who is familiar with (and able to apply) the main results of linear algebra, calculus, and differential equations. It is an advantage to be familiar with a number of more advanced subjects, especially differential geometry, vector spaces, and finite difference methods for partial differential equations. However, since many graduate students tend to need a brush up on these topics, the initial chapters cover the mathematical preliminaries just mentioned.

The ability to program in a standard imperative programming language such as C++, C, C#, Java or similar will be a distinct advantage if the reader intends to put the material in this book to actual use. Provided the reader is familiar with such a

programming language, he or she should be able to implement many of the methods presented in this book. The implementation will, however, be much easier if a library of basic data structures and algorithms for dealing with linear algebra and geometric data is available.

Supplemental Resources

At the web page of this book, http://www.springer.com/978-1-4471-4074-0, we provide three types of supplementary material.

1. Data for exercises. This comprises point sets and polygonal meshes suitable for solving some of the exercise problems which are listed at the end of each chapter.
2. The GEL library. GEL is an abbreviation for *Geometry and Linear algebra Library*'—a collection of C++ classes and functions distributed as source code. GEL is useful for geometry processing and visualization tasks in particular and most of the algorithms in this book have been implemented on top of GEL.
3. Example C++ programs. Readers interested in implementing the material in this book using GEL will probably find it very convenient to use our example programs. These programs build on GEL and should make it easy and convenient to get started. The example programs are fairly generic, but for all programming one of the examples should serve as a convenient starting point.

Notes to the Instructor

As mentioned above, the first three chapters in this book are considered to be prerequisite material, and would typically not be part of a course syllabus. For instance, we expect students who follow our geometry processing course to have passed a course in differential geometry, but experience has taught us that not all come with the prerequisites. Therefore, we have provided the four initial chapters to give the students a chance to catch up on some of the basics.

In general, it might be a good idea to consider the grouping of chapters given in the overview above as the "atomic units". We do have references from one chapter to another, but the chapters can be read independently. The exception is that Chap. 5 introduces many notions pertaining to polygonal meshes without which it is hard to understand many of the later chapters, so we recommend that this chapter is not skipped in a course based on this book.

GEL is just one library amongst many others, but it is the one we used in the exercises from the aforementioned course. Since we strongly desire that the book should not be too closely tied to GEL and that it should be possible to use this book with other packages, no reference is made to GEL in the main description of each exercise, but in some of the exercises you will find paragraphs headed by

[GEL Users]

These paragraphs contain notes on material that can be used by GEL users.

Acknowledgements

A number of 3D models and images have been provided by courtesy of people or organizations outside the circle of authors.

- The Stanford bunny, provided courtesy of The Stanford Computer Graphics Laboratory, has been used in Chaps. 9, 11, 16, and 17. In most places Greg Turk's reconstruction (Turk and Levoy, Computer Graphics Proceedings, pp. 311–318, 1994) has been used, but in Chap. 17, the geometry of the bunny is reconstructed from the original range scans.
- The 3D scans of the Egea bust and the Venus statue both used in Chap. 11 are provided by the AIM@SHAPE Shape Repository.
- Stine Bærentzen provided the terrain data model used in Chaps. 9 and 11.
- Rasmus Reinhold Paulsen provided the 3D model of his own head (generated from a structured light scan), which was used in Chap. 11.
- In Fig. 10.3 we have used two pictures taken from Wikipedia. The Mercator projection by Peter Mercator, http://en.wikipedia.org/wiki/File:MercNormSph.png and Lambert azimuthal equal-area projection by Strebe, http://en.wikipedia.org/wiki/File:Lambert_azimuthal_equal-area_projection_SW.jpg.
- In Fig. 12.4, we have used the 3D tree picture taken from Wikipedia, http://en.wikipedia.org/wiki/File:3dtree.png.
- In Fig. 12.10, we have used the octree picture taken from Wikipedia, http://fr.wikipedia.org/wiki/Fichier:Octreend.png.
- In Fig. 12.11, we have used the r-tree picture taken from Wikipedia, http://en.wikipedia.org/wiki/File:R-tree.svg.
- In Fig. 12.12, we have used the 3D r-tree picture taken from Wikipedia, http://en.wikipedia.org/wiki/File:RTree-Visualization-3D.svg.

We would like to acknowledge our students, who, through their feedback, have helped us improve the course and the material which grew into this book. We would also like to thank the company 3Shape. Every year, we have taken our class to 3Shape for a brief visit to show them applications of the things they learn. That has been very helpful in motivating the course and, thereby, also the material in this book.

Research and university teaching is becoming more and more of a team sport, and as such we would also like to thank our colleagues at the Technical University of Denmark for their help and support in the many projects where we gained experience that has been distilled into this book.

Last but not least, we would like to thank our families for their help and support.

Kongens Lyngby, Denmark

Jakob Andreas Bærentzen
Jens Gravesen
François Anton
Henrik Aanæs

http://www.springer.com/978-1-4471-4074-0

Guide to Computational Geometry Processing
Foundations, Algorithms, and Methods
Bærentzen, J.A.; Gravesen, J.; Anton, F.; Aanæs, H.
2012, XVIII, 326 p., Hardcover
ISBN: 978-1-4471-4074-0