

Ray Tracing: A Tool for All

Jon Peddie

Ray Tracing: A Tool for All



Jon Peddie
Jon Peddie Research
Belvedere Tiburon, CA, USA

ISBN 978-3-030-17489-7 ISBN 978-3-030-17490-3 (eBook)
<https://doi.org/10.1007/978-3-030-17490-3>

© Springer Nature Switzerland AG 2019

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, 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.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Foreword I

Shaded rendering has been one of the central topics of computer graphics research since the 1960s. Over the following decades, researchers have developed rendering techniques that evolved step-by-step from smooth shading to realistic reflections and ultimately to a level of realism that allows us to ignore the fact that the rendered images are not real.

Back in the early 1980s, Jon Peddie and I met for breakfast one weekend morning to discuss ways to commercialize realistic rendering. At the time, the idea seemed far-fetched. Today, it has become commonplace and it has found applications we had not imagined.

Making use of realistic rendering, whether for Hollywood special effects, video games, or redecorating your living room still requires an understanding of what it is and how it works. Papers, journals, books, and courses dive into the topic and open up the world of rendering to the average programmer. But what if you are not a programmer? Today, most of us want to use the technology without programming it from scratch. Regardless, a sophisticated graphics system requires some understanding of the technology in order to get the most use from it. That's where Jon Peddie's text comes into play.

Somewhere between the ten thousand foot overview and the vast collection of GPU code hiding beneath the surface is a level of explanation and understanding that provides a prospective user with enough background to get started. This book covers that range in detail, but in a manner that can be understood without reviewing graduate-level mathematics. It does a particularly good job of identifying both hardware and software resources to enable a beginning practitioner to get up and running.

To me personally, the most compelling sections of the book are the ones that fall into the category labeled "I didn't know that!" The number of contributors and the diversity of approaches that have brought realistic rendering to its current state are remarkable. Rather than attempting to condense this all into a summary, Jon has opted for completeness and has given the reader a full view of the topic.

In essence, this book is a story. It explains the technology, the applications, and the products, while also providing a history. You, the reader, don't need to spend 40 years writing and reviewing technical papers on rendering and ray tracing. Just read the book.

Chapel Hill, North Carolina

Turner Whitted

Foreword II

Ray tracing is a topic that has inspired many engineers, artists, and storytellers. For some, it is a computer graphics course; for others, it is a degree; and for many—an entire career.

I count myself as one of those people so inspired. One day strolling through the engineering library, I browsed through a book entitled simply “Introduction to Ray Tracing.” It instantly captured my imagination. The images were strikingly beautiful (compared to the state of the art the time), the math was approachable, and it only took a few hours to produce a single image! I soon learned that I was not alone—that other students and researchers were exploring the frontiers “photorealism,” or the idea that a computer-generated image could be indistinguishable from a photograph.

For me, this inspiration launched a career. I am now a Vice President at Nvidia where I lead a team of dedicated engineers who are striving to make ray tracing fast enough to be used in real-time computer graphics. The goal is to bring the techniques that have brought advancements in visual effects and animation to gaming and design. Many in the industry share that goal.

It was this role that introduced me to Jon Peddie and his vast sphere of influence. I quickly learned that he travels the world in pursuit of technology, especially computer graphics, and ray tracing in particular. He is always learning, asking questions, probing technology, and getting all sides of a story. Along with Kathleen Maher, Jon integrates this information into some of the most influential reports in the computer graphics industry, including TechWatch and the JPR Workstation report. In addition, Jon gives countless lectures, serves on advisory boards, and has been recognized by ACM and CAAD for his efforts.

That is why I was intrigued when he told me of his plan to write a book on ray tracing. Many technical books have covered this topic in great depth, and Jon mentions many of them herein. However, “Ray Tracing: A Tool for All” brings an entirely fresh perspective to the topic. While he covers the technology and business in great depth, it is approachable by technical and non-technical readers alike.

Ray tracing has roots in medieval times, but received first attention for computer graphics via a paper by Turner Whitted at SIGGRAPH 1979. Ray tracing operates by simulating the physics of light as particles that interact with various surfaces, using very few simplifying assumptions. Because the human visual system is highly attuned to lighting in the physical world, subtle details can make the difference between an object looking “realistic” and “fake.” Ray tracing can capture these effects, such as global illumination, soft shadows and accurate materials. Consequently, ray tracing is nearly ubiquitous in computer animation and visual effects industry. It is also rapidly becoming the standard in product design, marketing, and even real-time gaming.

A ray tracing program can be simple enough to fit on a business card, but turning it into a fully functional system (or renderer) results in very large sophisticated software. The results can be undeniably beautiful, and this book highlights many of these examples.

What Jon has done is take all of this technical excitement—the passions of inventors, the curiosity of a student or researcher, the creativity of the artists—and mapped it to the ecosystem and companies in the modern world. He covers the businesses around ray tracing, the interplay between the technology and the companies, and he speculates on what the future will bring to the industry. True to Jon’s reputation, the book is filled with facts, data, and unique insight. It discusses the history, the workflows, the research papers, the hardware, the start-ups, and the primary technical challenges being tackled in the industry today.

Behind the technology and businesses, the primary goal of computer graphics is to use a visual illusion to tell a story. Whether used for entertainment, a product introduction, or to gain insight—ray tracing has found application in a broad set of industries. Jon outlines these applications and ecosystems in a clear manner.

I learned a lot from reading “Ray Tracing: A Tool for All.” It collects the under-documented aspects of computer graphics and paints a portrait that blends both technology and business. I expect that it will help inspire even more people to join in the quest of photorealistic rendering.

Dr. Steven G. Parker
Vice President
Professional Graphics, Nvidia
Chapel Hill, North Carolina

Acknowledgements

How could anyone write a book like this without having too many friends? I've met so many people over the years, starting with Turner Whited in 1979, who have made incredible discoveries and inventions. And then, demonstrating their extraordinary grace and charity, they took the time to edit (and mostly correct) the material I sent to them—I am truly blessed.

I know I'm missing someone or two in this list, and to you, should you read this, I am truly sorry—call me, I'll make amends.

My benefactors and mentors—the folks who really know what ray tracing is about, this book really would not exist without these people.

Helen Desmond, my patient and supportive editor at Springer

Alexander Keller, Nvidia—great comments and polite nudges

Alexandra Constantine, Autodesk

Ankit Patel, Nvidia

Brian Savery, AMD

Colin McLaughlin, Chaos—invaluable resource

Daniel Pohl, Intel—brilliant developer

David Harold, Imagination Technologies

David Laur, Pixar

David McGavran, Maxon

David Tracy, Chaos

Frederic Servant, Autodesk—a real pro

Glen Matthews, AMD

Henrik Edstrom, Autodesk

Igor Zanic, Houdini

Jama Jurabaev, Lucas Arts

John Hart, University Illinois

Joseph Taraborrelli, Sony

Josh Mings, Luxion

Katrina Felicano-Stoddard, Intel

Lon Grohs, Chaos

Ludwig von Reiche, Nvidia

Lynette Clee, Chaos

Oliver Meiseberg, Maxon

Phillip Miller, Chaos

Rolf Herken, ViewMagic

Sean Morrison, BRL-CAD

Steven Parker, Nvidia—who didn't sign up to be my editor and ended up reading every word

Tom Svilans, 3D modeler

Ton Roosendaal, Blender

Turner Whited, Nvidia (ret)—a big debt

My colleagues and collaborators

Kathleen Maher—encouraged me to see it through

Ruchike Saini

Jaydeep Bhattacharjee

Robert Dow

Peter McGuinness

Contents

1 Preface	1
1.1 About the Cover	3
1.2 Terminology and Definitions	4
2 Introduction	7
2.1 Who Needs It?	8
2.2 Ray Tracing Isn't New	9
2.3 A Little History	11
2.4 Ray Tracing not New	12
2.4.1 From Humble Beginnings	14
2.5 Realism, Accuracy, and Functionality	15
2.5.1 Three Types of Realism in Computer Graphics	16
2.5.2 Stylistic Versus Photorealistic	19
2.6 Technical Papers and Books	23
2.7 Material Libraries Critical	24
2.8 Rendering Becomes a Function of Price	24
2.9 Shortcuts and Semiconductors—The Need for Speed	25
2.10 Challenges	27
References	27
3 The Rendering Industry	29
3.1 Leading Companies Rendering in AEC and Product Design	30
3.2 The Future	31
4 The Continuum	33
4.1 The Rendering Equation	35
4.2 Scanline Rendering	36
4.2.1 Z-Buffering	37
4.2.2 Painter's Algorithm	38

- 4.3 Ray Tracing 39
 - 4.3.1 Path Tracing 46
 - 4.3.2 The Difference Between Path Tracing and Ray Tracing 47
 - 4.3.3 Noise in Ray Tracing 47
 - 4.3.4 Global Illumination 49
 - 4.3.5 The Difference Between Ray Tracing and Ray Casting 49
 - 4.3.6 Recursive Ray Tracing 52
- 4.4 Photon Mapping 53
- 4.5 Brute Force 55
- 4.6 Radiosity 56
- 4.7 Light-Field Rendering 57
 - 4.7.1 Voxels 59
- 4.8 Problems Ray Tracing Doesn't Solve 61
 - 4.8.1 Photorealism 61
 - 4.8.2 Surface Complexity 62
 - 4.8.3 Scale 62
- 4.9 Summary 63
- References 64
- 5 Work Flow and Material Standards 65**
 - 5.1 Biased Versus Unbiased 65
 - 5.1.1 Biased Versus Consistent 66
 - 5.1.2 Radiosity 66
 - 5.1.3 Rasterization 67
 - 5.2 Importance of Material Library 67
 - 5.2.1 Standards (USPs, OSL, Etc.) 69
 - 5.2.2 Physically Based Rendering 71
 - 5.2.3 Allegorithmic's Substance Designer 73
 - 5.2.4 Everyday Material Collection 75
 - 5.2.5 MaterialX 76
 - 5.2.6 Nvidia's MDL 76
 - 5.2.7 X-Rite's AxF 79
 - 5.3 Quality Issues 80
 - 5.3.1 Skin and Subsurface Scattering 81
 - 5.3.2 Variance-Based Adaptive Sampling 83
 - 5.3.3 Hybrid 84
 - 5.3.4 Summary 84
 - 5.4 Importance of HDR Monitors 85
 - 5.5 Importance of Full-Color Printers 89
 - References 90

- 6 Applications of Ray Tracing** 91
 - 6.1 The Pipeline 92
 - 6.1.1 Conception—STAGE ONE 93
 - 6.1.2 Design and Engineering—STAGE TWO 111
 - 6.1.3 Manufacturing and Production—STAGE THREE 119
 - 6.1.4 Marketing—STAGE FOUR 122
 - 6.2 Summary 127
 - References 128
- 7 Ray-Tracing Hardware** 129
 - 7.1 Shortcuts and Semiconductors—The Need for Speed 129
 - 7.2 Local 132
 - 7.2.1 CPU 132
 - 7.2.2 GPU 135
 - 7.2.3 Dedicated 153
 - 7.2.4 RT on Mobiles 157
 - 7.3 Remote 161
 - 7.3.1 Cloud-Based Visualization 162
 - 7.3.2 Public Cloud Rendering Services 168
 - 7.3.3 Private Rendering Services—Farms 171
 - 7.3.4 Rendering Service Organizations 172
 - 7.4 Benchmarking Ray Tracing 173
 - 7.4.1 SPEC 173
 - 7.4.2 Underwriter Labs Futuremark 176
 - 7.4.3 Blender’s Open Data Benchmark 176
 - 7.4.4 Chaos Group 178
 - 7.4.5 Redshift Benchmark 179
 - 7.4.6 Summary 179
 - References 180
- 8 Ray-Tracing Programs and Plug-ins** 181
 - 8.1 Stand-Alone Ray-Tracing Programs 184
 - 8.1.1 3Delight—Illumination Research 184
 - 8.1.2 Appleseed 187
 - 8.1.3 Arnold—Autodesk (Solid Angle) 189
 - 8.1.4 Cero—PTC 194
 - 8.1.5 Indigo Renderer—Glare Technologies 195
 - 8.1.6 Cinema 4D—Maxon Computer 199
 - 8.1.7 Corona Renderer—Render Legion 203
 - 8.1.8 Iray—Nvidia 203
 - 8.1.9 KeyShot—Luxion 213
 - 8.1.10 Lumion 8 and Pro—Act-3D B.V 216
 - 8.1.11 Maxwell Render—Next Limit 220
 - 8.1.12 Mitsuba 225

- 8.1.13 Nebula Render 228
- 8.1.14 OctaneRender—Otoy 231
- 8.1.15 OSPRay—Intel 233
- 8.1.16 Pica—SEED/Electronics Arts 236
- 8.1.17 ProRender—AMD 237
- 8.1.18 POV-Ray 239
- 8.1.19 Redshift Renderer 242
- 8.1.20 RenderMan—Pixar 246
- 8.1.21 Rigid Gems—FerioWorks.LLC 251
- 8.1.22 Tachyon 252
- 8.1.23 V-Ray—Chaos Group 253
- 8.1.24 VRED—Autodesk 264
- 8.1.25 Other 267
- 8.1.26 Lightworks Design 267
- 8.1.27 Manuka—Weta 269
- 8.2 Integrated (Programs with Native Ray Tracers) 272
 - 8.2.1 Cycles—Blender 272
 - 8.2.2 Carrara—Daz 3D 276
 - 8.2.3 Dimension CC—Adobe 279
 - 8.2.4 Mantra—SideFX 282
 - 8.2.5 ART (Autodesk Ray Tracer) 286
 - 8.2.6 Unreal Studio—Epic Games 287
 - 8.2.7 Visualize—Dassault Systèmes/SolidWorks 290
 - 8.2.8 PhotoView 360—Dassault Systèmes/SolidWorks 295
- 8.3 Plug-in Programs 298
 - 8.3.1 3Delight—Illumination Technologies 298
 - 8.3.2 Arnold—Autodesk 298
 - 8.3.3 Corona Renderer—Chaos Group (Legion Team) 298
 - 8.3.4 Cycles—Blender 304
 - 8.3.5 finalRender—Cebas 304
 - 8.3.6 Iray—Nvidia 306
 - 8.3.7 KeyShot—Luxion 306
 - 8.3.8 Lumion 307
 - 8.3.9 LuxCoreRender 307
 - 8.3.10 Maxwell 309
 - 8.3.11 ProRender 310
 - 8.3.12 Redshift 310
 - 8.3.13 V-Ray, Chaos Group 310
- 8.4 Middleware 313
 - 8.4.1 Embree 314
 - 8.4.2 OptiX—Nvidia 315
 - 8.4.3 Radeon-Rays—AMD 317

- 8.5 Cloud-Based 321
 - 8.5.1 CL3VER—Cloud Rendering 322
 - 8.5.2 OneRender—Prefixa 323
 - 8.5.3 RealityServer—Migenius 323
- 8.6 Other 325
 - 8.6.1 The Ray Tracer Challenge 325
 - 8.6.2 Tiny Ray Tracer Fits in 64 Bytes 326
 - 8.6.3 A Ray Tracer for Bare Metal x86 327
 - 8.6.4 Tiny Metaball Ray Tracer in x86/x87 Assembly 328
- References 329

- Appendix A** 331
- Glossary** 339
- Index** 353

List of Figures

Fig. 1.1	Saya. <i>Source</i> Teruyuki and Yuki Ishikawa	2
Fig. 1.2	Rendered and photograph of the World Trade Center.	4
Fig. 2.1	BRL-CAD overview	10
Fig. 2.2	Famous SGI cube logo was created using BRL-CAD ray-tracing program. <i>Credit</i> Sean Morrison.	10
Fig. 2.3	Appel projected light at a 3D computer model and displayed the results on a plotter using a form of tone mapping to create light and dark areas. <i>Source</i> Arthur Appel	12
Fig. 2.4	Translucent Bradley fighting vehicle rendered in BRL-CAD. <i>Source</i> Sean Morrison (2002)	13
Fig. 2.5	Lightning McQueen in Cars 2—circa 2011. <i>Source</i> Wikipedia	15
Fig. 2.6	A hare versus a variation of the Utah teapot. <i>Source</i> Wikipedia	16
Fig. 2.7	In animation you want perfect reflections and shadows, but the objects may be pure fantasy: Toy Story. <i>Source</i> Pixar Wiki	18
Fig. 2.8	Ray tracing used in stylistic fantasy film, “Terminator 6”. <i>Source</i> Jama Jurabaev	20
Fig. 2.9	Ray-traced robot before (left) and after environmental layering (right). <i>Source</i> Jama Jurabaev	20
Fig. 2.10	Car design is one of the most popular and demanding ray-tracing applications (Soviet Moskvich 412—1974, Daniartist90)	21
Fig. 2.11	Ray-tracing programs	22
Fig. 2.12	Papers on ray tracing published in academic journals since 1982.	23
Fig. 2.13	An example of various materials applied to the same object. <i>Source</i> Epic	25

Fig. 2.14 Use of variance-based adaptive sampling on this model of Christmas cookies from Autodesk 3ds Max provided a better final image in record time. *Source* Chaos Group 26

Fig. 4.1 The rendering equation describes how light behaves 35

Fig. 4.2 Scanline algorithm example 37

Fig. 4.3 Example of the painter’s algorithm. 39

Fig. 4.4 Rays in ray tracing 40

Fig. 4.5 Still life with RenderMan 20. *Source* Dylan Sisson
RenderMan community. 41

Fig. 4.6 Relative performance versus quality in various modes of ray tracing 42

Fig. 4.7 Star Wars stormtroopers rendered in real time with ray tracing (Nvidia) 43

Fig. 4.8 Crytek Sponza scene—a common scene for showcasing global illumination (model from McGuire Graphics Data) 44

Fig. 4.9 This scene renders at about 30–40 fps using just one ray per pixel. It is noisy but one can get a good sense of what the image looks like. *Source* Notch 48

Fig. 4.10 Iconic Wolfenstein 3D screenshot. *Source* Wikipedia. 50

Fig. 4.11 Ray-traced image of glasses showing the perfect reflections and refractions, as well as shadows. *Source* Gilles Tran. 51

Fig. 4.12 Coffee room rendering using OneRender ray tracer. *Source* OneRender 53

Fig. 4.13 Zero GI bounces with photon mapping. 54

Fig. 4.14 Multiple and secondary GI bounces with photon mapping 55

Fig. 4.15 Scene rendered with radiosity renderer and visualizer (By David Bařina, Kamil Dudka, Jakub Filák, Lukáš) 56

Fig. 4.16 Light-field rendering (IEEE VR 2003 tutorial) 57

Fig. 4.17 Field of voxel-rendered oranges was rendered and shown in real time (25–40 fps at 768 lines) in 2009. *Source* Unlimited detail. 59

Fig. 4.18 Point-to-voxels surfacing example. *Source* Nvidia 60

Fig. 4.19 Continuum of rendering 63

Fig. 5.1 Light’s interaction with materials determines the image’s believability 68

Fig. 5.2 Materials are used in photorealistic and fantasy images. *Source* Blender.org 68

Fig. 5.3 Blender’s Cycles’ material library. *Source* Blender.org. 69

Fig. 5.4 Diffusion and reflections 71

Fig. 5.5 Free PBR where you can download 100% free PBR materials and texture files. *Source* FreePBR.com. 72

Fig. 5.6 An example of materials from Allegorhmic’s substance source library (image Raphael Rau) 74

Fig. 5.7	Greyscalegorilla material collection includes 12 different categories of in-demand textures and shaders	75
Fig. 5.8	MaterialX has been used Lucas films (© and ™ 2017 Lucasfilm Ltd. all rights reserved)	76
Fig. 5.9	Several different materials (Nvidia).	77
Fig. 5.10	Realistic skin is rendered using subsurface scattering and specific materials. <i>Source</i> Nvidia	81
Fig. 5.11	Comparison of BRDF to BSSRDF reflections. <i>Source</i> Mike Seymour	82
Fig. 5.12	Translucent grapes. <i>Source</i> Pixar	82
Fig. 5.13	Chaos Group says the use of variance-based adaptive sampling on this model of Christmas cookies from Autodesk 3ds Max provided a better final image in record time. <i>Source</i> Chaos Group.	83
Fig. 5.14	CIE 1931 chromaticity diagram (Wikipedia).	87
Fig. 5.15	Color/luminance volume: BT.2020 (10,000 nits) versus BT.709 (100 nits); Yxy. <i>Source</i> Sony.	88
Fig. 5.16	Various gamma curves (Insight Media).	89
Fig. 5.17	The colors of CMYK	90
Fig. 6.1	This is an example of a Boeing 797 blended wing concept airplane that was never built - realistic looking isn't it. <i>Source</i> Wikipedia—Popular Science magazine	94
Fig. 6.2	The Ford GT90 was never built. <i>Source</i> Ford.	94
Fig. 6.3	241-floor, 3162-ft-high structure would be named “The Bride” and sit in the middle of Basra, Iraq. <i>Courtesy</i> of AMBS architects	95
Fig. 6.4	Proposed Airbus A390. <i>Source</i> Airbus	95
Fig. 6.5	The most popular video games sold in the USA in 2017. <i>Source</i> Statista	97
Fig. 6.6	Wolfenstein 3D made use of ray casting algorithms in 1992. <i>Source</i> Wikipedia	99
Fig. 6.7	A scene from Schied’s ray-traced version of Quake II. <i>Source</i> Christoph Schied (2018).	100
Fig. 6.8	Ray-traced Quake: the water reflects the environment and the player. <i>Source</i> Pohl (2006).	101
Fig. 6.9	Lara’s unrealistic dirty face and arms	102
Fig. 6.10	Artyom’s perfectly clean mask and gun after a decade of fighting in tunnels and snow storms	103
Fig. 6.11	Car reflecting a nearby fire in Battlefield V	103
Fig. 6.12	Perfectly flat, perfectly clean Dutch windows in WWII Amsterdam	104
Fig. 6.13	Market scene in Shadow of the Tomb Raider.	105
Fig. 6.14	A ray-traced scene from A4’s “Metro Exodus”.	106

Fig. 6.15 Proposed mixed-use development. *Source* Tom Svilans, rendered with Indigo Renderer 107

Fig. 6.16 London’s 20 Fenchurch Street tower. *Source* Nvidia 108

Fig. 6.17 Doing the analysis before would have revealed the risk. *Source* Nvidia. 108

Fig. 6.18 HBO logo ray-traced and animated as molten metal. *Source* © HBO 109

Fig. 6.19 Is this the future for automobiles?. 110

Fig. 6.20 Cut glass and jewelry design requires ray tracing to catch all the reflections of the piece and show it off best (Rendered in FluidRay RT, design by Manuel Angel Piñeiro Solsona) 113

Fig. 6.21 Marvelous Designer’s user interface and design tools. *Source* Marvelous Designer 115

Fig. 6.22 Fashion design with Clo3d. *Source* Clo3d 116

Fig. 6.23 Ray tracing used in packing design and marketing. *Source* iC3D. 117

Fig. 6.24 LightTools’ illumination and lighting design. *Source* LightTools. 121

Fig. 6.25 DIAL’s lighting design software user’s interface. *Source* DIAL 121

Fig. 6.26 Gran Turismo the Circuit de Barcelona-Catalunya. *Source* Sony’s Polyphony Digital 122

Fig. 6.27 Ray-tracing pipeline 123

Fig. 6.28 Ray-traced car with neutral background; any scene could be applied. *Source* Chevrolet 124

Fig. 6.29 Modern office buildings. *Source* Mike Mareen 124

Fig. 6.30 Consumer product with neutral background. *Source* V-Ray 125

Fig. 6.31 Ryff lets advertisers place any virtual object into commercials and films. *Source* Ryff 125

Fig. 6.32 Packaging complex, nonlinear reflective surface containers. *Source* Creative Edge Software. 126

Fig. 6.33 Notch makes clever uses of denoising filters to produce high-quality ray-traced videos in real time. *Source* Notch. 127

Fig. 7.1 Use of variance-based adaptive sampling on this model of Christmas cookies from Autodesk 3ds Max provided a better final image in record time. *Source* Chaos Group 130

Fig. 7.2 Workload distribution through the pipeline. 132

Fig. 7.3 Amdahl’s Law. *Source* Wikipedia. 134

Fig. 7.4 Portion of a 600 x 400-pixel image from Parker’s system ran at 15 frames per second. *Source* Steve Parker 138

Fig. 7.5 Nvidia’s last decade of GPUs, served more than just graphics applications. 142

Fig. 7.6 RTX technology on Volta accelerates ray tracing through machine learning. *Source* Nvidia 144

Fig. 7.7 Fast 8- and 4-bit integer processing improves inferencing performance and power efficiency. *Source* Nvidia 145

Fig. 7.8 Turing’s RT Core focuses on determination of cumbersome ray/object intersection. *Source* Nvidia 146

Fig. 7.9 Hybrid rendering pipeline 148

Fig. 7.10 Blinn’s law of render time versus processor performance over time. 152

Fig. 7.11 Real-time renderings on the RPU prototype using a single FPGA running at 66 MHz and 512 × 384 resolution: SPD Balls (1.2 fps, with shadows and refractions), a conference room (5.5 fps, without shadows), reflective and refractive spheres-RT in an office (4.5 fps), and UT2003 a scene from a current computer game (7.5 fps, precomputed illumination) 155

Fig. 7.12 PowerVR ray tracing delivering real-time, photorealistic rendering. 156

Fig. 7.13 Physically based rendering for the whole team: Nvidia’s 56 TFLOPS (and \$50,000) Quadro VCA. *Source* Nvidia 158

Fig. 7.14 Dynamically aligned structures versus conventional Path Tracing 160

Fig. 7.15 Real-time ray-tracing performance comparison. *Source* Adshir 161

Fig. 7.16 Cloud-based visualization car design concept. *Source* Autodesk 162

Fig. 7.17 Rendering created in REDsdk 4.3; note the cloudy skies in the background. *Source* intrimSIM 164

Fig. 7.18 OSPRay parallel rendering on TACC’s 328 Megapixel Stallion Tiled Display. *Source* Intel 166

Fig. 7.19 Suplex development rendering. *Source* Super-Cheap Architectural Renders 172

Fig. 7.20 “Tribute to Myrna Loy” by Ive (2008). The figure is Vicky 4.1 from DAZ. The author, Ive, created it with Blender by using all images of her that he could find as reference. Rendered with POV-Ray beta 25 using 7 light sources (and the “area_illumination” feature) 174

Fig. 7.21 Scene from the updated LuxRender workload. 175

Fig. 7.22 Real-time ray tracing promises to bring new levels of realism to in-game graphics. *Source* Underwriter Labs 177

Fig. 7.23 Blender’s Cycles six benchmark test scenes 178

Fig. 7.24 V-Ray benchmark tests. *Source* Chaos Group. 178

Fig. 8.1 Rendering in the cloud using GPUs. *Source* OneRender 182

Fig. 8.2 Ray-tracing taxonomy. 183

Fig. 8.3 Start-up of ray-tracing companies over time 183

Fig. 8.4 An example of 3Delight capabilities. *Source* Illumination Research 185

Fig. 8.5 Country Kitchen by Blend Swap user Jay-Artist. 188

Fig. 8.6 Diffusion versus random subsurface scattering. *Source* Autodesk. 191

Fig. 8.7 Creo Render Studio uses Luxion’s KeyShot (PTC) 194

Fig. 8.8 A demonstration of Indigo Renderer. *Source* Glare Technologies/Indigo 196

Fig. 8.9 A demonstration of fast flexible region rendering. *Source* Glare Technologies/Indigo 196

Fig. 8.10 RGB color curves. *Source* Glare Technologies/Indigo 197

Fig. 8.11 An example of sequence overrides which can render multiple regions at once. *Source* Glare Technologies/Indigo. 198

Fig. 8.12 Maxwell’s ProRender comes with an extensive material library (Maxwell) 201

Fig. 8.13 Cinema 4D rendering in Vectorworks’ Renderworks visualization program. *Source* Vectorworks 202

Fig. 8.14 A living room rendered using Iray. *Source* Nvidia 205

Fig. 8.15 Iray with artificial intelligence on and off. *Source* Nvidia. 207

Fig. 8.16 Iray with virtual reality. *Source* Nvidia. 207

Fig. 8.17 Physically based lighting. *Source* Nvidia 208

Fig. 8.18 Physically based material. *Source* Nvidia 208

Fig. 8.19 Changing impact of color control through a material. *Source* Nvidia 209

Fig. 8.20 Changing impact of distance control through material. *Source* Nvidia 209

Fig. 8.21 Changing impact of particle density. *Source* Nvidia 209

Fig. 8.22 Changing impact of change of index of refraction. *Source* Nvidia 210

Fig. 8.23 Comparison between photorealistic and interactive processes. *Source* Nvidia 210

Fig. 8.24 Light path expressions. *Source* Nvidia 211

Fig. 8.25 Nvidia’s ray-traced visualization of its proposed building Endeavor. *Source* Nvidia 212

Fig. 8.26 Abilities of the KeyShot program. The initial picture is the input with the final picture being the last drawn image from the tool. *Source* Luxion 214

Fig. 8.27 KeyShot has increased stability and improved workflow options. *Source* Specialized Levo by TB&O. 215

Fig. 8.28 Introductory image for Lumion 9. *Source* Act-3D B.V. 217

Fig. 8.29 Image showing the ability to use a hand-drawn outline. *Source* Act-3D B.V. 218

Fig. 8.30 Image showing the soft and fine shadows feature.
Source Act-3D B.V. 218

Fig. 8.31 Image showing the new grouping function of Lumion.
Source Act-3D B.V. 219

Fig. 8.32 Image showing the softening of hard images.
Source Act-3D B.V. 219

Fig. 8.33 Maxwell Studio has been used in several architectural presentations 223

Fig. 8.34 Before and after example of the denoising program.
Source Next Limit 223

Fig. 8.35 Multilight offers the flexibility to change lights even without a Maxwell license. *Source* Net Limit 225

Fig. 8.36 Voxelized scarf model rendered using full multiple scattering and an anisotropic scattering model (microflakes). *Dataset* courtesy Jon Kaldoe and Manuel Vargas 227

Fig. 8.37 Golden dragon rendered with a model downloaded on Archive 3D. 229

Fig. 8.38 Urban exterior modeled by Hai le. Sun and the sky are the light sources. Cube map is courtesy of Spiney 229

Fig. 8.39 A BMW i8 downloaded model with an aluminum material. Lightning is mainly from an environment map made by Emil Persson 230

Fig. 8.40 OctaneRender image of a living room. *Source* Otoy. 232

Fig. 8.41 OSPRay’s software stack (Intel) 235

Fig. 8.42 Electronic Art’s SEEd Pico AI simulator using ray tracing. *Source* EA 237

Fig. 8.43 AMD’s out-of-core render (AMD) 238

Fig. 8.44 AMD’s ProRender road map 238

Fig. 8.45 A ray-traced image of glasses rendered in POV-Ray showing the perfect reflections and refractions, as well as shadows.
Source Gilles Tran 240

Fig. 8.46 Out-of-core geometry and textures. *Source* Redshift. 243

Fig. 8.47 Without (left) and with (right) GI—notice the color bleeding.
Source Redshift. 244

Fig. 8.48 A scene out of Incredibles 2. *Source* Pixar/Disney 247

Fig. 8.49 Impact of MNEE on the image. *Source* Pixar/Disney. 248

Fig. 8.50 Rendered by XPU, a massive scene from Coco (without shaders and lights) 250

Fig. 8.51 Sparkly ray-traced gems. *Source* FerioWorks 252

Fig. 8.52 Satellite tobacco mosaic virus molecular graphics produced in VMD and rendered using Tachyon (John Stone) 253

Fig. 8.53 Architectural image rendered in V-Ray. *Source* Chaos. 254

Fig. 8.54 Ray-traced image rendered in V-Ray for 3ds Max.
Source © Toni Bratincevic 255

Fig. 8.55	Image rendered in V-Ray GPU. <i>Source</i> © Double Aye	257
Fig. 8.56	Screenshot of V-Ray for Unreal. Image courtesy of Chaos Group	259
Fig. 8.57	Screenshot of Project Lavina ray tracing 300 billion triangles in real time. Image courtesy of Chaos Group	259
Fig. 8.58	Image rendered in V-Ray using VRscans scanned materials. <i>Source</i> © Visual State.	260
Fig. 8.59	Procedural Stochastic Flakes material rendered in V-Ray. Image courtesy of Chaos Group	261
Fig. 8.60	Comparison of original versus denoised render using the V-Ray Denoiser. Image courtesy of Chaos Group	262
Fig. 8.61	Image rendered in Corona. © Gustavo Coutinho Alves	263
Fig. 8.62	Fashion design firms use ray tracing and CAD mesh to design perfectly fitting clothes. © 3D art-studio Pompidou	264
Fig. 8.63	Autodesk’s VRED supports ambient occlusions and baked shadows. <i>Source</i> Autodesk	265
Fig. 8.64	A frame from the War for the Planet of the Apes movie, rendered in Manuka. Image courtesy of Weta Digital, ©2017 Twentieth Century Fox Film Corporation. All rights reserved	270
Fig. 8.65	Astro, Pratik Solanki.	272
Fig. 8.66	Cycles is used as a production rendering engine for animation movies. Frame from “Agent 327, Operation Barbershop”— by Blender’s animation studio	273
Fig. 8.67	Blender was used for concept art in Jurassic World. <i>Source</i> Jama Jurabaev.	276
Fig. 8.68	Daz 3D’s Genesis 8 figure platforms produces photorealistic 3D composition results. <i>Source</i> Daz 3D	277
Fig. 8.69	Daz 3D’s renderer is well known for its realistic human and non-human figures. <i>Source</i> Daz 3D	278
Fig. 8.70	Dimension enables people with absolutely no 3D skills to create a composite shot of a 3D model within a 2D environment. <i>Source</i> Adobe	280
Fig. 8.71	(Left) Interactive view of the design mode canvas. (Middle) The Render Preview window. (Right) A final render produced from Render Mode. <i>Source</i> Adobe	281
Fig. 8.72	A demonstration of preview rendering. <i>Source</i> SideFx.	283
Fig. 8.73	A demonstration of Mini Render. <i>Source</i> SideFx	283
Fig. 8.74	A demonstration of sampling. <i>Source</i> SideFx	284
Fig. 8.75	A demonstration of volume modification. <i>Source</i> SideFx	285
Fig. 8.76	Houdini’s white-water simulator renders video-realistic waves that interact with rocks, sand, and one another. <i>Credit</i> Igor Zanic	285

Fig. 8.77 Unreal Studio has expanded its material library’s support.
Source Epic 289

Fig. 8.78 Jacketing and defeaturing in Unreal Studio. *Source Epic* 290

Fig. 8.79 Ray tracing a model in Visualize. *Source Dassault Systèmes*. 292

Fig. 8.80 Lighting makes all the difference in a rendering of a shiny product with no flat surfaces. 293

Fig. 8.81 Dassault’s 3DExperience UI logo 295

Fig. 8.82 PhotoView 360 example rendering. *Source SolidWorks*. 296

Fig. 8.83 Realistic ray-traced Corvette rendered using 3DEXCITE Deltagen. *Source Dassault Systèmes*. 297

Fig. 8.84 Heterogeneous media shading. *Source Corona Renderer*. 301

Fig. 8.85 Old “on surface” versus new “inside volume”.
Source Corona Renderer. 302

Fig. 8.86 Material Library Update. *Source Corona Renderer* 303

Fig. 8.87 Motion blur example. *Source Corona Renderer* 303

Fig. 8.88 A soap bubble rendered with spectral wavelength rendering.
Source Cebas 305

Fig. 8.89 Lux and Love by Charles Nandeya Ehouman (Sharlybg) using BBBB and LuxCoreRender. 308

Fig. 8.90 Using Maxwell renderer in SketchUp. *Source Next Limit* 309

Fig. 8.91 Sample picture for V-Ray tool integration for 3ds Max.
Source Chaos Group. 311

Fig. 8.92 Sample picture for V-Ray tool integration for SketchUp. *Source Chaos Group* 312

Fig. 8.93 Progressive rendering of the imperial crown of Austria. Model courtesy of Martin Lubich, <http://www.loramel.net> 315

Fig. 8.94 A Julia set drawn with Nvidia OptiX—this is a sample of the SDK. 316

Fig. 8.95 Nvidia’s OptiX block diagram 317

Fig. 8.96 Previous lightmapping solutions would take hours to compute even moderate-sized scenes. Expansive outdoor environments could take days. *Source AMD* 319

Fig. 8.97 Baking hardware performance comparison. *Source AMD*. 320

Fig. 8.98 Ray-tracing hardware performance comparison.
Source AMD. 321

Fig. 8.99 CL3VER real-time cloud rendering. 322

Fig. 8.100 How OneRender works. *Source OneRender* 323

Fig. 8.101 Before and after: a typical SketchUp architectural model before (above) and after (below) using Bloom Unit.
Source Migenius. 324

Fig. 8.102 Video of 64-byte ray caster. *Source Hellmood* 327

Fig. 8.103 Bare metal x86 ray tracer 328

Fig. 8.104 Meatball ray tracer 329

Fig. A.1 A young Hare but Albrecht Dürer, 1502.
Source Wikipedia 333

Fig. A.2 Rendering examples using a hare: **a** shadow casting, **b** ray casting, **c** Whitted ray tracing, and **d** Path Tracing.
Source Ray Tracing on Programmable Graphics Hardware (Purcell et al. 2002) 334

List of Tables

Table 5.1	Biased versus unbiased in different rendering schemes	67
Table 7.1	Lower-cost Turing spins leverage same proportion of Tensor and RT Cores	147
Table 8.1	Product comparison of RT 4 and Renderer 4 (<i>Source</i> Glare Technologies/Indigo).	198
Table 8.2	Cycle's features by processor type	274
Table 8.3	Software comparison	278
Table 8.4	A table of programs [y] with plug-ins [x]	299
Table A.1	Ray tracing plug-in programs	332
Table A.2	Technical papers on ray tracing published since 1982.	336
Table A.3	Books on ray tracing	337