Contents

Foreword .................................................. xxvii
Preface ................................................... xxix
Contributors ............................................. xxxiii

PART I  GEOMETRY

Chapter 1
Generating Complex Procedural Terrains Using the GPU........... 7
Ryan Geiss, NVIDIA Corporation

1.1 Introduction ......................................... 7
1.2 Marching Cubes and the Density Function ....................... 7
  1.2.1 Generating Polygons Within a Cell ................ 9
  1.2.2 Lookup Tables ................................ 11
1.3 An Overview of the Terrain Generation System ................. 12
  1.3.1 Generating the Polygons Within a Block of Terrain .... 13
  1.3.2 Generating the Density Values .................... 13
  1.3.3 Making an Interesting Density Function ............ 14
  1.3.4 Customizing the Terrain ......................... 18
1.4 Generating the Polygons Within a Block of Terrain .......... 20
  1.4.1 Margin Data .................................. 22
  1.4.2 Generating a Block: Method 1 .................... 23
  1.4.3 Generating a Block: Method 2 .................... 25
  1.4.4 Generating a Block: Method 3 .................... 26
1.5 Texturing and Shading ................................ 29
1.6 Considerations for Real-World Applications .................... 35
  1.6.1 Level of Detail ................................ 35
  1.6.2 Collisions and Lighting of Foreign Objects ......... 36
1.7 Conclusion ........................................... 37
1.8 References .......................................... 37
# Chapter 2

**Animated Crowd Rendering** ................................. 39

*Bryan Dudash, NVIDIA Corporation*

- 2.1 Motivation ................................................. 39
- 2.2 A Brief Review of Instancing ......................... 40
- 2.3 Details of the Technique .......................... 42
  - 2.3.1 Constants-Based Instancing .................. 43
  - 2.3.2 Palette Skinning Using an Animation Texture 44
  - 2.3.3 Geometry Variations .......................... 48
  - 2.3.4 The Level-of-Detail System ................... 49
- 2.4 Other Considerations ............................... 50
  - 2.4.1 Color Variations ............................... 50
  - 2.4.2 Performance ................................. 50
  - 2.4.3 Integration .................................. 51
- 2.5 Conclusion ........................................... 51
- 2.6 References ......................................... 52

# Chapter 3

**DirectX 10 Blend Shapes: Breaking the Limits** ............... 53

*Tristan Lorach, NVIDIA Corporation*

- 3.1 Introduction .......................................... 53
- 3.2 How Does It Work? ................................... 56
  - 3.2.1 Features of DirectX 10 ......................... 56
  - 3.2.2 Defining the Mesh ............................ 56
  - 3.2.3 The Stream-Out Method ....................... 57
  - 3.2.4 The Buffer-Template Method ................. 60
- 3.3 Running the Sample ............................... 66
- 3.4 Performance ......................................... 66
- 3.5 References ......................................... 67

# Chapter 4

**Next-Generation SpeedTree Rendering** ....................... 69

*Alexander Khramakov, NVIDIA Corporation*

*Iain Cantlay, NVIDIA Corporation*

*Yury Stepanenko, NVIDIA Corporation*

- 4.1 Introduction .......................................... 69
- 4.2 Silhouette Clipping .................................. 69
  - 4.2.1 Silhouette Fin Extrusion .................... 71
  - 4.2.2 Height Tracing ................................ 72
  - 4.2.3 Silhouette Level of Detail .................. 76
Chapter 5
Generic Adaptive Mesh Refinement .......................... 93
Tamy Boubekeur, LaBRI–INRIA, University of Bordeaux
Christophe Schlick, LaBRI–INRIA, University of Bordeaux
5.1 Introduction ............................................. 94
5.2 Overview .............................................. 95
5.3 Adaptive Refinement Patterns .............................. 96
  5.3.1 Implementation ..................................... 96
5.4 Rendering Workflow ....................................... 98
  5.4.1 Depth-Tagging ...................................... 98
  5.4.2 The CPU-Level Rendering Loop ..................... 99
  5.4.3 The GPU-Level Refinement Process ................. 99
5.5 Results .................................................. 100
5.6 Conclusion and Improvements .............................. 103
5.7 References ............................................. 104

Chapter 6
GPU-Generated Procedural Wind Animations for Trees ........ 105
Renaldas Zioma, Electronic Arts/Digital Illusions CE
6.1 Introduction ............................................. 105
6.2 Procedural Animations on the GPU ....................... 106
6.3 A Phenomenological Approach ............................ 106
  6.3.1 The Wind Field .................................... 106
  6.3.2 The Conceptual Structure of a Tree ................. 107
  6.3.3 The Two Categories of Simulation .................... 107
6.4 The Simulation Step ..................................... 113
  6.4.1 The Quaternion Library in HLSL .................... 115
8.3 Percentage-Closer Filtering .............................................. 159
  8.3.1 Problems with Percentage-Closer Filtering .......... 159
8.4 Variance Shadow Maps ............................................. 161
  8.4.1 Filtering the Variance Shadow Map ................. 162
  8.4.2 Biasing ......................................................... 164
  8.4.3 Light Bleeding ............................................... 166
  8.4.4 Numeric Stability ........................................... 169
  8.4.5 Implementation Notes ...................................... 171
  8.4.6 Variance Shadow Maps and Soft Shadows ......... 172
8.5 Summed-Area Variance Shadow Maps ....................... 174
  8.5.1 Generating Summed-Area Tables .................... 175
  8.5.2 Numeric Stability Revisited. ......................... 175
  8.5.3 Results .......................................................... 177
8.6 Percentage-Closer Soft Shadows .............................. 178
  8.6.1 The Blocker Search ........................................ 179
  8.6.2 Penumbra Size Estimation ............................. 180
  8.6.3 Shadow Filtering ............................................ 180
  8.6.4 Results .......................................................... 180
8.7 Conclusion .............................................................. 181
8.8 References .............................................................. 181

Chapter 9
Interactive Cinematic Relighting with Global Illumination ......... 183
Fabio Pellacini, Dartmouth College
Miloš Hašan, Cornell University
Kavita Bala, Cornell University

  9.1 Introduction ......................................................... 183
  9.2 An Overview of the Algorithm .............................. 184
  9.3 Gather Samples .................................................. 186
  9.4 One-Bounce Indirect Illumination ...................... 188
  9.5 Wavelets for Compression ................................. 189
  9.6 Adding Multiple Bounces .................................... 192
  9.7 Packing Sparse Matrix Data ................................. 193
  9.8 A GPU-Based Relighting Engine ......................... 195
    9.8.1 Direct Illumination ..................................... 196
    9.8.2 Wavelet Transform ..................................... 197
    9.8.3 Sparse Matrix Multiplication ....................... 198
  9.9 Results .............................................................. 200
  9.10 Conclusion ........................................................ 201
  9.11 References ........................................................ 201
Chapter 10
Parallel-Split Shadow Maps on Programmable GPUs ............. 203
Fan Zhang, The Chinese University of Hong Kong
Hanqiu Sun, The Chinese University of Hong Kong
Oskari Nyman, Helsinki University of Technology

10.1 Introduction ............................................. 203
10.2 The Algorithm ........................................... 205
  10.2.1 Step 1: Splitting the View Frustum ..................... 206
  10.2.2 Step 2: Calculating Light’s Transformation Matrices .... 209
  10.2.3 Steps 3 and 4: Generating PSSMs and Synthesizing
    Shadows .................................. 214
10.3 Hardware-Specific Implementations ............................ 214
  10.3.1 The Multipass Method ................................ 215
  10.3.2 DirectX 9-Level Acceleration ......................... 217
  10.3.3 DirectX 10-Level Acceleration ......................... 220
10.4 Further Optimizations ..................................... 232
10.5 Results ............................................... 233
10.6 Conclusion ............................................. 233
10.7 References ............................................. 235

Chapter 11
Efficient and Robust Shadow Volumes Using Hierarchical
Occlusion Culling and Geometry Shaders ......................... 239
Martin Stich, mental images
Carsten Wächter, Ulm University
Alexander Keller, Ulm University

11.1 Introduction ............................................. 239
11.2 An Overview of Shadow Volumes ............................ 240
  11.2.1 Z-Pass and Z-Fail .................................. 240
  11.2.2 Volume Generation ............................... 242
  11.2.3 Performance and Optimizations ..................... 243
11.3 Our Implementation ..................................... 244
  11.3.1 Robust Shadows for Low-Quality Meshes .............. 244
  11.3.2 Dynamic Volume Generation with Geometry Shaders .. 246
  11.3.3 Improving Performance with Hierarchical Occlusion
    Culling .................................. 252
11.4 Conclusion ............................................. 254
11.5 References ............................................. 254
### Chapter 12
#### High-Quality Ambient Occlusion ................................................. 257

*Jared Hoberock, University of Illinois at Urbana-Champaign*

*Yuntao Jia, University of Illinois at Urbana-Champaign*

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1</td>
<td>257</td>
</tr>
<tr>
<td>12.2</td>
<td>258</td>
</tr>
<tr>
<td>12.2.1</td>
<td></td>
</tr>
<tr>
<td>12.2.2</td>
<td>261</td>
</tr>
<tr>
<td>12.3</td>
<td>261</td>
</tr>
<tr>
<td>12.3.1</td>
<td></td>
</tr>
<tr>
<td>12.3.2</td>
<td>263</td>
</tr>
<tr>
<td>12.4</td>
<td>267</td>
</tr>
<tr>
<td>12.5</td>
<td>269</td>
</tr>
<tr>
<td>12.6</td>
<td>270</td>
</tr>
<tr>
<td>12.6.1</td>
<td></td>
</tr>
<tr>
<td>12.6.2</td>
<td>271</td>
</tr>
<tr>
<td>12.7</td>
<td>273</td>
</tr>
<tr>
<td>12.8</td>
<td>274</td>
</tr>
</tbody>
</table>

### Chapter 13
#### Volumetric Light Scattering as a Post-Process ............................... 275

*Kenny Mitchell, Electronic Arts*

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.1</td>
<td>275</td>
</tr>
<tr>
<td>13.2</td>
<td>276</td>
</tr>
<tr>
<td>13.3</td>
<td>277</td>
</tr>
<tr>
<td>13.3.1</td>
<td></td>
</tr>
<tr>
<td>13.4</td>
<td>279</td>
</tr>
<tr>
<td>13.5</td>
<td>281</td>
</tr>
<tr>
<td>13.5.1</td>
<td></td>
</tr>
<tr>
<td>13.5.2</td>
<td>282</td>
</tr>
<tr>
<td>13.5.3</td>
<td>282</td>
</tr>
<tr>
<td>13.6</td>
<td>282</td>
</tr>
<tr>
<td>13.7</td>
<td>283</td>
</tr>
<tr>
<td>13.8</td>
<td>284</td>
</tr>
<tr>
<td>13.9</td>
<td>284</td>
</tr>
<tr>
<td>13.10</td>
<td>284</td>
</tr>
</tbody>
</table>
PART III  RENDERING

Chapter 14
Advanced Techniques for Realistic Real-Time Skin Rendering  . . . . 293
Eugene d'Eon, NVIDIA Corporation
David Luebke, NVIDIA Corporation

14.1 The Appearance of Skin ........................................... 293
  14.1.1 Skin Surface Reflectance ................................. 295
  14.1.2 Skin Subsurface Reflectance ............................. 296
14.2 An Overview of the Skin-Rendering System .................. 297
14.3 Specular Surface Reflectance ................................. 299
  14.3.1 Implementing a Physically Based Specular Reflectance Model for Skin .................................................. 300
14.4 Scattering Theory .................................................. 305
  14.4.1 Diffusion Profiles ........................................... 305
  14.4.2 Rendering with Diffusion Profiles ......................... 306
  14.4.3 The Shape of Diffusion Profiles .......................... 307
  14.4.4 A Sum-of-Gaussians Diffusion Profile ................. 308
  14.4.5 Fitting Predicted or Measured Profiles ................ 311
  14.4.6 Plotting Diffusion Profiles ............................... 312
  14.4.7 A Sum-of-Gaussians Fit for Skin ...................... 312
14.5 Advanced Subsurface Scattering ............................ 314
  14.5.1 Texture-Space Diffusion ................................. 314
  14.5.2 Improved Texture-Space Diffusion ....................... 316
  14.5.3 Modified Translucent Shadow Maps ............... 336
14.6 A Fast Bloom Filter ............................................ 342
14.7 Conclusion ....................................................... 342
  14.7.1 Future Work .............................................. 343
14.8 References ....................................................... 345

Chapter 15
Playable Universal Capture ........................................ 349
George Borshukov, Electronic Arts
Jefferson Montgomery, Electronic Arts
John Hable, Electronic Arts

15.1 Introduction ...................................................... 349
15.2 The Data Acquisition Pipeline ................................ 350
Chapter 15
15.3 Compression and Decompression of the Animated Textures
  15.3.1 Principal Component Analysis
  15.3.2 Compression
  15.3.3 Decompression
  15.3.4 Variable PCA
  15.3.5 Practical Considerations
  15.4 Sequencing Performances
  15.5 Conclusion
  15.6 References

Chapter 16
Vegetation Procedural Animation and Shading in Crysis
Tiago Sousa, Crytek
  16.1 Procedural Animation
    16.1.1 Implementation Details
  16.2 Vegetation Shading
    16.2.1 Ambient Lighting
    16.2.2 Edge Smoothing
    16.2.3 Putting It All Together
    16.2.4 Implementation Details
  16.3 Conclusion
  16.4 References

Chapter 17
Robust Multiple Specular Reflections and Refractions
Tamás Umenhoffer, Budapest University of Technology and Economics
Gustavo Patow, University of Girona
László Szirmay-Kalos, Budapest University of Technology and Economics
  17.1 Introduction
  17.2 Tracing Secondary Rays
    17.2.1 Generation of Layered Distance Maps
    17.2.2 Ray Tracing Layered Distance Maps
  17.3 Reflections and Refractions
  17.4 Results
  17.5 Conclusion
  17.6 References
Chapter 22
Baking Normal Maps on the GPU ........................................... 491
Diogo Teixeira, Move Interactive

22.1 The Traditional Implementation .................................... 492
   22.1.1 Projection ......................................................... 492
   22.1.2 The Boundary Cage ........................................... 492
22.2 Acceleration Structures .............................................. 493
   22.2.1 The Uniform Grid ............................................. 494
   22.2.2 The 3D Digital Differential Analyzer ....................... 495
22.3 Feeding the GPU .................................................... 496
   22.3.1 Indexing Limitations ........................................ 497
   22.3.2 Memory and Architectural Limitations .................... 498
22.4 Implementation ..................................................... 498
   22.4.1 Setup and Preprocessing ................................... 499
   22.4.2 The Single-Pass Implementation .......................... 501
   22.4.3 The Multipass Implementation ............................. 507
   22.4.4 Antialiasing ................................................... 508
22.5 Results ............................................................. 508
22.6 Conclusion ......................................................... 511
22.7 References .......................................................... 511

Chapter 23
High-Speed, Off-Screen Particles ......................................... 513
Iain Cantlay, NVIDIA Corporation

23.1 Motivation .......................................................... 513
23.2 Off-Screen Rendering ............................................... 514
   23.2.1 Off-Screen Depth Testing ................................ 514
   23.2.2 Acquiring Depth ............................................. 515
23.3 Downsampling Depth ............................................... 517
   23.3.1 Point Sampling Depth ...................................... 517
   23.3.2 Maximum of Depth Samples .............................. 519
23.4 Depth Testing and Soft Particles ................................. 519
23.5 Alpha Blending ..................................................... 520
23.6 Mixed-Resolution Rendering ..................................... 522
   23.6.1 Edge Detection .............................................. 522
   23.6.2 Composing with Stenciling ............................... 523
23.7 Results ............................................................. 525
   23.7.1 Image Quality ................................................. 525
   23.7.2 Performance .................................................. 526
23.8 Conclusion .......................................................... 527
23.9 References .......................................................... 528
Chapter 24
The Importance of Being Linear ............................................. 529
Larry Gritz, NVIDIA Corporation
Eugene d’Eon, NVIDIA Corporation

24.1 Introduction ......................................................... 529
24.2 Light, Displays, and Color Spaces ................................. 529
  24.2.1 Problems with Digital Image Capture, Creation,
  and Display ....................................................... 529
  24.2.2 Digression: What Is Linear? ............................... 530
  24.2.3 Monitors Are Nonlinear, Renderers Are Linear .......... 531
24.3 The Symptoms ...................................................... 533
  24.3.1 Nonlinear Input Textures .................................. 533
  24.3.2 Mipmaps ..................................................... 533
  24.3.3 Illumination .................................................. 534
  24.3.4 Two Wrongs Don’t Make a Right ....................... 535
24.4 The Cure ............................................................ 538
  24.4.1 Input Images (Scans, Paintings, and Digital Photos) ..... 539
  24.4.2 Output Images (Final Renders) ............................. 540
  24.4.3 Intermediate Color Buffers ............................... 541
24.5 Conclusion .......................................................... 541
24.6 Further Reading ..................................................... 542

Chapter 25
Rendering Vector Art on the GPU ................................. 543
Charles Loop, Microsoft Research
Jim Blinn, Microsoft Research

25.1 Introduction ......................................................... 543
25.2 Quadratic Splines .................................................. 544
25.3 Cubic Splines ....................................................... 546
  25.3.1 Serpentine ................................................... 552
  25.3.2 Loop ......................................................... 553
  25.3.3 Cusp ......................................................... 554
  25.3.4 Quadratic .................................................... 555
25.4 Triangulation ......................................................... 555
25.5 Antialiasing ........................................................ 556
25.6 Code .............................................................. 558
25.7 Conclusion ........................................................ 559
25.8 References ........................................................ 560

Copyright NVIDIA Corporation. All rights reserved.
Chapter 30
Real-Time Simulation and Rendering of 3D Fluids ..................... 633
Keenan Crane, University of Illinois at Urbana-Champaign
Ignacio Llamas, NVIDIA Corporation
Sarah Tariq, NVIDIA Corporation

30.1 Introduction .................................................. 633
30.2 Simulation .................................................... 634
  30.2.1 Background ............................................. 634
  30.2.2 Equations of Fluid Motion ............................. 635
  30.2.3 Solving for Velocity ................................. 636
  30.2.4 Solid-Fluid Interaction ............................... 642
  30.2.5 Smoke ................................................ 658
  30.2.6 Fire .................................................. 659
  30.2.7 Water ................................................ 659
  30.2.8 Performance Considerations ......................... 660
  30.2.9 Storage ............................................... 661
  30.2.10 Numerical Issues ................................. 662
30.3 Rendering ................................................... 665
  30.3.1 Volume Rendering ..................................... 665
  30.3.2 Rendering Liquids .................................... 671
30.4 Conclusion .................................................. 672
30.5 References .................................................. 673

Chapter 31
Fast N-Body Simulation with CUDA ................................. 677
Lars Nyland, NVIDIA Corporation
Mark Harris, NVIDIA Corporation
Jan Prins, University of North Carolina at Chapel Hill

31.1 Introduction .................................................. 677
31.2 All-Pairs N-Body Simulation ............................... 679
31.3 A CUDA Implementation of the All-Pairs N-Body Algorithm 680
  31.3.1 Body-Body Force Calculation ....................... 681
  31.3.2 Tile Calculation ..................................... 682
  31.3.3 Clustering Tiles into Thread Blocks ................ 683
  31.3.4 Defining a Grid of Thread Blocks ................... 685
Chapter 34
Signed Distance Fields Using Single-Pass GPU Scan
Conversion of Tetrahedra ........................................ 741

Kenny Erleben, University of Copenhagen
Henrik Dohlmann, 3Dfacto R&D

34.1 Introduction .................................................. 741
34.1.1 Overview of Signed Distance Fields ................. 741
34.1.2 Overview of Our Method .......................... 742
34.2 Leaking Artifacts in Scan Methods .................... 742
34.2.1 The Plane Test ....................................... 743
34.2.2 How a Bounding Volume Is Constructed ............ 744
34.2.3 Folds in the Polygonal Model ...................... 746
34.3 Our Tetrahedra GPU Scan Method .................... 747
34.3.1 Computing the Shell ................................. 749
34.3.2 Computing the Cross Section of a Tetrahedron .... 750
34.3.3 Computing Signed Distance Using Angle-Weighted Pseudonormals ......................... 751
34.4 Results ..................................................... 756
34.5 Conclusion .................................................. 758
34.6 Future Work ................................................ 759
34.6.1 Improvements to the Algorithm .................... 759
34.6.2 Improvements to Our Implementation .......... 760
34.7 Further Reading ........................................... 760
34.8 References ................................................ 762
## Chapter 37
Efficient Random Number Generation and Application Using CUDA

Lee Howes, Imperial College London
David Thomas, Imperial College London

### 37.1 Monte Carlo Simulations

### 37.2 Random Number Generators

- 37.2.1 Introduction
- 37.2.2 Uniform-to-Gaussian Conversion Generator
- 37.2.3 Types of Gaussian Transforms
- 37.2.4 The Wallace Gaussian Generator
- 37.2.5 Integrating the Wallace Gaussian Generator into a Simulation

### 37.3 Example Applications

- 37.3.1 Asian Option
- 37.3.2 Variant on a Lookback Option
- 37.3.3 Results

### 37.4 Conclusion

### 37.5 References

## Chapter 38
Imaging Earth’s Subsurface Using CUDA

Bernard Deschizeaux, CGGVeritas
Jean-Yves Blanc, CGGVeritas

### 38.1 Introduction

### 38.2 Seismic Data

### 38.3 Seismic Processing

- 38.3.1 Wave Propagation
- 38.3.2 Seismic Migration Using the SRMIP Algorithm

### 38.4 The GPU Implementation

- 38.4.1 GPU/CPU Communication
- 38.4.2 The CUDA Implementation
- 38.4.3 The Wave Propagation Kernel

### 38.5 Performance

### 38.6 Conclusion

### 38.7 References
# Chapter 39
Parallel Prefix Sum (Scan) with CUDA

Mark Harris, NVIDIA Corporation  
Shubhabrata Sengupta, University of California, Davis  
John D. Owens, University of California, Davis

## 39.1 Introduction

### 39.1.1 Sequential Scan and Work Efficiency

### 39.2 Implementation

#### 39.2.1 A Naive Parallel Scan

#### 39.2.2 A Work-Efficient Parallel Scan

#### 39.2.3 Avoiding Bank Conflicts

#### 39.2.4 Arrays of Arbitrary Size

#### 39.2.5 Further Optimization and Performance Results

#### 39.2.6 The Advantages of CUDA over the OpenGL Implementation

## 39.3 Applications of Scan

#### 39.3.1 Stream Compaction

#### 39.3.2 Summed-Area Tables

#### 39.3.3 Radix Sort

#### 39.3.4 Previous Work

## 39.4 Conclusion

## 39.5 References

---

# Chapter 40
Incremental Computation of the Gaussian

Ken Turkowski, Adobe Systems

## 40.1 Introduction and Related Work

## 40.2 Polynomial Forward Differencing

## 40.3 The Incremental Gaussian Algorithm

## 40.4 Error Analysis

## 40.5 Performance

## 40.6 Conclusion

## 40.7 References
Composition, the organization of elemental operations into a nonobvious whole, is the essence of imperative programming. The instruction set architecture (ISA) of a microprocessor is a versatile composition interface, which programmers of software renderers have used effectively and creatively in their quest for image realism. Early graphics hardware increased rendering performance, but often at a high cost in composability, and thus in programmability and application innovation. Hardware with microprocessor-like programmability did evolve (for example, the Ikonas Graphics System), but the dominant form of graphics hardware acceleration has been organized around a fixed sequence of rendering operations, often referred to as the graphics pipeline. Early interfaces to these systems—such as CORE and later, PHIGS—allowed programmers to specify rendering results, but they were not designed for composition.

OpenGL, which I helped to evolve from its Silicon Graphics-defined predecessor IRIS GL in the early 1990s, addressed the need for composability by specifying an architecture (informally called the OpenGL Machine) that was accessed through an imperative programmatic interface. Many features—for example, tightly specified semantics; table-driven operations such as stencil and depth-buffer functions; texture mapping exposed as a general 1D, 2D, and 3D lookup function; and required repeatability properties—ensured that programmers could compose OpenGL operations with powerful and reliable results. Some of the useful techniques that OpenGL enabled include texture-based volume rendering, shadow volumes using stencil buffers, and constructive solid geometry algorithms such as capping (the computation of surface planes at the intersections of clipping planes and solid objects defined by polygons). Ultimately, Mark Peercy and the coauthors of the SIGGRAPH 2000 paper “Interactive Multi-Pass Programmable Shading” demonstrated that arbitrary RenderMan shaders could be accelerated through the composition of OpenGL rendering operations.

During this decade, increases in the raw capability of integrated circuit technology allowed the OpenGL architecture (and later, Direct3D) to be extended to expose an
ISA interface. These extensions appeared as programmable vertex and fragment shaders within the graphics pipeline and now, with the introduction of CUDA, as a data-parallel ISA in near parity with that of the microprocessor. Although the cycle toward complete microprocessor-like versatility is not complete, the tremendous power of graphics hardware acceleration is more accessible than ever to programmers.

And what computational power it is! At this writing, the NVIDIA GeForce 8800 Ultra performs over 400 billion floating-point operations per second—more than the most powerful supercomputer available a decade ago, and five times more than today's most powerful microprocessor. The data-parallel programming model the Ultra supports allows its computational power to be harnessed without concern for the number of processors employed. This is critical, because while today's Ultra already includes over 100 processors, tomorrow's will include thousands, and then more. With no end in sight to the annual compounding of integrated circuit density known as Moore's Law, massively parallel systems are clearly the future of computing, with graphics hardware leading the way.

*GPU Gems 3* is a collection of state-of-the-art GPU programming examples. It is about putting data-parallel processing to work. The first four sections focus on graphics-specific applications of GPUs in the areas of geometry, lighting and shadows, rendering, and image effects. Topics in the fifth and sixth sections broaden the scope by providing concrete examples of nongraphical applications that can now be addressed with data-parallel GPU technology. These applications are diverse, ranging from rigid-body simulation to fluid flow simulation, from virus signature matching to encryption and decryption, and from random number generation to computation of the Gaussian.

Where is this all leading? The cover art reminds us that the mind remains the most capable parallel computing system of all. A long-term goal of computer science is to achieve and, ultimately, to surpass the capabilities of the human mind. It's exciting to think that the computer graphics community, as we identify, address, and master the challenges of massively parallel computing, is contributing to the realization of this dream.

*Kurt Akeley*

*Microsoft Research*
Preface

It has been only three years since the first *GPU Gems* book was introduced, and some areas of real-time graphics have truly become ultrarealistic. Chapter 14, “Advanced Techniques for Realistic Real-Time Skin Rendering,” illustrates this evolution beautifully, describing a skin rendering technique that works so well that the data acquisition and animation will become the most challenging problem in rendering human characters for the next couple of years.

All this progress has been fueled by a sustained rhythm of GPU innovation. These processing units continue to become faster and more flexible in their use. Today’s GPUs can process enormous amounts of data and are used not only for rendering 3D scenes, but also for processing images or performing massively parallel computing, such as financial statistics or terrain analysis for finding new oil fields.

Whether they are used for computing or graphics, GPUs need a software interface to drive them, and we are in the midst of an important transition. The new generation of APIs brings additional orthogonality and exposes new capabilities such as generating geometry programmatically. On the computing side, the CUDA architecture lets developers use a C-like language to perform computing tasks rather than forcing the programmer to use the graphics pipeline. This architecture will allow developers without a graphics background to tap into the immense potential of the GPU.

More than 200 chapters were submitted by the GPU programming community, covering a large spectrum of GPU usage ranging from pure 3D rendering to nongraphics applications. Each of them went through a rigorous review process conducted both by NVIDIA’s engineers and by external reviewers.

We were able to include 41 chapters, each of which went through another review, during which feedback from the editors and peer reviewers often significantly improved the content. Unfortunately, we could not include some excellent chapters, simply due to the space restriction of the book. It was difficult to establish the final table of contents, but we would like to thank everyone who sent a submission.
Intended Audience

For the graphics-related chapters, we expect the reader to be familiar with the fundamentals of computer graphics including graphics APIs such as DirectX and OpenGL, as well as their associated high-level programming languages, namely HLSL, GLSL, or Cg. Anyone working with interactive 3D applications will find in this book a wealth of applicable techniques for today’s and tomorrow’s GPUs.

Readers interested in computing and CUDA will find it best to know parallel computing concepts. C programming knowledge is also expected.

Trying the Code Samples

GPU Gems 3 comes with a disc that includes samples, movies, and other demonstrations of the techniques described in this book. You can also go to the book’s Web page to find the latest updates and supplemental materials: developer.nvidia.com/gpugems3.

Acknowledgments

This book represents the dedication of many people—especially the numerous authors who submitted their most recent work to the GPU community by contributing to this book. Without a doubt, these inspirational and powerful chapters will help thousands of developers push the envelope in their applications.

Our section editors—Cyril Zeller, Evan Hart, Ignacio Castaño Aguado, Kevin Bjorke, Kevin Myers, and Nolan Goodnight—took on an invaluable role, providing authors with feedback and guidance to make the chapters as good as they could be. Without their expertise and contributions above and beyond their usual workload, this book could not have been published.

Ensuring the clarity of GPU Gems 3 required numerous diagrams, illustrations, and screen shots. A lot of diligence went into unifying the graphic style of about 500 figures, and we thank Michael Fornalski and Jim Reed for their wonderful work on these. We are grateful to Huey Nguyen and his team for their support for many of our projects. We also thank Rory Loeb for his contribution to the amazing book cover design and many other graphic elements of the book.

We would also like to thank Catherine Kilkenny and Teresa Saffaie for tremendous help with copyediting as chapters were being worked on.
Randy Fernando, the editor of the previous *GPU Gems* books, shared his wealth of experience acquired in producing those volumes.

We are grateful to Kurt Akeley for writing our insightful and forward-looking foreword.

At Addison-Wesley, Peter Gordon, John Fuller, and Kim Boedigheimer managed this project to completion before handing the marketing aspect to Curt Johnson. Christopher Keane did fantastic work on the copyediting and typesetting.

The support from many executive staff members from NVIDIA was critical to this endeavor: Tony Tamasi and Dan Vivoli continually value the creation of educational material and provided the resources necessary to accomplish this project.

We are grateful to Jen-Hsun Huang for his continued support of the *GPU Gems* series and for creating an environment that encourages innovation and teamwork.

We also thank everyone at NVIDIA for their support and for continually building the technology that changes the way people think about computing.

*Hubert Nguyen*

*NVIDIA Corporation*