CRC Press
Taylor & Francis Group
AN A K PETERS BOOK

3D Engine Design for Virtual Globes

Patrick Cozzi • Kevin Ring

3D Engine Design for Virtual Globes

Patrick Cozzi
Kevin Ring

Cover design by Francis X. Kelly.
3D cover art by Jason K. Martin.

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed in the United States of America on acid-free paper
Version Date: 20110505

International Standard Book Number: 978-1-56881-711-8 (Hardback)

Library of Congress Cataloging-in-Publication Data

Cozzi, Patrick.
3D Engine design for virtual globes / Patrick Cozzi, Kevin Ring.
p. cm. -- (An A K Peters book)
Includes bibliographical references and index.
ISBN 978-1-56881-711-8 (hardback)
1. Globes--Computer-assisted instruction. 2. Digital mapping. 3. Texture mapping. 4. Computer graphics. I. Ring, Kevin. II. Title.

G3170.C69 2011
912.0285'6693--dc22 2011010460

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the CRC Press Web site at
http://www.crcpress.com

3D Engine Design for Virtual Globes

To my parents, who bought me my first computer in 1994. Honestly, I just wanted to play games; I didn't think anything productive would come of it.

Patrick Says

When I was seven years old, I declared that I wanted to make my own computer games. This book is dedicated to my mom and dad, who thought that was a neat idea.

Kevin Says

Foreword

Do not let the title of this book fool you. What the title tells you is that if you have an interest in learning about high-performance and robust terrain rendering for games, this book is for you. If you are impressed by the features and performance of mapping programs such as NASA World Wind or Google Earth and you want to know how to write software of this type, this book is for you.

Some authors write computer books that promise to tell you everything you need to know about a topic, yet all that is delivered is a smattering of high-level descriptions but no low-level details that are essential to help you bridge the gap between theoretical understanding and practical source code. This is not one of those books. You are given a quality tutorial about globe and terrain rendering; the details about real-time 3D rendering of high-precision data, including actual source code to work with; and the mathematical foundations needed to be an expert in this field. Moreover, you will read about state-of-the-art topics such as geometry clipmapping and other level-of-detail algorithms that deal efficiently with massive terrain datasets. The book's bibliography is extensive, allowing you to investigate the large body of research on which globe rendering is built.

What the title of the book does not tell you is that there are many more chapters and sections about computing with modern hardware in order to exploit parallelism. Included are discussions about multithreaded engine design, out-of-core rendering, task-level parallelism, and the basics necessary to deal with concurrency, synchronization, and shared resources. Although necessary and useful for globe rendering, this material is invaluable for any application that involves scientific computing or visualization and processing of a large amount of data. Effectively, the authors are providing you with two books for the price of one. I prefer to keep only a small number of technical books at my office, opting for books with large information-per-page density. This book is now one of those.

—Dave Eberly

Preface

Planet rendering has a long history in computer graphics. Some of the earliest work was done by Jim Blinn at NASA's Jet Propulsion Laboratory (JPL) in the late 1970s and 80s to create animations of space missions. Perhaps the most famous animations are the flybys of Jupiter, Saturn, Uranus, and Neptune from the Voyager mission.

Today, planet rendering is not just in the hands of NASA. It is at the center of a number of games, such as *Spore* and *EVE Online*. Even non-planet-centric games use globes in creative ways; for example, *Mario Kart Wii* uses a globe to show player locations in online play.

The popularity of virtual globes such as Google Earth, NASA World Wind, Microsoft Bing Maps 3D, and Esri ArcGIS Explorer has also brought significant attention to globe rendering. These applications enable viewing massive real-world datasets for terrain, imagery, vector data, and more.

Given the widespread use of globe rendering, it is surprising that no single book covers the topic. We hope this book fills the gap by providing an in-depth treatment of rendering algorithms utilized by virtual globes. Our focus is on accurately rendering real-world datasets by presenting the core rendering algorithms for globes, terrain, imagery, and vector data.

Our knowledge in this area comes from our experience developing Analytical Graphics, Inc.'s (AGI) Satellite Tool Kit (STK) and Insight3D. STK is a modeling and analysis application for space, defense, and intelligence systems that has incorporated a virtual globe since 1993 (admittedly, we were not working on it back then). Insight3D is a 3D visualization component for aerospace and geographic information systems (GIS) applications. We hope our real-world experience has resulted in a pragmatic discussion of virtual globe rendering.

Intended Audience

This book is written for graphics developers interested in rendering algorithms and engine design for virtual globes, GIS, planets, terrain, and massive worlds. The content is diverse enough that it will appeal to a wide audience: practitioners, researchers, students, and hobbyists. We hope that our survey-style explanations satisfy those looking for an overview or a more theoretical treatment, and our tutorial-style code examples suit those seeking hands-on "in the trenches" coverage.

No background in virtual globes or terrain is required. Our treatment includes both fundamental topics, like rendering ellipsoids and terrain representations, and more advanced topics, such as depth buffer precision and multithreading.

You should have a basic knowledge of computer graphics, including vectors and matrices; experience with a graphics API, such as OpenGL or Direct3D; and some exposure to a shading language. If you understand how to implement a basic shader for per-fragment lighting, you are well equipped. If you are new to graphics—welcome! Our website contains links to resources to get you up to speed: http://www.virtualglobebook.com/.

This is also the place to go for the example code and latest book-related news.

Finally, you should have working knowledge of an object-oriented programming language like C++, C#, or Java.

Acknowledgments

The time and energy of many people went into the making of this book. Without the help of others, the manuscript would not have the same content and quality.

We knew writing a book of this scope would not be an easy task. We owe much of our success to our incredibly understanding and supportive employer, Analytical Graphics, Inc. We thank Paul Graziani, Frank Linsalata, Jimmy Tucholski, Shashank Narayan, and Dave Vallado for their initial support of the project. We also thank Deron Ohlarik, Mike Bartholomew, Tom Fili, Brett Gilbert, Frank Stoner, and Jim Woodburn for their involvement, including reviewing chapters and tirelessly answering questions. In particular, Deron played an instrumental role in the initial phases of our project, and the derivations in Chapter 2 are largely thanks to Jim. We thank Francis Kelly, Jason Martin, and Glenn Warrington for their fantastic work on the cover.

This book may have never been proposed if it were not for the encouragement of our friends at the University of Pennsylvania, namely Norm Badler, Steve Lane, and Joe Kider. Norm initially encouraged the idea and suggested A K Peters as a publisher, who we also owe a great deal of thanks to. In particular, Sarah Cutler, Kara Ebrahim, and Alice and Klaus Peters helped us through the entire process. Eric Haines (Autodesk) also provided a great deal of input to get us started in the right direction.

We're fortunate to have worked with a great group of chapter reviewers, whose feedback helped us make countless improvements. In alphabetical order, they are Quarup Barreirinhas (Google), Eric Bruneton (Laboratoire Jean Kuntzmann), Christian Dick (Technische Universität München), Hugues Hoppe (Microsoft Research), Jukka Jylänki (University of Oulu), Dave Kasik (Boeing), Brano Kemen (Outerra), Anton Frühstück Malischew (Greentube Internet Entertainment Solutions), Emil Persson (Avalanche Studios), Aras Pranckevičius (Unity Technologies), Christophe Riccio (Imagination Technologies), Ian Romanick (Intel), Chris Thorne (VRshed), Jan Paul Van Waveren (id Software), and Mattias Widmark (DICE).

Two reviewers deserve special thanks. Dave Eberly (Geometric Tools, LLC), who has been with us since the start, reviewed several chapters multiple times and always provided encouraging and constructive feedback. Aleksandar Dimitrijević (University of Niš) promptly reviewed many chapters; his enthusiasm for the field is energizing.

Last but not least, we wish to thank our family and friends who have missed us during many nights, weekends, and holidays. (Case-in-point: we are writing this section on Christmas Eve.) In particular, we thank Kristen Ring, Peg Cozzi, Margie Cozzi, Anthony Cozzi, Judy MacIver, David Ring, Christy Rowe, and Kota Chrome.

Dataset Acknowledgments

Virtual globes are fascinating because they provide access to a seemingly limitless amount of GIS data, including terrain, imagery, and vector data. Thankfully, many of these datasets are freely available. We graciously acknowledge the providers of datasets used in this book.

Natural Earth

Natural Earth (http://www.naturalearthdata.com/) provides public domain raster and vector datasets at 1 : 10, 1 : 50, and 1 : 110 million scales. We use the image in Figure 1 and Natural Earth's vector data throughout this book.

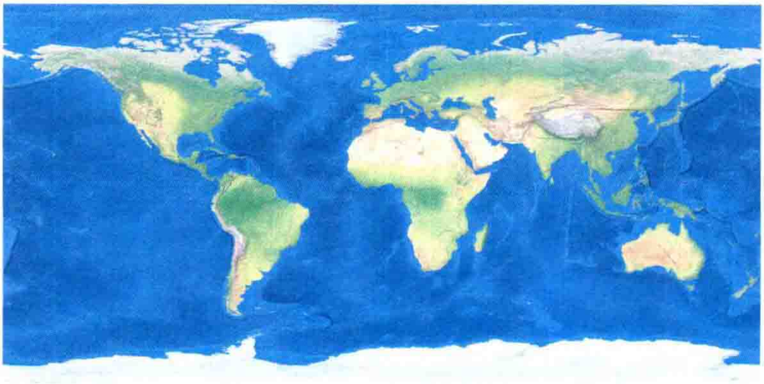


Figure 1. Satellite-derived land imagery with shaded relief and water from Natural Earth.

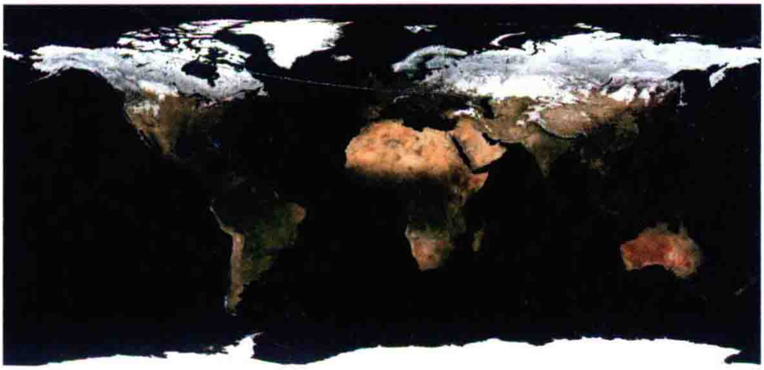
NASA Visible Earth

NASA Visible Earth (http://visibleearth.nasa.gov/) provides a wide array of satellite images. We use the images shown in Figure 2 throughout this book. The images in Figure 2(a) and 2(b) are part of NASA's Blue Marble collection and are credited to Reto Stockli, NASA Earth Observatory. The city lights image in Figure 2(c) is by Craig Mayhew and Robert Simmon, NASA GSFC. The data for this image are courtesy of Marc Imhoff, NASA GSFC, and Christopher Elvidge, NOAA NGDC.

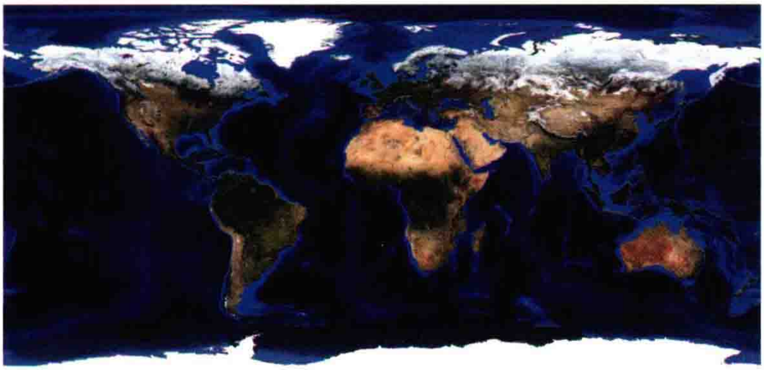
NASA World Wind

We use NASA World Wind's `mergedElevations` terrain dataset (http://worldwindcentral.com/wiki/World_Wind_Data_Sources) in our terrain implementation. This dataset has 10 m resolution terrain for most of the United States, and 90 m resolution data for other parts of the world. It is derived from three sources: the Shuttle Radar Topography Mission (SRTM) from NASA's Jet Propulsion Laboratory;[1] the National Elevation

[1] http://www2.jpl.nasa.gov/srtm/



(a)



(b)

(c)

Figure 2. Images from NASA Visible Earth.

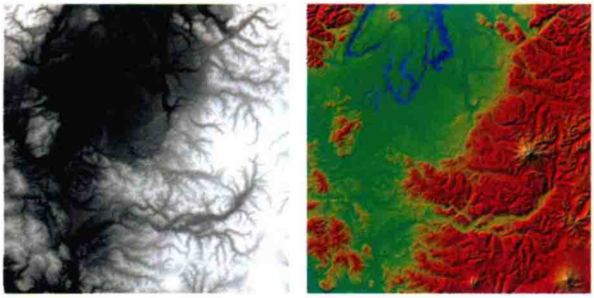
Dataset (NED) from the United States Geological Survey (USGS);[2] and SRTM30_PLUS: SRTM30, coastal and ridge multibeam, estimated topography, from the Institute of Geophysics and Planetary Physics, Scripps Institution of Oceanography, University of California San Diego.[3]

National Atlas of the United States of America

The National Atlas of the United States of America (http://www.nationalatlas.gov/atlasftp.html) provides a plethora of map data at no cost. In our discussion of vector data rendering, we use their airport and Amtrak terminal datasets. We acknowledge the Administration's Research and Innovative Technology Administration/Bureau of Transportation Statistics (RITA/BTS) National Transportation Atlas Databases (NTAD) 2005 for the latter dataset.

Georgia Institute of Technology

Like many developers working on terrain algorithms, we've used the terrain dataset for Puget Sound in Washington state, shown in Figure 3. These data are part of the Large Geometric Models Archive at the Georgia Institute of Technology (http://www.cc.gatech.edu/projects/large_models/ps.html). The original dataset[4] was obtained from the USGS and made available by the University of Washington. This subset was extracted by Peter Lindstrom and Valerio Pascucci.



(a) (b)

Figure 3. (a) A height map and (b) color map (texture) of Puget Sound from the Large Geometric Models Archive at the Georgia Institute of Technology.

[2] http://ned.usgs.gov/

[3] http://topex.ucsd.edu/WWW_html/srtm30_plus.html

[4] http://rocky.ess.washington.edu/data/raster/tenmeter/onebytwo10/

Yusuke Kamiyamane

The icons used in our discussion of vector data rendering were created by Yusuke Kamiyamane, who provides a large icon collection under the Creative Commons Attribution 3.0 license (http://p.yusukekamiyamane.com/).

Feedback

You are encouraged to email us with feedback, suggestions, or corrections at authors@virtualglobebook.com.

Contents

Foreword vii

Preface vii

1 Introduction 1
- 1.1 Rendering Challenges in Virtual Globes 1
- 1.2 Contents Overview 5
- 1.3 OpenGlobe Architecture 8
- 1.4 Conventions 10

I Fundamentals 11

2 Math Foundations 13
- 2.1 Virtual Globe Coordinate Systems 13
- 2.2 Ellipsoid Basics 17
- 2.3 Coordinate Transformations 22
- 2.4 Curves on an Ellipsoid 34
- 2.5 Resources 39

3 Renderer Design 41
- 3.1 The Need for a Renderer 42
- 3.2 Bird's-Eye View 46
- 3.3 State Management 50
- 3.4 Shaders 63
- 3.5 Vertex Data 84
- 3.6 Textures 101
- 3.7 Framebuffers 112
- 3.8 Putting It All Together: Rendering a Triangle 115
- 3.9 Resources 119

4 Globe Rendering 121
4.1 Tessellation 121
4.2 Shading 133
4.3 GPU Ray Casting 149
4.4 Resources 154

II Precision 155

5 Vertex Transform Precision 157
5.1 Jittering Explained 158
5.2 Rendering Relative to Center 164
5.3 Rendering Relative to Eye Using the CPU 169
5.4 Rendering Relative to Eye Using the GPU 171
5.5 Recommendations 177
5.6 Resources 180

6 Depth Buffer Precision 181
6.1 Causes of Depth Buffer Errors 181
6.2 Basic Solutions 188
6.3 Complementary Depth Buffering 189
6.4 Logarithmic Depth Buffer 191
6.5 Rendering with Multiple Frustums 194
6.6 W-Buffer 198
6.7 Algorithms Summary 198
6.8 Resources 199

III Vector Data 201

7 Vector Data and Polylines 203
7.1 Sources of Vector Data 203
7.2 Combating Z-Fighting 204
7.3 Polylines 207
7.4 Resources 219

8 Polygons 221
8.1 Render to Texture 221
8.2 Tessellating Polygons 222
8.3 Polygons on Terrain 241
8.4 Resources 250

9 Billboards 251
9.1 Basic Rendering 252
9.2 Minimizing Texture Switches 258
9.3 Origins and Offsets 267
9.4 Rendering Text 271
9.5 Resources 273

10 Exploiting Parallelism in Resource Preparation 275
10.1 Parallelism Everywhere 275
10.2 Task-Level Parallelism in Virtual Globes 278
10.3 Architectures for Multithreading 280
10.4 Multithreading with OpenGL 292
10.5 Resources 304

IV Terrain 305

11 Terrain Basics 307
11.1 Terrain Representations 308
11.2 Rendering Height Maps 313
11.3 Computing Normals 335
11.4 Shading 343
11.5 Resources 363

12 Massive-Terrain Rendering 365
12.1 Level of Detail 367
12.2 Preprocessing 376
12.3 Out-of-Core Rendering 381
12.4 Culling 390
12.5 Resources 400

13 Geometry Clipmapping 403
13.1 The Clipmap Pyramid 406
13.2 Vertex Buffers 408
13.3 Vertex and Fragment Shaders 411
13.4 Blending 414
13.5 Clipmap Update 417
13.6 Shading 435
13.7 Geometry Clipmapping on a Globe 436
13.8 Resources 443

14 Chunked LOD 445
14.1 Chunks 447
14.2 Selection 448
14.3 Cracks between Chunks 449
14.4 Switching 450
14.5 Generation 452
14.6 Shading 459
14.7 Out-of-Core Rendering 460
14.8 Chunked LOD on a Globe 462
14.9 Chunked LOD Compared to Geometry Clipmapping . . . 463
14.10 Resources 465

A Implementing a Message Queue 467

Bibliography 477

Index 491