

# Contents

---

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

## **PART I NATURAL EFFECTS** **1**

---

<b>Chapter 1</b>	
<b>Effective Water Simulation from Physical Models .....</b>	<b>5</b>
<i>Mark Finch, Cyan Worlds</i>	
1.1 Goals and Scope .....	5
1.2 The Sum of Sines Approximation .....	7
1.2.1 Selecting the Waves .....	7
1.2.2 Normals and Tangents .....	9
1.2.3 Geometric Waves .....	11
1.2.4 Texture Waves .....	15
1.3 Authoring .....	18
1.3.1 Using Depth .....	19
1.3.2 Overrides .....	21
1.3.3 Edge-Length Filtering .....	21
1.3.4 Texture Coordinates .....	22
1.4 Runtime Processing .....	23
1.4.1 Bump-Environment Mapping Parameters .....	23
1.4.2 Vertex and Pixel Processing .....	27
1.5 Conclusion .....	28
1.6 References .....	28

## Chapter 2

### Rendering Water Caustics . . . . . 31

*Juan Guardado, NVIDIA*

*Daniel Sánchez-Crespo, Universitat Pompeu Fabra/Novarama Technology*

2.1 Introduction . . . . .	31
2.2 Computing Caustics . . . . .	32
2.3 Our Approach . . . . .	35
2.4 Implementation Using OpenGL . . . . .	37
2.5 Implementation Using a High-Level Shading Language . . . . .	38
2.6 Conclusion . . . . .	44
2.7 References . . . . .	44

## Chapter 3

### Skin in the “Dawn” Demo . . . . . 45

*Curtis Beeson, NVIDIA*

*Kevin Bjorke, NVIDIA*

3.1 Introduction . . . . .	45
3.2 Skin Shading . . . . .	45
3.3 Lighting the Scene . . . . .	47
3.3.1 A High-Dynamic-Range Environment . . . . .	47
3.3.2 The Occlusion Term . . . . .	49
3.4 How Skin Responds to Light . . . . .	50
3.5 Implementation . . . . .	51
3.5.1 The Vertex Shader . . . . .	52
3.5.2 The Fragment Shader . . . . .	58
3.6 Conclusion . . . . .	61
3.7 References . . . . .	61

## Chapter 4

### Animation in the “Dawn” Demo . . . . . 63

*Curtis Beeson, NVIDIA*

4.1 Introduction . . . . .	63
4.2 Mesh Animation . . . . .	63
4.3 Morph Targets . . . . .	65
4.3.1 Morph Targets in a High-Level Language . . . . .	65
4.3.2 Morph Target Implementation . . . . .	67
4.4 Skinning . . . . .	69
4.4.1 Accumulated Matrix Skinning . . . . .	71
4.5 Conclusion . . . . .	72
4.6 References . . . . .	72

<b>Chapter 5</b>	
<b>Implementing Improved Perlin Noise</b> . . . . .	<b>73</b>
<i>Ken Perlin, New York University</i>	
5.1 The Noise Function . . . . .	73
5.2 The Original Implementation . . . . .	74
5.3 Deficiencies of the Original Implementation . . . . .	75
5.4 Improvements to Noise . . . . .	78
5.5 How to Make Good Fake Noise in Pixel Shaders . . . . .	80
5.6 Making Bumps Without Looking at Neighboring Vertices . . . . .	82
5.7 Conclusion . . . . .	84
5.8 References . . . . .	85

<b>Chapter 6</b>	
<b>Fire in the “Vulcan” Demo</b> . . . . .	<b>87</b>
<i>Hubert Nguyen, NVIDIA</i>	
6.1 Creating Realistic Flames . . . . .	87
6.2 Implementing Animated Sprites . . . . .	89
6.2.1 Animating Flames and Smoke . . . . .	90
6.2.2 Adding Variety to the Flames . . . . .	92
6.2.3 Storing the Animation . . . . .	94
6.2.4 Blending Flames and Smoke . . . . .	94
6.3 Particle Motion . . . . .	94
6.4 Performance . . . . .	96
6.4.1 Layer Composition . . . . .	97
6.4.2 Custom Sprites . . . . .	97
6.5 Post-Rendering Effects . . . . .	99
6.5.1 Glow . . . . .	99
6.5.2 Heat Shimmer . . . . .	101
6.5.3 Grain . . . . .	101
6.5.4 The Final Program . . . . .	104
6.6 Conclusion . . . . .	105

<b>Chapter 7</b>	
<b>Rendering Countless Blades of Waving Grass</b> . . . . .	<b>107</b>
<i>Kurt Pelzer, Piranha Bytes</i>	
7.1 Introduction . . . . .	107
7.2 Overview . . . . .	107
7.3 Preparation of the Grass Objects . . . . .	108
7.3.1 Grass Texture . . . . .	109
7.3.2 Grass Objects . . . . .	109

7.4 Animation . . . . .	111
7.4.1 The General Idea . . . . .	112
7.4.2 Animation per Cluster of Grass Objects . . . . .	113
7.4.3 Animation per Vertex . . . . .	115
7.4.4 Animation per Grass Object . . . . .	117
7.5 Conclusion . . . . .	120
7.6 Further Reading . . . . .	121

## Chapter 8

### Simulating Diffraction . . . . . 123

*Jos Stam, Alias Systems*

8.1 What Is Diffraction? . . . . .	123
8.1.1 The Wave Theory of Light . . . . .	124
8.1.2 The Physics of Diffraction . . . . .	124
8.2 Implementation . . . . .	127
8.3 Results . . . . .	131
8.4 Conclusion . . . . .	132
8.5 References . . . . .	132

## PART II LIGHTING AND SHADOWS

133

## Chapter 9

### Efficient Shadow Volume Rendering . . . . . 137

*Morgan McGuire, Brown University*

9.1 Introduction . . . . .	137
9.1.1 Where to Use Shadow Volumes . . . . .	138
9.2 Program Structure . . . . .	141
9.2.1 Multipass Rendering . . . . .	141
9.2.2 Vertex Buffer Structure . . . . .	144
9.2.3 Working at Infinity . . . . .	145
9.3 Detailed Discussion . . . . .	148
9.3.1 The Math . . . . .	148
9.3.2 The Code . . . . .	151
9.3.3 The <code>markShadows</code> Method . . . . .	151
9.3.4 The <code>findBackfaces</code> Method . . . . .	153
9.3.5 Light and Dark Caps . . . . .	153
9.3.6 Sides . . . . .	155
9.4 Debugging . . . . .	157

9.5	Geometry Optimizations . . . . .	158
9.5.1	Directional Lights . . . . .	158
9.5.2	Point Lights and Spotlights . . . . .	159
9.5.3	Culling Shadow Volumes . . . . .	159
9.5.4	Uncapped Performance . . . . .	160
9.6	Fill-Rate Optimizations . . . . .	162
9.6.1	Finite Volumes . . . . .	162
9.6.2	XY Clipping . . . . .	162
9.6.3	Z-Bounds . . . . .	163
9.7	Future Shadows . . . . .	165
9.8	References . . . . .	166

## Chapter 10

### Cinematic Lighting . . . . . 167

*Fabio Pellacini, Pixar Animation Studios*

*Kiril Vidimce, Pixar Animation Studios*

10.1	Introduction . . . . .	167
10.2	A Direct Lighting Illumination Model . . . . .	169
10.2.1	Selection . . . . .	170
10.2.2	Color . . . . .	170
10.2.3	Shaping . . . . .	170
10.2.4	Shadowing . . . . .	171
10.2.5	Texturing . . . . .	173
10.2.6	Results . . . . .	173
10.3	The Uberlight Shader . . . . .	175
10.4	Performance Concerns . . . . .	182
10.4.1	Speed . . . . .	182
10.4.2	Expense . . . . .	182
10.4.3	Optimization . . . . .	182
10.5	Conclusion . . . . .	182
10.6	References . . . . .	183

## Chapter 11

### Shadow Map Antialiasing. . . . . 185

*Michael Bunnell, NVIDIA*

*Fabio Pellacini, Pixar Animation Studios*

11.1	Introduction . . . . .	185
11.2	Percentage-Closer Filtering . . . . .	186
11.3	A Brute-Force Implementation . . . . .	187

11.4 Using Fewer Samples . . . . .	188
11.5 Why It Works . . . . .	189
11.6 Conclusion . . . . .	192
11.7 References . . . . .	192

## Chapter 12

### Omnidirectional Shadow Mapping . . . . . 193

*Philipp S. Gerasimov, iXBT.com*

12.1 Introduction . . . . .	193
12.1.1 Stencil Shadows . . . . .	195
12.1.2 Shadow Mapping . . . . .	195
12.2 The Shadow-Mapping Algorithm . . . . .	195
12.2.1 Conditions . . . . .	195
12.2.2 The Algorithm . . . . .	196
12.2.3 Texture Format . . . . .	197
12.2.4 The Size of the Shadow Map . . . . .	198
12.2.5 The Range of Values for Geometry . . . . .	198
12.3 Implementation . . . . .	198
12.3.1 System Requirements . . . . .	198
12.3.2 Resource Creation . . . . .	199
12.3.3 Rendering Phase 1: Rendering into the Shadow Map . . . . .	199
12.3.4 Rendering Phase 2: Base Rendering . . . . .	200
12.3.5 The Lighting Calculation . . . . .	200
12.3.6 The Shadow Calculation . . . . .	200
12.3.7 Tips and Tricks . . . . .	201
12.3.8 Finalizing the Shading Pass (Lighting × Shadow) . . . . .	202
12.4 Adding Soft Shadows . . . . .	202
12.5 Conclusion . . . . .	203
12.6 References . . . . .	203

## Chapter 13

### Generating Soft Shadows Using Occlusion Interval Maps . . . . . 205

*William Donnelly, University of Waterloo*

*Joe Demers, NVIDIA*

13.1 The Gas Station . . . . .	205
13.2 The Algorithm . . . . .	207
13.3 Creating the Maps . . . . .	208

13.4	Rendering . . . . .	209
13.5	Limitations . . . . .	211
13.6	Conclusion . . . . .	213
13.7	References . . . . .	215

## Chapter 14

### Perspective Shadow Maps: Care and Feeding. . . . . 217

*Simon Kozlov, SoftLab-NSK*

14.1	Introduction . . . . .	217
14.2	Problems with the PSM Algorithm . . . . .	219
14.2.1	Virtual Cameras . . . . .	219
14.2.2	The Light Camera . . . . .	225
14.2.3	Biasing . . . . .	233
14.3	Tricks for Better Shadow Maps . . . . .	237
14.3.1	Filtering . . . . .	237
14.3.2	Blurring . . . . .	239
14.4	Results . . . . .	242
14.5	References . . . . .	244

## Chapter 15

### Managing Visibility for Per-Pixel Lighting. . . . . 245

*John O'Rorke, Monolith Productions*

15.1	Visibility in a GPU Book? . . . . .	245
15.2	Batches and Per-Pixel Lighting . . . . .	246
15.2.1	A Per-Pixel Example . . . . .	246
15.2.2	Just How Many Batches, Anyway? . . . . .	247
15.3	Visibility As Sets . . . . .	248
15.3.1	The Visible Set . . . . .	248
15.3.2	The Lights Set . . . . .	248
15.3.3	The Illumination Set . . . . .	248
15.3.4	The Shadow Set . . . . .	249
15.4	Generating Sets . . . . .	250
15.4.1	The Visible Set . . . . .	250
15.4.2	The Lights Set . . . . .	250
15.4.3	The Illumination Set . . . . .	251
15.4.4	The Shadow Set . . . . .	251
15.5	Visibility for Fill Rate . . . . .	255
15.6	Practical Application . . . . .	255
15.7	Conclusion . . . . .	256
15.8	References . . . . .	257

## Chapter 16

**Real-Time Approximations to Subsurface Scattering..... 263***Simon Green, NVIDIA*

16.1 The Visual Effects of Subsurface Scattering.....	263
16.2 Simple Scattering Approximations .....	264
16.3 Simulating Absorption Using Depth Maps.....	266
16.3.1 Implementation Details.....	270
16.3.2 More Sophisticated Scattering Models .....	271
16.4 Texture-Space Diffusion .....	272
16.4.1 Possible Future Work.....	275
16.5 Conclusion .....	277
16.6 References .....	277

## Chapter 17

**Ambient Occlusion .....** 279*Matt Pharr, NVIDIA**Simon Green, NVIDIA*

17.1 Overview .....	280
17.2 The Preprocessing Step .....	281
17.3 Hardware-Accelerated Occlusion .....	283
17.4 Rendering with Ambient Occlusion Maps .....	284
17.4.1 Environment Lighting.....	286
17.5 Conclusion.....	289
17.6 Further Reading .....	292

## Chapter 18

**Spatial BRDFs .....** 293*David McAllister, NVIDIA*

18.1 What Is an SBRDF? .....	293
18.2 Details of the Representation .....	294
18.3 Rendering Using Discrete Lights.....	296
18.3.1 Texture Sampling .....	299
18.4 Rendering Using Environment Maps .....	299
18.4.1 The Math .....	300
18.4.2 The Shader Code .....	302
18.5 Conclusion .....	306
18.6 References.....	306



## Chapter 19

### Image-Based Lighting . . . . . 307

*Kevin Bjorke, NVIDIA*

19.1	Localizing Image-Based Lighting . . . . .	307
19.2	The Vertex Shader . . . . .	312
19.3	The Fragment Shader . . . . .	314
19.3.1	Additional Shader Details . . . . .	315
19.4	Diffuse IBL . . . . .	317
19.5	Shadows . . . . .	317
19.6	Using Localized Cube Maps As Backgrounds . . . . .	318
19.7	Conclusion . . . . .	320
19.8	Further Reading . . . . .	321

## Chapter 20

### Texture Bombing . . . . . 323

*R. Steven Glanville, NVIDIA*

20.1	Texture Bombing 101 . . . . .	323
20.1.1	Finding the Cell . . . . .	324
20.1.2	Sampling the Image . . . . .	324
20.1.3	Adjacent Cells' Images . . . . .	325
20.1.4	Image Priority . . . . .	326
20.1.5	Procedural Images . . . . .	326
20.1.6	Random Image Selection from Multiple Choices . . . . .	330
20.2	Technical Considerations . . . . .	331
20.2.1	Efficiency Issues . . . . .	332
20.3	Advanced Features . . . . .	332
20.3.1	Scaling and Rotating . . . . .	332
20.3.2	Controlled Variable Density . . . . .	333
20.3.3	Procedural 3D Bombing . . . . .	333
20.3.4	Time-Varying Textures . . . . .	335
20.3.5	Voronoi-Related Cellular Methods . . . . .	335
20.4	Conclusion . . . . .	337
20.5	References . . . . .	338

## Chapter 21

## Real-Time Glow . . . . . 343

*Greg James, NVIDIA**John O'Rorke, Monolith Productions*

21.1 Overview of the Technique . . . . .	346
21.2 Rendering Glows: Step by Step . . . . .	347
21.2.1 Specifying and Rendering the Sources of Glow . . . . .	347
21.2.2 Blurring the Glow Sources. . . . .	349
21.2.3 Adapting the Separable Convolution . . . . .	349
21.2.4 Convolution on the GPU. . . . .	351
21.3 Hardware-Specific Implementations . . . . .	352
21.3.1 Direct3D 9 . . . . .	352
21.3.2 Direct3D 8 . . . . .	354
21.3.3 Direct3D 7 . . . . .	355
21.4 Other Uses for Blur . . . . .	355
21.5 Adding the Effects to a Game Engine . . . . .	356
21.5.1 Rendering Context . . . . .	356
21.5.2 Aliasing Issues . . . . .	357
21.5.3 DirectX 7 Accuracy Issues . . . . .	358
21.5.4 The After-Image Effect . . . . .	359
21.5.5 The Variable Ramping Effect. . . . .	359
21.6 Conclusion . . . . .	361
21.7 References . . . . .	361

## Chapter 22

## Color Controls . . . . . 363

*Kevin BJORKE, NVIDIA*

22.1 Introduction . . . . .	363
22.2 Channel-Based Color Correction . . . . .	364
22.2.1 Levels . . . . .	364
22.2.2 Curves. . . . .	366
22.3 Multichannel Color Correction and Conversion. . . . .	368
22.3.1 Grayscale Conversion . . . . .	369
22.3.2 Color-Space Conversions. . . . .	370
22.4 References . . . . .	372

## Chapter 23

### Depth of Field: A Survey of Techniques . . . . . 375

*Joe Demers, NVIDIA*

23.1	What Is Depth of Field? . . . . .	375
23.1.1	Calculating the Circle of Confusion. . . . .	377
23.1.2	Major Techniques . . . . .	377
23.2	Ray-Traced Depth of Field . . . . .	378
23.3	Accumulation-Buffer Depth of Field . . . . .	379
23.4	Layered Depth of Field . . . . .	379
23.4.1	Analysis. . . . .	380
23.5	Forward-Mapped Z-Buffer Depth of Field . . . . .	382
23.5.1	Analysis. . . . .	382
23.6	Reverse-Mapped Z-Buffer Depth of Field . . . . .	383
23.6.1	Analysis. . . . .	386
23.7	Conclusion . . . . .	388
23.8	References. . . . .	389

## Chapter 24

### High-Quality Filtering . . . . . 391

*Kevin Bjorke, NVIDIA*

24.1	Quality vs. Speed. . . . .	391
24.1.1	Filter Kernels: Image-to-Image Conversions . . . . .	392
24.2	Understanding GPU Derivatives. . . . .	405
24.3	Analytical Antialiasing and Texturing . . . . .	407
24.3.1	Interacting with Graphics API Antialiasing . . . . .	414
24.4	Conclusion . . . . .	415
24.5	References . . . . .	415

## Chapter 25

### Fast Filter-Width Estimates with Texture Maps . . . . . 417

*Matt Pharr, NVIDIA*

25.1	The Need for Derivatives in Shaders . . . . .	418
25.2	Computing Filter Width with Textures . . . . .	420
25.3	Discussion. . . . .	423
25.4	Further Reading . . . . .	424

## Chapter 26

### The OpenEXR Image File Format ..... 425

*Florian Kainz, Industrial Light & Magic*

*Rod Bogart, Industrial Light & Magic*

*Drew Hess, Industrial Light & Magic*

26.1	What Is OpenEXR? .....	425
26.1.1	High-Dynamic-Range Images .....	425
26.1.2	A “Half” Format .....	428
26.1.3	Range of Representable Values .....	429
26.1.4	Color Resolution .....	429
26.1.5	C++ Interface .....	429
26.2	The OpenEXR File Structure .....	430
26.2.1	The Header .....	430
26.2.2	The Pixels .....	430
26.3	OpenEXR Data Compression .....	431
26.4	Using OpenEXR .....	431
26.4.1	Reading and Displaying an OpenEXR Image .....	431
26.4.2	Rendering and Writing an OpenEXR Image .....	433
26.5	Linear Pixel Values .....	438
26.6	Creating and Using HDR Images .....	441
26.7	Conclusion .....	443
26.8	References .....	443

## Chapter 27

### A Framework for Image Processing ..... 445

*Frank Jargstorff, NVIDIA*

27.1	Introduction .....	445
27.2	Framework Design .....	447
27.2.1	Operators and Filters .....	448
27.2.2	Image Data .....	449
27.2.3	Missing Pieces .....	451
27.3	Implementation .....	452
27.3.1	The Image Class .....	454
27.3.2	The ImageFilter Class .....	458
27.3.3	Implementing a Filter .....	460
27.4	A Sample Application .....	463
27.5	Performance and Limitations .....	465
27.6	Conclusion .....	467
27.7	References .....	467

**Chapter 28****Graphics Pipeline Performance . . . . . 473***Cem Cebenoyan, NVIDIA*

28.1	Overview . . . . .	473
28.1.1	The Pipeline . . . . .	473
28.1.2	Methodology . . . . .	474
28.2	Locating the Bottleneck . . . . .	475
28.2.1	Raster Operations . . . . .	476
28.2.2	Texture Bandwidth . . . . .	476
28.2.3	Fragment Shading . . . . .	476
28.2.4	Vertex Processing . . . . .	477
28.2.5	Vertex and Index Transfer . . . . .	478
28.3	Optimization . . . . .	478
28.3.1	Optimizing on the CPU . . . . .	478
28.3.2	Reducing the Cost of Vertex Transfer . . . . .	480
28.3.3	Optimizing Vertex Processing . . . . .	480
28.3.4	Speeding Up Fragment Shading . . . . .	482
28.3.5	Reducing Texture Bandwidth . . . . .	483
28.3.6	Optimizing Frame-Buffer Bandwidth . . . . .	484
28.4	Conclusion . . . . .	485
28.5	References . . . . .	486

**Chapter 29****Efficient Occlusion Culling . . . . . 487***Dean Sekulic, Croteam*

29.1	What Is Occlusion Culling? . . . . .	487
29.1.1	Occlusion Query . . . . .	487
29.1.2	Early-Z Rejection . . . . .	488
29.2	How Does Occlusion Query Work? . . . . .	488
29.3	Beginning to Use Occlusion Queries . . . . .	489
29.3.1	How to Use Occlusion Queries Properly . . . . .	490
29.3.2	Occluders vs. Occludees . . . . .	491
29.4	One Step Further . . . . .	492
29.4.1	Sorting Things Out . . . . .	493
29.4.2	A Small Caveat . . . . .	494
29.5	A Word About Bounding Boxes . . . . .	494
29.5.1	Static Objects . . . . .	494
29.5.2	Animated Objects . . . . .	495

29.6	Other Issues	496
29.6.1	CPU Overhead Too High	496
29.6.2	Rendering at High Resolutions	497
29.6.3	Fast Depth-Writing Performance	497
29.6.4	Frustum Culling	498
29.7	A Little Reminder	498
29.8	An Application: Lens Flares	499
29.8.1	Rendering Lens Flares the Old Way	501
29.8.2	Rendering Lens Flares the New Way	502
29.9	Conclusion	503
29.10	References	503

## Chapter 30

### The Design of FX Composer ..... 505

*Christopher Maughan, NVIDIA*

30.1	Tools Development	505
30.2	Initial Features and Target Audience	506
30.3	Object Design	506
30.3.1	Benefits of the Interface Approach	509
30.4	File Format	512
30.5	User Interface	514
30.6	Direct3D Graphics Implementation	515
30.6.1	Device Windows	515
30.6.2	Direct3D Effects	516
30.6.3	ID3DXEffectCompiler	516
30.6.4	ID3DXEffect	517
30.7	Scene Management	517
30.7.1	Geometry Pipes	517
30.8	Conclusion	518
30.9	References	519

## Chapter 31

### Using FX Composer ..... 521

*Christopher Maughan, NVIDIA*

31.1	Getting Started	521
31.1.1	The Materials Panel	523
31.1.2	The Scene Graph Panel	524
31.1.3	The Editor Window	525
31.1.4	The Shader Perf Panel	527

31.1.5	The Properties Panel . . . . .	528
31.1.6	The Scene Panel . . . . .	530
31.1.7	The Textures Panel . . . . .	531
31.1.8	The Tasks Panel . . . . .	532
31.1.9	The Log Panel . . . . .	533
31.2	Sample Project . . . . .	533
31.3	Conclusion . . . . .	535

## Chapter 32

### An Introduction to Shader Interfaces. . . . . 537

*Matt Pharr, NVIDIA*

32.1	The Basics of Shader Interfaces . . . . .	539
32.2	A Flexible Description of Lights . . . . .	542
32.3	Material Trees . . . . .	545
32.4	Conclusion . . . . .	549
32.5	References . . . . .	550

## Chapter 33

### Converting Production RenderMan Shaders to Real-Time . . . . . 551

*Stephen Marshall, Sony Pictures Imageworks*

33.1	Introduction . . . . .	551
33.1.1	Conversion by Example . . . . .	552
33.2	Lights . . . . .	552
33.2.1	Light Sources . . . . .	552
33.2.2	Light Source Shaders . . . . .	553
33.2.3	Additional Lighting Parameters . . . . .	554
33.3	The Vertex Program vs. the Fragment Program . . . . .	555
33.4	Using Vertex and Fragment Programs . . . . .	555
33.5	Optimization Techniques on the Fragment Program . . . . .	556
33.5.1	Moving Code to the Application Level . . . . .	557
33.5.2	Moving Code to the Vertex Program . . . . .	557
33.5.3	Optimizing Through Texture Lookups . . . . .	559
33.5.4	Optimizing for Vectorization . . . . .	560
33.5.5	Final Optimizations . . . . .	560
33.6	Results and Conclusions . . . . .	561
33.7	References . . . . .	565

## Chapter 34

### Integrating Hardware Shading into Cinema 4D ..... 567

*Jörn Loviscach, Hochschule Bremen*

34.1	Introduction .....	567
34.2	Connecting Cinema 4D to CgFX .....	570
34.3	Shader and Parameter Management .....	572
34.4	Emulating the Offline Renderer .....	573
34.5	Results and Performance .....	577
34.6	Lessons Learned .....	578
34.7	References .....	580

## Chapter 35

### Leveraging High-Quality Software Rendering Effects

### in Real-Time Applications ..... 581

*Alexandre Jean Claude, Softimage*

*Marc Stevens, Softimage*

35.1	Introduction .....	581
35.2	The Content Pipeline for Hardware Rendering .....	582
35.3	Components of Hardware Rendering .....	584
35.3.1	Geometric Data .....	584
35.3.2	Attribute Maps .....	584
35.4	Generating the Components .....	587
35.4.1	Creating the Geometry .....	588
35.4.2	Rendering to Textures and Vertices .....	588
35.5	Test Case and Results .....	593
35.6	Conclusion .....	598
35.7	References .....	599

## Chapter 36

### Integrating Shaders into Applications ..... 601

*John O'Rorke, Monolith Productions*

36.1	Introduction .....	601
36.2	About Shaders .....	602
36.2.1	Shader Languages .....	602
36.3	The Anatomy of an Effect File .....	605
36.3.1	Variables .....	605
36.3.2	Structures .....	606
36.3.3	Passes .....	606
36.3.4	Techniques .....	606
36.3.5	Annotations .....	606



36.4	Types of Shader Data	607
36.4.1	Scene Information	608
36.4.2	Materials	608
36.4.3	Renderer Context	608
36.4.4	Vertex Data	609
36.5	Communicating with the Shader	609
36.5.1	Scene Information	609
36.5.2	Material Parameters	610
36.5.3	Vertex Format	611
36.5.4	Context	612
36.5.5	Techniques vs. Passes for Context	613
36.6	Extending the Effect File Format	613
36.6.1	Supporting the Preprocessor	613
36.6.2	Supporting Shader Variations	614
36.6.3	Adding Shader Inheritance	615
36.7	Conclusion	615
36.8	References	615

## PART VI BEYOND TRIANGLES

617

### Chapter 37

#### A Toolkit for Computation on GPUs . . . . . 621

*Ian Buck, Stanford University*

*Tim Purcell, Stanford University*

37.1	Computing with the GPU	621
37.1.1	The Programming Model	622
37.1.2	Parallel Programming	623
37.1.3	Advanced GPU Programming	624
37.2	Reduce	625
37.2.1	Parallel Reductions	625
37.2.2	Reduce Tidbits	626
37.3	Sort and Search	627
37.3.1	Bitonic Merge Sort	627
37.3.2	Binary Search	630
37.4	Challenges	633
37.4.1	Limited Outputs	633
37.4.2	Slow Readback	633
37.4.3	GPU vs. CPU	634
37.5	Conclusion	634
37.6	Further Reading	635

## Chapter 38

### Fast Fluid Dynamics Simulation on the GPU . . . . . 637

*Mark J. Harris, University of North Carolina at Chapel Hill*

38.1	Introduction . . . . .	637
38.1.1	Our Goal . . . . .	638
38.1.2	Our Assumptions . . . . .	638
38.1.3	Our Approach . . . . .	639
38.2	Mathematical Background . . . . .	639
38.2.1	The Navier-Stokes Equations for Incompressible Flow . . . . .	641
38.2.2	Terms in the Navier-Stokes Equations . . . . .	642
38.2.3	A Brief Review of Vector Calculus . . . . .	642
38.2.4	Solving the Navier-Stokes Equations . . . . .	644
38.3	Implementation . . . . .	650
38.3.1	CPU–GPU Analogies . . . . .	651
38.3.2	Slab Operations . . . . .	653
38.3.3	Implementation in Fragment Programs . . . . .	653
38.4	Applications . . . . .	659
38.4.1	Simulating Liquids and Gases . . . . .	660
38.4.2	Buoyancy and Convection . . . . .	660
38.5	Extensions . . . . .	661
38.5.1	Vorticity Confinement . . . . .	662
38.5.2	Three Dimensions . . . . .	663
38.5.3	Staggered Grid . . . . .	663
38.5.4	Arbitrary Boundaries . . . . .	664
38.5.5	Free Surface Flow . . . . .	664
38.6	Conclusion . . . . .	664
38.7	References . . . . .	665

## Chapter 39

### Volume Rendering Techniques . . . . . 667

*Milan Ikits, University of Utah*

*Joe Kniss, University of Utah*

*Aaron Lefohn, University of California, Davis*

*Charles Hansen, University of Utah*

39.1	Introduction . . . . .	667
39.2	Volume Rendering . . . . .	668
39.3	Texture-Based Volume Rendering . . . . .	670
39.3.1	A Simple Example . . . . .	673

39.4	Implementation Details . . . . .	674
39.4.1	Data Representation and Processing . . . . .	674
39.4.2	Proxy Geometry . . . . .	676
39.4.3	Rendering . . . . .	678
39.5	Advanced Techniques . . . . .	681
39.5.1	Volumetric Lighting . . . . .	682
39.5.2	Procedural Rendering . . . . .	687
39.6	Performance Considerations . . . . .	688
39.6.1	Rasterization Bottlenecks . . . . .	688
39.7	Summary . . . . .	690
39.8	References . . . . .	690

## Chapter 40

### Applying Real-Time Shading to 3D Ultrasound Visualization . . . . . 693

*Thilaka Sumanaweera, Siemens Medical Solutions USA, Inc.*

40.1	Background . . . . .	693
40.2	Introduction . . . . .	696
40.2.1	Volume Rendering Data in a Cartesian Grid . . . . .	696
40.2.2	Volume Rendering Data in Pyramidal Grids . . . . .	699
40.3	Results . . . . .	706
40.4	Conclusion . . . . .	706
40.5	References . . . . .	707

## Chapter 41

### Real-Time Stereograms . . . . . 709

*Fabio Policarpo, Paralelo Computação Ltda.*

41.1	What Is a Stereogram? . . . . .	709
41.1.1	Stereo Photography . . . . .	709
41.1.2	Random-Dot Stereograms . . . . .	710
41.1.3	Single-Image Stereograms . . . . .	711
41.2	Creating a Single-Image Stereogram . . . . .	713
41.2.1	Parameters . . . . .	713
41.2.2	Rendering . . . . .	715
41.2.3	Creating Animated Single-Image Stereograms . . . . .	717
41.2.4	Fragment Program . . . . .	719
41.3	Sample Application . . . . .	721
41.3.1	Application Options . . . . .	721
41.4	References . . . . .	722

## Chapter 42

### Deformers ..... 723

*Eugene d'Eon, University of Waterloo*

42.1 What Is a Deformer? ..... 723

42.1.1 Mathematical Definition ..... 724

42.2 Deforming on the GPU ..... 725

42.2.1 Formulating the Deformer ..... 725

42.2.2 Writing the Vertex Program ..... 725

42.2.3 Deforming Normals ..... 726

42.3 Limitations ..... 729

42.4 Performance ..... 729

42.5 Example: Wave Deformer ..... 730

42.6 Conclusion ..... 732

Index ..... 733