Foundations of 3D Computer Graphics

"49265_1P_9270_000.tex" — 1/5/2012 — 14:38 — page i

FOUNDATIONS OF 3D COMPUTER GRAPHICS

Steven J. Gortler

The MIT Press
Cambridge, Massachusetts
London, England

MIT Press books may be purchased at special quantity discounts for business or sales promotional use. For information, please email special_sales@mitpress.mit.edu or write to Special Sales Department, The MIT Press, 55 Hayward Street, Cambridge, MA 02142.

This book was set in Syntax and Times Roman by Westchester Book Group. Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data

10 9 8 7 6 5 4 3 2 1

To A.L.G., F.J.G, and O.S.G.

Contents

−1
0
1

Preface

This book developed out of an introductory computer graphics course I have been teaching at Harvard since 1996. Over the years, I have had the pleasure of teaching many amazing students. During class, these students have asked many good questions. In light of these questions, I often realized that some of my explanations in class were a bit sloppy and that I didn't fully understand the topic I had just tried to explain. This would often lead me to rethink the material and change the way I taught it the next time around. Many of these ideas have found their way into this book. Throughout the course of the book, I cover mostly standard material but with an emphasis on understanding some of the more subtle concepts involved.

In this book, we will introduce the basic algorithmic technology needed to produce three-dimensional (3D) computer graphics. We will cover the basic ideas of how 3D shapes are represented and moved around algorithmically. We will cover how a camera can be algorithmically modeled turning this 3D data into a two-dimensional (2D) image made up of a discrete set of dots, or *pixels*, on a screen. Later in the book, we will cover some advanced topics on the basics of color and light representations. We will also briefly introduce some advanced topics on light simulation for producing photo-realistic images, on various ways of dealing with geometric representations, and on producing animated computer graphics.

In this book, we include material that is both above and below the API-hood. Much of the material (especially early on) is stuff you simply need to know to do 3D computer graphics. But we also spend time to explain what is going on inside of OpenGL. This is necessary to understand in order to be a highly competent computer graphics programmer. But also, it is simply fascinating to learn the hows and whys of our amazing computer graphics computational infrastructure.

We will not cover the hardware and compiler aspects of computer graphics in this book. We will also not focus on 2D computer graphics or human–computer interfaces. These topics are all interesting in their own rights but are fairly distinct from the algorithmic side of 3D computer graphics.

In this book, we structure our explanations around OpenGL, a real-time "rasterization-based" rendering environment. We have done this (rather than, say, a "ray tracing–based"

environment) because so much of computer graphics is done in this setting. Anyone, for example, who works in 3D video games needs to master this material (and more). We have chosen the OpenGL API (with the GLSL shading language) in particular, as it can be run on a wide variety of computing platforms.

This book is intended for upper-level computer science/math/physics undergraduate students with at least a year of programming under their belts and at least a rudimentary understanding of linear algebra.

For the Professor

In the following paragraphs, I will describe some of the subtle issues that require some care to get right and are often not taught clearly. I hope that students will master these topics from this book.

Chapters 2–4: In computer graphics, we need to think about points and vectors from both a coordinate-free and a coordinate-full approach. We need to use coordinates to obtain a concrete representation ultimately resulting in our rendered images. But it is often important to represent and transform our points with respect to different coordinate systems. As such it is important to

- Distinguish between a geometric point and the coordinates used to represent that point with respect to some frame.
- Use a notation to explicitly keep track of the basis used for a set of coordinates.
- Distinguish in our notation between matrix equations that represent basis changes and matrix expressions that represent geometric transformations being applied to points.

This ultimately leads to what we call the *left-of* rule, which allows us to interpret matrix expressions and understand with respect to which basis a transformation is acting.

It is our hope that by mastering this explicit notational framework, a student can easily figure out how to do complicated transformations. This is in contrast to the "try lots of orderings and inverses of sequences of matrices until the program does the right thing" approach. One loose inspiration for our approach is the manuscript by Tony DeRose [16].

Chapters 5 and 6: We describe an organized framework for dealing with frames in computer graphics and how this translates into simple 3D OpenGL code. In particular, we derive useful ways to move objects around using a "mixed auxiliary frame." This allows us to, say, rotate an object correctly about its own center but in directions that correspond naturally to those on the screen.

Chapter 7: This is a simple and straightforward description of the quaternion representation for rotations. We also derive how properly to combine quaternions and translation vectors to define a rigid-body transformation data type that multiplies and inverts just like matrices.

Chapter 8: This is a simple and straightforward description of the trackball and arcball rotation interface. We also show why the trackball interface is mouse path–invariant.

Chapter 9: We do a quick and dirty introduction to Bezier and Catmull–Rom splines.

Chapters 10–13: In these chapters, we describe how camera projection is modeled using 4 by 4 matrices. We also describe the fixed function operations in OpenGL. We pay special attention to deriving the correct formulas for interpolating the varying variables. Some of the background on affine functions and interpolation is relegated to appendix B. Many of the subtleties here are described nicely in essays by Jim Blinn [5]. In these chapters, we do not cover details about clipping or rasterization algorithms.

Chapters 14 and 15: We give some simple example shaders for diffuse, shiny, and anisotropic materials. We also point to more advanced real-time rendering techniques such as multipass rendering and shadow mapping. These sections are admittedly too brief. Presumably, students pursuing more aggressive rendering projects will need to learn a lot more about the ever-evolving techniques (and tricks) used in modern real-time rendering.

Chapters 16–18: We cover the basics of why filtering is needed to compute discrete images and how basis functions are used to reconstruct continuous ones. In particular, we show how these two concepts need to be combined to do filtering properly during texture mapping (à la Paul Heckbert's M.S. thesis [28]). We do not delve into the details of Fourier analysis here, as we think it would pull us a bit too far off the main narrative (and, in the end, we use box filters anyway).

Chapter 19: We describe the basic theory of human color perception. From a mathematical perspective, we attempt to be clear about the very definition of what a color is and why such things form a linear space. A related treatment to ours can be found in Feynman's lectures [20]. For many of the technical issues of color computations, we rely on the color FAQ of Charles Poynton [58].

Chapter 20: For completeness, we briefly describe ray tracing computations. As this is not the focus of the course, we only touch on the topic.

Chapter 21: As an introduction to advanced topics in photo-realistic rendering, we do a careful introduction to the physical units for describing light and to the reflection, rendering, and measurement equations. One thing we pay close attention to, for example, is why reflection is measured in the particular units used. A good reference on these basics and more is Eric Veach's Ph.D. thesis [71].

Chapter 22: We outline some of the ways that surfaces are modeled and represented in computer graphics. This is a nontechnical discussion that gives a quick introduction to this rich topic. We do go into enough detail to be able to implement Catmull–Rom subdivision surfaces (assuming you have a mesh data structure handy), as these are a quick and easy way to represent a broad family of surfaces.

Chapter 23: We outline some of the ways animation is done in computer graphics. Again, this is a nontechnical discussion that gives a quick introduction to this rich topic.

One nice place to start for this material is Adrien Treuille's course on the CMU website (http://www.cs.cmu.edu/∼15869-f10).

Appendix A: We try to get up and running as painlessly as possible with a first OpenGL program. Because we will be using a modern version of OpenGL, there is thankfully not much API left to teach anymore. All OpenGL needs to do is manage the shader programs, vertex buffer objects, and textures. We do not teach any of the deprecated elements of old OpenGL, as they are not used in modern computer graphics programming. Appendix A may be read at any time before chapter 6.

Appendix B: We summarize important facts about affine functions. This is helpful for reading chapter 12 and essential for reading chapter 13.

This book covers a bit more than what would be possible to do in a one-semester course. It does not attempt to be an encyclopedic reference to all of the rich topics in computer graphics theory and practice. Additionally, this book stays close to material in wide use today. We do not cover the many current research results and ideas that may one day become part of standard practice.

The website for the book may be found at http://mitpress.mit.edu/foundations.

Aknowledgments

During the development of this book over the past year, I received lots of helpful input from Hamilton Chong, Guillermo Diez-Canas, and Dylan Thurston.

Helpful advice also came from Julie Dorsey, Hugues Hoppe, Zoe Wood, Yuanchen Zhu, and Todd Zickler.

Other comments and assistance came from Gilbert Bernstein, Fredo Durand, Ladislav Kavan, Michael Kazhdan, and Peter-Pike Sloan.

During the development of the course, I was greatly assisted by Xiangfeng Gu, Danil Kirsanov, Chris Mihelich, Pedro Sander, and Abraham Stone.

Over the years, the course has had many talented students acting as teaching assistants. They have contributed in many ways to the evolution of this material and include Brad Andalman, Keith Baldwin, Forrester Cole, Ashley Eden, David Holland, Brad Kittenbrink, Sanjay Mavinkurve, Jay Moorthi, Doug Nachand, Brian Osserman, Ed Park, David Ryu, Razvan Surdulescu, and Geetika Tewari.

Finally, I thank my parents for their years of love and support.