

---

# Contents

<b>Introduction</b>	<b>xiii</b>
<b>Contributor Biographies</b>	<b>xv</b>
<b>About the Editor</b>	<b>xxi</b>
<b>Part I Game Engine Design</b>	<b>1</b>
<hr/>	
<b>Chapter 1 What to Look for When Evaluating Middleware for Integration</b>	<b>3</b>
<i>Jason Hughes</i>	
1.1 Middleware, How Do I Love Thee?	3
1.2 Integration Complexity and Modularity	3
1.3 Memory Management	4
1.4 Mass Storage I/O Access	5
1.5 Logging	5
1.6 Error Handling	6
1.7 Stability and Performance Consistency	6
1.8 Custom Profiling Tools	7
1.9 Customer Support	7
1.10 Demands on the Maintainers	7
1.11 Source Code Availability	8
1.12 Quality of Source Code	8

1.13 Platform Portability	8
1.14 Licensing Requirements	9
1.15 Cost	10
<b>Chapter 2 The Game Asset Pipeline</b>	<b>11</b>
<i>Rémi Arnaud</i>	
2.1 Asset Pipeline Overview	11
2.2 Asset Pipeline Design	18
2.3 Push or Pull Pipeline Model	20
2.4 COLLADA, A Standard Intermediate Language	22
2.5 OpenCOLLADA	30
2.6 User Content	33
<b>Chapter 3 Volumetric Representation of Virtual Environments</b>	<b>39</b>
<i>David Williams</i>	
3.1 Introduction	39
3.2 Overview	40
3.3 Data Structures	42
3.4 Surface Extraction	46
3.5 Rendering	52
3.6 Physics	56
3.7 The Future	57
<b>Chapter 4 High-Level Pathfinding</b>	<b>61</b>
<i>Daniel Higgins</i>	
4.1 Terms	62
4.2 Start Your Engines	62
4.3 Why High-Level Pathfinding?	63
4.4 Preprocess Phase	64
4.5 Fuzzy Pathing Phase	71
4.6 Detailed Paths Phase	75
4.7 Why Go Through All This Trouble?	76

<b>Chapter 5</b>	<b>Environment Sound Culling</b>	<b>79</b>
<i>Simon Franco</i>		
5.1	The Problem	79
5.2	A Sound Culling Solution	80
5.3	Constructing the Sound Grid	83
5.4	Processing the Sound Grid	84
5.5	Supporting Multiple Listeners	90
5.6	Extensions	90
<b>Chapter 6</b>	<b>A GUI Framework and Presentation Layer</b>	<b>91</b>
<i>Adrian Hirst</i>		
6.1	GUI Systems	91
6.2	Design Patterns: Model View Controller (MVC)	92
6.3	A GUI Design	93
6.4	And Finally	104
<b>Chapter 7</b>	<b>World's Best Palettizer</b>	<b>107</b>
<i>Jason Hughes</i>		
7.1	Palettes? Whatever for?	107
7.2	Understanding Quantization	108
7.3	Hard-Earned Lessons	109
7.4	Algorithm Overview	111
7.5	Future Work	120
7.6	Results	120
<b>Chapter 8</b>	<b>3D Stereoscopic Rendering: An Overview of Implementation Issues</b>	<b>123</b>
<i>Anders Hast</i>		
8.1	Mechanisms of Plano-stereoscopic Viewing	124
8.2	Stereo Techniques	131
8.3	Design Considerations for 3D Scenes	133
8.4	Outlook	135

<b>Chapter 9</b>	<b>A Multithreaded 3D Renderer</b>	<b>139</b>
	<i>Sebastien Schertenleib</i>	
9.1	The Memory Model	140
9.2	Building the Display Lists in Parallel	141
9.3	Parallel Models	143
9.4	Synchronizing the GPU and CPU	144
9.5	Using Additional Processing Resources	145
9.6	Reducing the Pressure on the Memory Bandwidth	145
9.7	Performing Graphical Operations in Parallel	146
<b>Chapter 10</b>	<b>Camera-Centric Engine Design for Multithreaded Rendering</b>	<b>149</b>
	<i>Colt McAnlis</i>	
10.1	Uses of Multi-core in Video Games	150
10.2	Multithreaded Command Buffers	151
10.3	Device-Independent Command Buffers	152
10.4	A Camera-Centric Design	157
10.5	Future Work	164
<b>Chapter 11</b>	<b>A GPU-Managed Memory Pool</b>	<b>167</b>
	<i>Jeremy Moore</i>	
11.1	Background	168
11.2	The Memory Pool	169
11.3	Synchronization Issues	170
11.4	The Staging Buffer	172
11.5	Memory Pool Defragmentation	173
11.6	Memory Pool Eviction	174
11.7	Platform-Specific Considerations	174
11.8	Future Work	175
<b>Chapter 12</b>	<b>Precomputed 3D Velocity Field for Simulating Fluid Dynamics</b>	<b>177</b>
	<i>Khalid Djado and Richard Egli</i>	
12.1	Introduction	177

12.2 Velocity Field Computation	178
12.3 Physics Simplification	181
12.4 Results and Discussion	184
<b>Chapter 13 Mesh Partitioning for Fun and Profit</b>	<b>187</b>
<i>Jason Hughes</i>	
13.1 Desirable Algorithm Properties	187
13.2 Lessons Learned	189
13.3 When Greedy Is Good	191
13.4 Future Work	193
13.5 Graphical Walkthrough	194
<b>Chapter 14 Moments of Inertia for Common Shapes</b>	<b>197</b>
<i>Eric Lengyel</i>	
14.1 Center of Mass	197
14.2 The Inertia Tensor	198
14.3 Derivation of Moments of Inertia	201
14.4 Summary	215
<b>Part II Rendering Techniques</b>	<b>217</b>
<hr/>	
<b>Chapter 15 Physically-Based Outdoor Scene Lighting</b>	<b>219</b>
<i>Frank Kane</i>	
15.1 Positioning the Sun and Moon	219
15.2 Computing Natural Sunlight	221
15.3 Moonlight and Other Nighttime Light Sources	223
15.4 Tone-Mapping the Light	224
15.5 Implementation Notes	226
<b>Chapter 16 Rendering Physically-Based Skyboxes</b>	<b>229</b>
<i>Frank Kane</i>	

16.1	Generating and Drawing the Skybox	229
16.2	Computing the Skybox Vertex Colors	231
16.3	Integrating the Skybox with Your Scene	233
16.4	Embellishing Your Skybox	234
<b>Chapter 17</b>	<b>Motion Blur and the Velocity-Depth-Gradient Buffer</b>	<b>235</b>
	<i>Eric Lengyel</i>	
17.1	Technique Overview	236
17.2	Rendering to the Velocity-Depth-Gradient Buffer	238
17.3	Rendering the Post-Processing Effect	242
17.4	Grid Optimization	245
<b>Chapter 18</b>	<b>Fast Screen-space Ambient Occlusion and Indirect Lighting</b>	<b>249</b>
	<i>László Szirmay-Kalos, Balázs Tóth, and Tamás Umenhoffer</i>	
18.1	Introduction	249
18.2	A General Ambient Illumination Model	250
18.3	Screen-space Representation of the Scene	252
18.4	Volumetric Ambient Occlusion	253
18.5	Indirect Lighting of the Near Geometry	257
18.6	Implementation	257
18.7	Results	260
<b>Chapter 19</b>	<b>Real-Time Character Dismemberment</b>	<b>263</b>
	<i>Aurelio Reis</i>	
19.1	What is Character Damage Modeling?	263
19.2	Methods of Mutilation	264
19.3	Bone Matrix Flattening	265
19.4	Improvements	266
19.5	Demo	269
<b>Chapter 20</b>	<b>A Deferred Decal Rendering Technique</b>	<b>271</b>
	<i>Jan Krassnigg</i>	
20.1	The Problem	272

20.2 The General Idea	272
20.3 Geometry Rendering	273
20.4 Fade Out And Wrap-Around	275
20.5 Surface Clipping	277
20.6 Limitations	279
20.7 Additional Features	279

---

## **Part III Programming Methods** **281**

### **Chapter 21 Multithreaded Object Models** **283**

*Jon Parise*

21.1 Explicit Locking	283
21.2 Message-Based Updates	284
21.3 Multiple Thread Contexts	285
21.4 Buffered State Changes	286
21.5 Selecting the Best Approach	287

### **Chapter 22 Holistic Task Parallelism for Common Game Architecture Patterns** **289**

*Brad Werth*

22.1 Tasks Versus Threads in Games	289
22.2 The Task Scheduler	290
22.3 Decomposing Game Patterns into Tasks	291
22.4 The Future of Task Parallelism in Games	296

### **Chapter 23 Dynamic Code Execution Hierarchies** **297**

*Martin Linklater*

23.1 What are Code Execution Hierarchies?	297
23.2 Design Features	300
23.3 Benefits & Pitfalls	303

<b>Chapter 24 Key-Value Dictionary</b>	<b>305</b>
<i>Martin Linklater</i>	
24.1 Design	305
24.2 Using the KVD	306
24.3 Code Details	307
24.4 Caveats	310
<b>Chapter 25 A Basic Scheduler</b>	<b>311</b>
<i>John Bolton</i>	
25.1 Overview	311
25.2 Task Functionality	311
25.3 Scheduler Functionality	312
25.4 Implementation	313
25.5 Additional Functionality	314
<b>Chapter 26 The Game State Observer Pattern</b>	<b>315</b>
<i>Ron Barbosa</i>	
26.1 Creating a Game State Manager	317
26.2 The Interfaces of the Game State Observer Pattern	321
26.3 Making GameState Observable	322
26.4 Creating Observers	325
26.5 Managing Functionality by Game State	327
<b>Chapter 27 Fast Trigonometric Operations Using CORDIC Methods</b>	<b>329</b>
<i>John Bolton</i>	
27.1 Rotation Mode Algorithm	329
27.2 Vectoring Mode Algorithm	331
27.3 Applications	332
27.4 Implementation	332
27.5 Considerations	337
27.6 Extensions	337

**Chapter 28 Inter-Process Communication Based on Your Own RPC Subsystem 339**

*Kurt Pelzer*

28.1 History of Remote Procedure Call	340
28.2 How RPC Works: Internal Architecture of RPC	341
28.3 How to Build Your Own RPC Subsystem	343
28.4 Why RPC is Useful for Game Engines	347

---

**About the DVD 351**

**Index 353**