

# Contents

Foreword .....	xxix
Preface .....	xxxi
Contributors .....	xxxv

## **PART I GEOMETRIC COMPLEXITY**

---

**1**

### **Chapter 1**

#### **Toward Photorealism in Virtual Botany .....** 7

*David Whitley, Simutronics Corporation*

1.1 Scene Management .....	7
1.1.1 The Planting Grid .....	8
1.1.2 Planting Strategy .....	9
1.1.3 Real-Time Optimization .....	10
1.2 The Grass Layer .....	11
1.2.1 Simulating Alpha Transparency via Dissolve .....	13
1.2.2 Variation .....	15
1.2.3 Lighting .....	15
1.2.4 Wind .....	17
1.3 The Ground Clutter Layer .....	17
1.4 The Tree and Shrub Layers .....	18
1.5 Shadowing .....	20
1.6 Post-Processing .....	22
1.6.1 Sky Dome Blooming .....	23
1.6.2 Full-Scene Glow .....	24
1.7 Conclusion .....	24
1.8 References .....	24

## Chapter 2

### Terrain Rendering Using GPU-Based Geometry Clipmaps . . . . . 27

*Arul Asirvatham, Microsoft Research*

*Hugues Hoppe, Microsoft Research*

2.1	Review of Geometry Clipmaps . . . . .	27
2.2	Overview of GPU Implementation . . . . .	30
2.2.1	Data Structures . . . . .	31
2.2.2	Clipmap Size . . . . .	31
2.3	Rendering . . . . .	32
2.3.1	Active Levels . . . . .	32
2.3.2	Vertex and Index Buffers . . . . .	33
2.3.3	View Frustum Culling . . . . .	35
2.3.4	DrawPrimitive Calls . . . . .	35
2.3.5	The Vertex Shader . . . . .	36
2.3.6	The Pixel Shader . . . . .	38
2.4	Update . . . . .	39
2.4.1	Upsampling . . . . .	40
2.4.2	Residuals . . . . .	42
2.4.3	Normal Map . . . . .	42
2.5	Results and Discussion . . . . .	43
2.6	Summary and Improvements . . . . .	43
2.6.1	Vertex Textures . . . . .	44
2.6.2	Eliminating Normal Maps . . . . .	44
2.6.3	Memory-Free Terrain Synthesis . . . . .	44
2.7	References . . . . .	44

## Chapter 3

### Inside Geometry Instancing . . . . . 47

*Francesco Carucci, Lionhead Studios*

3.1	Why Geometry Instancing? . . . . .	48
3.2	Definitions . . . . .	49
3.2.1	Geometry Packet . . . . .	49
3.2.2	Instance Attributes . . . . .	49
3.2.3	Geometry Instance . . . . .	50
3.2.4	Render and Texture Context . . . . .	50
3.2.5	Geometry Batch . . . . .	50

3.3 Implementation . . . . .	53
3.3.1 Static Batching . . . . .	54
3.3.2 Dynamic Batching . . . . .	56
3.3.3 Vertex Constants Instancing . . . . .	57
3.3.4 Batching with the Geometry Instancing API . . . . .	61
3.4 Conclusion . . . . .	65
3.5 References . . . . .	67

## Chapter 4

### Segment Buffering . . . . . 69

*Jon Olick, 2015*

4.1 The Problem Space . . . . .	69
4.2 The Solution . . . . .	70
4.3 The Method . . . . .	71
4.3.1 Segment Buffering, Step 1 . . . . .	71
4.3.2 Segment Buffering, Step 2 . . . . .	71
4.3.3 Segment Buffering, Step 3 . . . . .	72
4.4 Improving the Technique . . . . .	72
4.5 Conclusion . . . . .	72
4.6 References . . . . .	73

## Chapter 5

### Optimizing Resource Management with Multistreaming . . . . . 75

*Oliver Hoeller, Piranha Bytes*

*Kurt Pelzer, Piranha Bytes*

5.1 Overview . . . . .	76
5.2 Implementation . . . . .	77
5.2.1 Multistreaming with DirectX 9.0 . . . . .	78
5.2.2 Resource Management . . . . .	81
5.2.3 Processing Vertices . . . . .	83
5.3 Conclusion . . . . .	89
5.4 References . . . . .	90

## Chapter 6

### Hardware Occlusion Queries Made Useful . . . . . 91

*Michael Wimmer, Vienna University of Technology*

*Jiří Bittner, Vienna University of Technology*

6.1 Introduction . . . . .	91
6.2 For Which Scenes Are Occlusion Queries Effective? . . . . .	92

6.3	What Is Occlusion Culling? . . . . .	93
6.4	Hierarchical Stop-and-Wait Method . . . . .	94
6.4.1	The Naive Algorithm, or Why Use Hierarchies at All? . . . . .	94
6.4.2	Hierarchies to the Rescue! . . . . .	95
6.4.3	Hierarchical Algorithm . . . . .	95
6.4.4	Problem 1: Stalls . . . . .	96
6.4.5	Problem 2: Query Overhead . . . . .	97
6.5	Coherent Hierarchical Culling. . . . .	97
6.5.1	Idea 1: Being Smart and Guessing . . . . .	97
6.5.2	Idea 2: Pull Up, Pull Up . . . . .	98
6.5.3	Algorithm . . . . .	99
6.5.4	Implementation Details . . . . .	100
6.5.5	Why Are There Fewer Stalls? . . . . .	103
6.5.6	Why Are There Fewer Queries? . . . . .	104
6.5.7	How to Traverse the Hierarchy. . . . .	104
6.6	Optimizations . . . . .	105
6.6.1	Querying with Actual Geometry . . . . .	105
6.6.2	Z-Only Rendering Pass . . . . .	105
6.6.3	Approximate Visibility . . . . .	105
6.6.4	Conservative Visibility Testing. . . . .	106
6.7	Conclusion . . . . .	106
6.8	References . . . . .	108

## Chapter 7

### Adaptive Tessellation of Subdivision Surfaces with

### Displacement Mapping . . . . . 109

*Michael Bunnell, NVIDIA Corporation*

7.1	Subdivision Surfaces . . . . .	109
7.1.1	Some Definitions . . . . .	110
7.1.2	Catmull-Clark Subdivision. . . . .	110
7.1.3	Using Subdivision for Tessellation. . . . .	111
7.1.4	Patching the Surface. . . . .	112
7.1.5	The GPU Tessellation Algorithm . . . . .	114
7.1.6	Watertight Tessellation . . . . .	118
7.2	Displacement Mapping . . . . .	119
7.2.1	Changing the Flatness Test. . . . .	120
7.2.2	Shading Using Normal Mapping. . . . .	120
7.3	Conclusion . . . . .	122
7.4	References . . . . .	122

## Chapter 8

### Per-Pixel Displacement Mapping with Distance Functions . . . . . 123

*William Donnelly, University of Waterloo*

8.1 Introduction . . . . .	123
8.2 Previous Work . . . . .	125
8.3 The Distance-Mapping Algorithm . . . . .	126
8.3.1 Arbitrary Meshes . . . . .	129
8.4 Computing the Distance Map . . . . .	130
8.5 The Shaders . . . . .	130
8.5.1 The Vertex Shader . . . . .	130
8.5.2 The Fragment Shader . . . . .	130
8.5.3 A Note on Filtering . . . . .	132
8.6 Results . . . . .	132
8.7 Conclusion . . . . .	134
8.8 References . . . . .	135

## **PART II SHADING, LIGHTING, AND SHADOWS**

**137**

## Chapter 9

### Deferred Shading in *S.T.A.L.K.E.R.* . . . . . 143

*Oles Shishkovtsov, GSC Game World*

9.1 Introduction . . . . .	143
9.2 The Myths . . . . .	145
9.3 Optimizations . . . . .	147
9.3.1 What to Optimize . . . . .	147
9.3.2 Lighting Optimizations . . . . .	148
9.3.3 G-Buffer-Creation Optimizations . . . . .	151
9.3.4 Shadowing Optimizations . . . . .	153
9.4 Improving Quality . . . . .	154
9.4.1 The Power of “Virtual Position” . . . . .	155
9.4.2 Ambient Occlusion . . . . .	156
9.4.3 Materials and Surface-Light Interaction . . . . .	157
9.5 Antialiasing . . . . .	158
9.5.1 Efficient Tone Mapping . . . . .	161
9.5.2 Dealing with Transparency . . . . .	162
9.6 Things We Tried but Did Not Include in the Final Code . . . . .	162
9.6.1 Elevation Maps . . . . .	163
9.6.2 Real-Time Global Illumination . . . . .	163
9.7 Conclusion . . . . .	164
9.8 References . . . . .	165

## Chapter 10

### Real-Time Computation of Dynamic Irradiance

#### Environment Maps. . . . . 167

*Gary King, NVIDIA Corporation*

10.1 Irradiance Environment Maps . . . . .	167
10.2 Spherical Harmonic Convolution . . . . .	170
10.3 Mapping to the GPU. . . . .	172
10.3.1 Spatial to Frequency Domain . . . . .	172
10.3.2 Convolution and Back Again . . . . .	173
10.4 Further Work. . . . .	175
10.5 Conclusion . . . . .	176
10.6 References . . . . .	176

## Chapter 11

### Approximate Bidirectional Texture Functions . . . . . 177

*Jan Kautz, Massachusetts Institute of Technology*

11.1 Introduction . . . . .	177
11.2 Acquisition . . . . .	179
11.2.1 Setup and Acquisition . . . . .	179
11.2.2 Assembling the Shading Map . . . . .	179
11.3 Rendering . . . . .	181
11.3.1 Detailed Algorithm . . . . .	181
11.3.2 Real-Time Rendering. . . . .	182
11.4 Results. . . . .	184
11.4.1 Discussion . . . . .	186
11.5 Conclusion . . . . .	187
11.6 References . . . . .	187

## Chapter 12

### Tile-Based Texture Mapping. . . . . 189

*Li-Yi Wei, NVIDIA Corporation*

12.1 Our Approach . . . . .	191
12.2 Texture Tile Construction . . . . .	191
12.3 Texture Tile Packing . . . . .	192
12.4 Texture Tile Mapping . . . . .	195
12.5 Mipmap Issues. . . . .	197
12.6 Conclusion . . . . .	198
12.7 References . . . . .	199

## Chapter 13

### Implementing the mental images Phenomena Renderer on the GPU . . . . . 201

*Martin-Karl Lefrançois, mental images*

13.1	Introduction . . . . .	201
13.2	Shaders and Phenomena . . . . .	202
13.3	Implementing Phenomena Using Cg . . . . .	205
13.3.1	The Cg Vertex Program and the Varying Parameters . . . . .	205
13.3.2	The <code>main()</code> Entry Point for Fragment Shaders . . . . .	207
13.3.3	The General Shader Interfaces . . . . .	207
13.3.4	Example of a Simple Shader . . . . .	208
13.3.5	Global State Variables . . . . .	211
13.3.6	Light Shaders . . . . .	211
13.3.7	Texture Shaders . . . . .	215
13.3.8	Bump Mapping . . . . .	216
13.3.9	Environment and Volume Shaders . . . . .	217
13.3.10	Shaders Returning Structures . . . . .	218
13.3.11	Rendering Hair . . . . .	220
13.3.12	Putting It All Together . . . . .	220
13.4	Conclusion . . . . .	221
13.5	References . . . . .	222

## Chapter 14

### Dynamic Ambient Occlusion and Indirect Lighting . . . . . 223

*Michael Bunnell, NVIDIA Corporation*

14.1	Surface Elements . . . . .	223
14.2	Ambient Occlusion . . . . .	225
14.2.1	The Multipass Shadowing Algorithm . . . . .	226
14.2.2	Improving Performance . . . . .	228
14.3	Indirect Lighting and Area Lights . . . . .	231
14.4	Conclusion . . . . .	232
14.5	References . . . . .	233

## Chapter 15

### Blueprint Rendering and “Sketchy Drawings” . . . . . 235

*Marc Nienhaus, University of Potsdam, Hasso-Plattner-Institute*

*Jürgen Döllner, University of Potsdam, Hasso-Plattner-Institute*

15.1	Basic Principles . . . . .	236
15.1.1	Intermediate Rendering Results . . . . .	236
15.1.2	Edge Enhancement . . . . .	236
15.1.3	Depth Sprite Rendering . . . . .	237
15.2	Blueprint Rendering . . . . .	238
15.2.1	Depth Peeling . . . . .	238
15.2.2	Extracting Visible and Nonvisible Edges . . . . .	241
15.2.3	Composing Blueprints . . . . .	241
15.2.4	Depth Masking . . . . .	242
15.2.5	Visualizing Architecture Using Blueprint Rendering . . . . .	244
15.3	Sketchy Rendering . . . . .	244
15.3.1	Edges and Color Patches . . . . .	245
15.3.2	Applying Uncertainty . . . . .	245
15.3.3	Adjusting Depth . . . . .	247
15.3.4	Variations of Sketchy Drawing . . . . .	247
15.3.5	Controlling Uncertainty . . . . .	248
15.3.6	Reducing the Shower-Door Effect . . . . .	250
15.4	Conclusion . . . . .	251
15.5	References . . . . .	252

## Chapter 16

### Accurate Atmospheric Scattering . . . . . 253

*Sean O’Neil*

16.1	Introduction . . . . .	253
16.2	Solving the Scattering Equations . . . . .	254
16.2.1	Rayleigh Scattering vs. Mie Scattering . . . . .	255
16.2.2	The Phase Function . . . . .	256
16.2.3	The Out-Scattering Equation . . . . .	256
16.2.4	The In-Scattering Equation . . . . .	257
16.2.5	The Surface-Scattering Equation . . . . .	257
16.3	Making It Real-Time . . . . .	258
16.4	Squeezing It into a Shader . . . . .	260
16.4.1	Eliminating One Dimension . . . . .	260
16.4.2	Eliminating the Other Dimension . . . . .	261



16.5	Implementing the Scattering Shaders . . . . .	262
16.5.1	The Vertex Shader . . . . .	262
16.5.2	The Fragment Shader . . . . .	264
16.6	Adding High-Dynamic-Range Rendering . . . . .	265
16.7	Conclusion . . . . .	266
16.8	References . . . . .	267

## Chapter 17

### Efficient Soft-Edged Shadows Using Pixel Shader Branching . . . . . 269

*Yury Uralsky, NVIDIA Corporation*

17.1	Current Shadowing Techniques . . . . .	270
17.2	Soft Shadows with a Single Shadow Map . . . . .	271
17.2.1	Blurring Hard-Edged Shadows . . . . .	271
17.2.2	Improving Efficiency . . . . .	274
17.2.3	Implementation Details . . . . .	277
17.3	Conclusion . . . . .	281
17.4	References . . . . .	282

## Chapter 18

### Using Vertex Texture Displacement for Realistic

### Water Rendering . . . . . 283

*Yuri Kryachko, 1C:Maddox Games*

18.1	Water Models . . . . .	283
18.2	Implementation . . . . .	284
18.2.1	Water Surface Model . . . . .	284
18.2.2	Implementation Details . . . . .	285
18.2.3	Sampling Height Maps . . . . .	286
18.2.4	Quality Improvements and Optimizations . . . . .	288
18.2.5	Rendering Local Perturbations . . . . .	292
18.3	Conclusion . . . . .	294
18.4	References . . . . .	294

## Chapter 19

### Generic Refraction Simulation . . . . . 295

*Tiago Sousa, Crytek*

19.1	Basic Technique . . . . .	296
19.2	Refraction Mask . . . . .	297

19.3	Examples	300
19.3.1	Water Simulation	300
19.3.2	Glass Simulation	303
19.4	Conclusion	305
19.5	References	305

## **PART III HIGH-QUALITY RENDERING**

**307**

### **Chapter 20**

#### **Fast Third-Order Texture Filtering** . . . . . **313**

*Christian Sigg, ETH Zurich*

*Markus Hadwiger, VRVis Research Center*

20.1	Higher-Order Filtering	314
20.2	Fast Recursive Cubic Convolution	315
20.3	Mipmapping	320
20.4	Derivative Reconstruction	324
20.5	Conclusion	327
20.6	References	328

### **Chapter 21**

#### **High-Quality Antialiased Rasterization** . . . . . **331**

*Dan Wexler, NVIDIA Corporation*

*Eric Enderton, NVIDIA Corporation*

21.1	Overview	331
21.2	Downsampling	334
21.2.1	Comparison to Existing Hardware and Software	334
21.2.2	Downsampling on the GPU	336
21.3	Padding	336
21.4	Filter Details	337
21.5	Two-Pass Separable Filtering	338
21.6	Tiling and Accumulation	339
21.7	The Code	339
21.7.1	The Rendering Loop	340
21.7.2	The Downsample Class	341
21.7.3	Implementation Details	343
21.8	Conclusion	344
21.9	References	344

## Chapter 22

### Fast Filtered Lines ..... 345

*Eric Chan, Massachusetts Institute of Technology*

*Frédo Durand, Massachusetts Institute of Technology*

22.1	Why Sharp Lines Look Bad. ....	345
22.2	Bandlimiting the Signal. ....	347
22.2.1	Prefiltering. ....	347
22.3	The Preprocess. ....	349
22.4	Runtime. ....	351
22.4.1	Line Setup (CPU). ....	352
22.4.2	Table Lookups (GPU). ....	353
22.5	Implementation Issues. ....	355
22.5.1	Drawing Fat Lines. ....	355
22.5.2	Compositing Multiple Lines. ....	355
22.6	Examples. ....	356
22.7	Conclusion. ....	358
22.8	References. ....	359

## Chapter 23

### Hair Animation and Rendering in the Nalu Demo ..... 361

*Hubert Nguyen, NVIDIA Corporation*

*William Donnelly, NVIDIA Corporation*

23.1	Hair Geometry. ....	362
23.1.1	Layout and Growth. ....	362
23.1.2	Controlling the Hair. ....	362
23.1.3	Data Flow. ....	364
23.1.4	Tessellation. ....	364
23.1.5	Interpolation. ....	364
23.2	Dynamics and Collisions. ....	366
23.2.1	Constraints. ....	366
23.2.2	Collisions. ....	367
23.2.3	Fins. ....	368
23.3	Hair Shading. ....	369
23.3.1	A Real-Time Reflectance Model for Hair. ....	369
23.3.2	Real-Time Volumetric Shadows in Hair. ....	375
23.4	Conclusion and Future Work. ....	378
23.5	References. ....	380

## Chapter 24

### Using Lookup Tables to Accelerate Color Transformations . . . . . 381

*Jeremy Selan, Sony Pictures Imageworks*

24.1	Lookup Table Basics . . . . .	381
24.1.1	One-Dimensional LUTs . . . . .	382
24.1.2	Three-Dimensional LUTs . . . . .	383
24.1.3	Interpolation . . . . .	385
24.2	Implementation . . . . .	386
24.2.1	Strategy for Mapping LUTs to the GPU . . . . .	386
24.2.2	Cg Shader . . . . .	386
24.2.3	System Integration . . . . .	389
24.2.4	Extending 3D LUTs for Use with High-Dynamic-Range Imagery . . . . .	390
24.3	Conclusion . . . . .	392
24.4	References . . . . .	392

## Chapter 25

### GPU Image Processing in Apple's Motion . . . . . 393

*Pete Warden, Apple Computer*

25.1	Design . . . . .	393
25.1.1	Loves and Loathings . . . . .	394
25.1.2	Pick a Language . . . . .	396
25.1.3	CPU Fallback . . . . .	396
25.2	Implementation . . . . .	397
25.2.1	GPU Resource Limits . . . . .	397
25.2.2	Division by Zero . . . . .	399
25.2.3	Loss of Vertex Components . . . . .	400
25.2.4	Bilinear Filtering . . . . .	400
25.2.5	High-Precision Storage . . . . .	405
25.3	Debugging . . . . .	406
25.4	Conclusion . . . . .	407
25.5	References . . . . .	408

## Chapter 26

### Implementing Improved Perlin Noise . . . . . 409

*Simon Green, NVIDIA Corporation*

26.1	Random but Smooth . . . . .	409
26.2	Storage vs. Computation . . . . .	410

26.3	Implementation Details . . . . .	411
26.3.1	Optimization . . . . .	415
26.4	Conclusion . . . . .	415
26.5	References . . . . .	416

## Chapter 27

### Advanced High-Quality Filtering. . . . . 417

*Justin Novosad, discreet*

27.1	Implementing Filters on GPUs . . . . .	417
27.1.1	Accessing Image Samples . . . . .	418
27.1.2	Convolution Filters . . . . .	419
27.2	The Problem of Digital Image Resampling . . . . .	422
27.2.1	Background . . . . .	423
27.2.2	Antialiasing . . . . .	423
27.2.3	Image Reconstruction . . . . .	427
27.3	Shock Filtering: A Method for Deblurring Images . . . . .	430
27.4	Filter Implementation Tips . . . . .	433
27.5	Advanced Applications . . . . .	433
27.5.1	Time Warping . . . . .	433
27.5.2	Motion Blur Removal . . . . .	434
27.5.3	Adaptive Texture Filtering . . . . .	434
27.6	Conclusion . . . . .	434
27.7	References . . . . .	435

## Chapter 28

### Mipmap-Level Measurement . . . . . 437

*Iain Cantlay, Climax Entertainment*

28.1	Which Mipmap Level Is Visible? . . . . .	438
28.2	GPU to the Rescue . . . . .	439
28.2.1	Counting Pixels . . . . .	439
28.2.2	Practical Considerations in an Engine . . . . .	442
28.2.3	Extensions . . . . .	445
28.3	Sample Results . . . . .	447
28.4	Conclusion . . . . .	448
28.5	References . . . . .	449

**Chapter 29**

**Streaming Architectures and Technology Trends . . . . . 457**

*John Owens, University of California, Davis*

29.1 Technology Trends . . . . . 457

    29.1.1 Core Technology Trends . . . . . 458

    29.1.2 Consequences . . . . . 458

29.2 Keys to High-Performance Computing . . . . . 461

    29.2.1 Methods for Efficient Computation . . . . . 461

    29.2.2 Methods for Efficient Communication . . . . . 462

    29.2.3 Contrast to CPUs . . . . . 463

29.3 Stream Computation . . . . . 464

    29.3.1 The Stream Programming Model . . . . . 464

    29.3.2 Building a Stream Processor . . . . . 466

29.4 The Future and Challenges . . . . . 468

    29.4.1 Challenge: Technology Trends . . . . . 468

    29.4.2 Challenge: Power Management . . . . . 468

    29.4.3 Challenge: Supporting More Programmability and  
        Functionality . . . . . 469

    29.4.4 Challenge: GPU Functionality Subsumed by CPU  
        (or Vice Versa)? . . . . . 470

29.5 References . . . . . 470

**Chapter 30**

**The GeForce 6 Series GPU Architecture . . . . . 471**

*Emmett Kilgariff, NVIDIA Corporation*

*Randima Fernando, NVIDIA Corporation*

30.1 How the GPU Fits into the Overall Computer System . . . . . 471

30.2 Overall System Architecture . . . . . 473

    30.2.1 Functional Block Diagram for Graphics Operations . . . . . 473

    30.2.2 Functional Block Diagram for Non-Graphics Operations . . . . . 478

30.3 GPU Features . . . . . 481

    30.3.1 Fixed-Function Features . . . . . 481

    30.3.2 Shader Model 3.0 Programming Model . . . . . 483

    30.3.3 Supported Data Storage Formats . . . . . 488

30.4 Performance . . . . . 488

30.5	Achieving Optimal Performance . . . . .	490
30.5.1	Use Z-Culling Aggressively . . . . .	490
30.5.2	Exploit Texture Math When Loading Data . . . . .	490
30.5.3	Use Branching in Fragment Programs Judiciously . . . . .	490
30.5.4	Use fp16 Intermediate Values Wherever Possible . . . . .	491
30.6	Conclusion . . . . .	491

## Chapter 31

### Mapping Computational Concepts to GPUs . . . . . 493

*Mark Harris, NVIDIA Corporation*

31.1	The Importance of Data Parallelism . . . . .	493
31.1.1	What Kinds of Computation Map Well to GPUs? . . . . .	494
31.1.2	Example: Simulation on a Grid . . . . .	495
31.1.3	Stream Communication: Gather vs. Scatter. . . . .	496
31.2	An Inventory of GPU Computational Resources. . . . .	497
31.2.1	Programmable Parallel Processors. . . . .	497
31.3	CPU-GPU Analogies . . . . .	500
31.3.1	Streams: GPU Textures = CPU Arrays . . . . .	500
31.3.2	Kernels: GPU Fragment Programs = CPU “Inner Loops”.. . . .	500
31.3.3	Render-to-Texture = Feedback. . . . .	501
31.3.4	Geometry Rasterization = Computation Invocation . . . . .	501
31.3.5	Texture Coordinates = Computational Domain . . . . .	501
31.3.6	Vertex Coordinates = Computational Range. . . . .	502
31.3.7	Reductions . . . . .	502
31.4	From Analogies to Implementation . . . . .	503
31.4.1	Putting It All Together: A Basic GPGPU Framework. . . . .	503
31.5	A Simple Example. . . . .	505
31.6	Conclusion . . . . .	508
31.7	References. . . . .	508

## Chapter 32

### Taking the Plunge into GPU Computing . . . . . 509

*Ian Buck, Stanford University*

32.1	Choosing a Fast Algorithm . . . . .	509
32.1.1	Locality, Locality, Locality . . . . .	510
32.1.2	Letting Computation Rule . . . . .	511
32.1.3	Considering Download and Readback. . . . .	512
32.2	Understanding Floating Point . . . . .	513
32.2.1	Address Calculation . . . . .	514

32.3	Implementing Scatter . . . . .	515
32.3.1	Converting to Gather . . . . .	515
32.3.2	Address Sorting . . . . .	516
32.3.3	Rendering Points . . . . .	518
32.4	Conclusion . . . . .	518
32.5	References . . . . .	519

## Chapter 33

### Implementing Efficient Parallel Data Structures on GPUs . . . . . 521

*Aaron Lefohn, University of California, Davis*

*Joe Kniss, University of Utah*

*John Owens, University of California, Davis*

33.1	Programming with Streams . . . . .	521
33.2	The GPU Memory Model . . . . .	524
33.2.1	Memory Hierarchy . . . . .	524
33.2.2	GPU Stream Types. . . . .	525
33.2.3	GPU Kernel Memory Access . . . . .	527
33.3	GPU-Based Data Structures . . . . .	528
33.3.1	Multidimensional Arrays . . . . .	528
33.3.2	Structures . . . . .	534
33.3.3	Sparse Data Structures . . . . .	535
33.4	Performance Considerations . . . . .	540
33.4.1	Dependent Texture Reads . . . . .	540
33.4.2	Computational Frequency . . . . .	541
33.4.3	Pbuffer Survival Guide. . . . .	541
33.5	Conclusion . . . . .	543
33.6	References . . . . .	544

## Chapter 34

### GPU Flow-Control Idioms . . . . . 547

*Mark Harris, NVIDIA Corporation*

*Ian Buck, Stanford University*

34.1	Flow-Control Challenges. . . . .	547
34.2	Basic Flow-Control Strategies . . . . .	549
34.2.1	Predication . . . . .	549
34.2.2	Moving Branching up the Pipeline . . . . .	549
34.2.3	Z-Cull . . . . .	550
34.2.4	Branching Instructions. . . . .	553
34.2.5	Choosing a Branching Mechanism. . . . .	553
34.3	Data-Dependent Looping with Occlusion Queries . . . . .	554
34.4	Conclusion . . . . .	555



## Chapter 35

### GPU Program Optimization. . . . . 557

*Cliff Woolley, University of Virginia*

35.1	Data-Parallel Computing. . . . .	557
35.1.1	Instruction-Level Parallelism . . . . .	558
35.1.2	Data-Level Parallelism . . . . .	560
35.2	Computational Frequency . . . . .	561
35.2.1	Precomputation of Loop Invariants . . . . .	563
35.2.2	Precomputation Using Lookup Tables . . . . .	564
35.2.3	Avoid Inner-Loop Branching. . . . .	566
35.2.4	The Swizzle Operator . . . . .	566
35.3	Profiling and Load Balancing . . . . .	568
35.4	Conclusion . . . . .	570
35.5	References . . . . .	570

## Chapter 36

### Stream Reduction Operations for GPGPU Applications . . . . . 573

*Daniel Horn, Stanford University*

36.1	Filtering Through Compaction . . . . .	574
36.1.1	Running Sum Scan . . . . .	574
36.1.2	Scatter Through Search/Gather . . . . .	575
36.1.3	Filtering Performance. . . . .	579
36.2	Motivation: Collision Detection . . . . .	579
36.3	Filtering for Subdivision Surfaces . . . . .	583
36.3.1	Subdivision on Streaming Architectures. . . . .	584
36.4	Conclusion . . . . .	587
36.5	References . . . . .	587

## **PART V IMAGE-ORIENTED COMPUTING**

**591**

## Chapter 37

### Octree Textures on the GPU . . . . . 595

*Sylvain Lefebvre, GRAVIR/IMAG – INRIA*

*Samuel Hornus, GRAVIR/IMAG – INRIA*

*Fabrice Neyret, GRAVIR/IMAG – INRIA*

37.1	A GPU-Accelerated Hierarchical Structure: The $N^3$ -Tree. . . . .	597
37.1.1	Definition . . . . .	597
37.1.2	Implementation. . . . .	598

37.2	Application 1: Painting on Meshes . . . . .	602
37.2.1	Creating the Octree . . . . .	603
37.2.2	Painting . . . . .	604
37.2.3	Rendering . . . . .	604
37.2.4	Converting the Octree Texture to a Standard 2D Texture . . . . .	607
37.3	Application 2: Surface Simulation . . . . .	611
37.4	Conclusion . . . . .	612
37.5	References . . . . .	613

## Chapter 38

### High-Quality Global Illumination Rendering Using Rasterization . . . . . 615

*Toshiya Hachisuka, The University of Tokyo*

38.1	Global Illumination via Rasterization . . . . .	616
38.2	Overview of Final Gathering . . . . .	617
38.2.1	Two-Pass Methods . . . . .	617
38.2.2	Final Gathering . . . . .	618
38.2.3	Problems with Two-Pass Methods . . . . .	619
38.3	Final Gathering via Rasterization . . . . .	621
38.3.1	Clustering of Final Gathering Rays . . . . .	621
38.3.2	Ray Casting as Multiple Parallel Projection . . . . .	623
38.4	Implementation Details . . . . .	625
38.4.1	Initialization . . . . .	625
38.4.2	Depth Peeling . . . . .	626
38.4.3	Sampling . . . . .	627
38.4.4	Performance . . . . .	627
38.5	A Global Illumination Renderer on the GPU . . . . .	627
38.5.1	The First Pass . . . . .	628
38.5.2	Generating Visible Points Data . . . . .	628
38.5.3	The Second Pass . . . . .	629
38.5.4	Additional Solutions . . . . .	629
38.6	Conclusion . . . . .	632
38.7	References . . . . .	632

## Chapter 39

### Global Illumination Using Progressive Refinement Radiosity . . . . . 635

*Greg Coombe, University of North Carolina at Chapel Hill*

*Mark Harris, NVIDIA Corporation*

39.1	Radiosity Foundations . . . . .	636
39.1.1	Progressive Refinement . . . . .	637

39.2 GPU Implementation . . . . .	638
39.2.1 Visibility Using Hemispherical Projection . . . . .	639
39.2.2 Form Factor Computation . . . . .	641
39.2.3 Choosing the Next Shooter . . . . .	643
39.3 Adaptive Subdivision . . . . .	643
39.3.1 Texture Quadtree . . . . .	644
39.3.2 Quadtree Subdivision . . . . .	644
39.4 Performance . . . . .	645
39.5 Conclusion . . . . .	645
39.6 References . . . . .	647

## Chapter 40

### Computer Vision on the GPU . . . . . 649

*James Fung, University of Toronto*

40.1 Introduction . . . . .	649
40.2 Implementation Framework . . . . .	650
40.3 Application Examples . . . . .	651
40.3.1 Using Sequences of Fragment Programs for Computer Vision . . . . .	651
40.3.2 Summation Operations . . . . .	655
40.3.3 Systems of Equations for Creating Image Panoramas . . . . .	658
40.3.4 Feature Vector Computations . . . . .	661
40.4 Parallel Computer Vision Processing . . . . .	664
40.5 Conclusion . . . . .	664
40.6 References . . . . .	665

## Chapter 41

### Deferred Filtering: Rendering from Difficult Data Formats . . . . . 667

*Joe Kniss, University of Utah*

*Aaron Lefohn, University of California, Davis*

*Nathaniel Fout, University of California, Davis*

41.1 Introduction . . . . .	667
41.2 Why Defer? . . . . .	668
41.3 Deferred Filtering Algorithm . . . . .	669
41.4 Why It Works . . . . .	673
41.5 Conclusions: When to Defer . . . . .	673
41.6 References . . . . .	674

## Chapter 42

### Conservative Rasterization. . . . . 677

*Jon Hasselgren, Lund University*

*Tomas Akenine-Möller, Lund University*

*Lennart Ohlsson, Lund University*

42.1 Problem Definition . . . . .	678
42.2 Two Conservative Algorithms . . . . .	679
42.2.1 Clip Space . . . . .	681
42.2.2 The First Algorithm. . . . .	681
42.2.3 The Second Algorithm . . . . .	683
42.3 Robustness Issues . . . . .	686
42.4 Conservative Depth . . . . .	687
42.5 Results and Conclusions . . . . .	689
42.6 References. . . . .	690

## **PART VI SIMULATION AND NUMERICAL ALGORITHMS**

**691**

## Chapter 43

### GPU Computing for Protein Structure Prediction . . . . . 695

*Paulius Micikevicius, Armstrong Atlantic State University*

43.1 Introduction . . . . .	695
43.2 The Floyd-Warshall Algorithm and Distance-Bound Smoothing. . . . .	697
43.3 GPU Implementation. . . . .	698
43.3.1 Dynamic Updates . . . . .	698
43.3.2 Indexing Data Textures . . . . .	698
43.3.3 The Triangle Approach . . . . .	699
43.3.4 Vectorization . . . . .	699
43.4 Experimental Results . . . . .	701
43.5 Conclusions and Further Work . . . . .	701
43.6 References . . . . .	702

## Chapter 44

### A GPU Framework for Solving Systems of Linear Equations . . . . . 703

*Jens Krüger, Technische Universität München*

*Rüdiger Westermann, Technische Universität München*

44.1 Overview. . . . .	703
44.2 Representation . . . . .	704
44.2.1 The “Single Float” Representation. . . . .	704
44.2.2 Vectors. . . . .	704
44.2.3 Matrices . . . . .	706

44.3	Operations	708
44.3.1	Vector Arithmetic	709
44.3.2	Vector Reduce	709
44.3.3	Matrix-Vector Product	710
44.3.4	Putting It All Together	712
44.3.5	Conjugate Gradient Solver	713
44.4	A Sample Partial Differential Equation	714
44.4.1	The Crank-Nicholson Scheme	716
44.5	Conclusion	718
44.6	References	718

## Chapter 45

### Options Pricing on the GPU ..... 719

*Craig Kolb, NVIDIA Corporation*

*Matt Pharr, NVIDIA Corporation*

45.1	What Are Options?	719
45.2	The Black-Scholes Model	721
45.3	Lattice Models	725
45.3.1	The Binomial Model	725
45.3.2	Pricing European Options	726
45.4	Conclusion	730
45.5	References	731

## Chapter 46

### Improved GPU Sorting ..... 733

*Peter Kipfer, Technische Universität München*

*Rüdiger Westermann, Technische Universität München*

46.1	Sorting Algorithms	733
46.2	A Simple First Approach	734
46.3	Fast Sorting	735
46.3.1	Implementing Odd-Even Merge Sort	737
46.4	Using All GPU Resources	738
46.4.1	Implementing Bitonic Merge Sort	743
46.5	Conclusion	745
46.6	References	746

## Chapter 47

### Flow Simulation with Complex Boundaries . . . . . 747

*Wei Li, Siemens Corporate Research*

*Zhe Fan, Stony Brook University*

*Xiaoming Wei, Stony Brook University*

*Arie Kaufman, Stony Brook University*

47.1	Introduction . . . . .	747
47.2	The Lattice Boltzmann Method . . . . .	748
47.3	GPU-Based LBM . . . . .	749
47.3.1	Algorithm Overview . . . . .	749
47.3.2	Packing. . . . .	751
47.3.3	Streaming . . . . .	752
47.4	GPU-Based Boundary Handling . . . . .	753
47.4.1	GPU-Based Voxelization . . . . .	754
47.4.2	Periodic Boundaries. . . . .	756
47.4.3	Outflow Boundaries. . . . .	756
47.4.4	Obstacle Boundaries. . . . .	757
47.5	Visualization . . . . .	759
47.6	Experimental Results. . . . .	760
47.7	Conclusion . . . . .	761
47.8	References. . . . .	763

## Chapter 48

### Medical Image Reconstruction with the FFT . . . . . 765

*Thilaka Sumanaweera, Siemens Medical Solutions USA*

*Donald Liu, Siemens Medical Solutions USA*

48.1	Background. . . . .	765
48.2	The Fourier Transform . . . . .	766
48.3	The FFT Algorithm. . . . .	767
48.4	Implementation on the GPU . . . . .	768
48.4.1	Approach 1: Mostly Loading the Fragment Processor. . . . .	770
48.4.2	Approach 2: Loading the Vertex Processor, the Rasterizer, and the Fragment Processor . . . . .	772
48.4.3	Load Balancing . . . . .	775
48.4.4	Benchmarking Results . . . . .	775
48.5	The FFT in Medical Imaging . . . . .	776
48.5.1	Magnetic Resonance Imaging . . . . .	776
48.5.2	Results in MRI . . . . .	778
48.5.3	Ultrasonic Imaging . . . . .	780
48.6	Conclusion . . . . .	783
48.7	References. . . . .	784

### Index . . . . . 785