

# **SHADERX<sup>7</sup>: ADVANCED RENDERING TECHNIQUES**

**WOLFGANG ENGEL**

**Charles River Media**

*A part of Course Technology, Cengage Learning*



**COURSE TECHNOLOGY**  
CENGAGE Learning™

---

Australia, Brazil, Japan, Korea, Mexico, Singapore, Spain, United Kingdom, United States



# Contents

<b>Part I</b>	<b>Geometry Manipulation</b>	<b>1</b>
1.1	<b>Scalar to Polygonal: Extracting Isosurfaces Using Geometry Shaders</b>	<b>3</b>
	Scalar Fields Versus Polygonal Representation	3
	Isosurface Extraction Using Marching Methods	6
	Hybrid Cubes/Tetrahedra Extraction	7
	Isosurface Extraction Results and Analysis	29
1.2	<b>Fast High-Quality Rendering with Real-Time Tessellation on GPUs</b>	<b>33</b>
	GPU Tessellation Pipeline	37
	Programming for GPU Tessellation	41
	Continuous Tessellation Mode	45
	Adaptive Tessellation Mode	48
	Rendering Characters with Tessellation	51
	Designing a Vertex Evaluation Shader for GPU Tessellation	53
	Accessing Per-Vertex Data Beyond Input Structure Declaration	59
	Tessellation API in Direct3D 10	60
	Lighting with Tessellation and Displacement Mapping	60
	Rendering Animated, Tessellated Characters	61
	Displacement Map Tips and Reducing Surface Cracks	68
1.3	<b>Dynamic Terrain Rendering on GPUs Using Real-Time Tessellation</b>	<b>73</b>
	Programming for Adaptive GPU Tessellation	76
	Transforming Mesh Using R2VB	79
	Computing Per-Edge Tessellation Factors	83
	Rendering Tessellated Mesh with Per-Edge Tessellation Factors	89
	Shading Tessellated Displaced Surfaces	93
	Performance Analysis	103
1.4	<b>Adaptive Re-Meshing for Displacement Mapping</b>	<b>107</b>
	Introduction	107
	Displacement Map	108
	Adaptive Re-Meshing for Displacement Mapping	109
	LOD	112
	Results	114

	Implementation	114
	Demo	116
	Conclusion	116
	Endnotes	116
<b>1.5</b>	<b>Fast Tessellation of Quadrilateral Patches for Dynamic Levels of Detail</b>	<b>119</b>
	Introduction	119
	The Method	120
	How Many Strips Will Be Needed?	123
	Where Will Each Strip Be Located?	124
	How Do We Tessellate Each Strip?	124
	A 3D Interpolation to Place Triangle Strips	124
	A 2D Interpolation to Tessellate Strips	126
	Results	127
	Discussion	128
<b>Part II</b>	<b>Rendering Techniques</b>	<b>131</b>
<b>2.1</b>	<b>Quick Noise for GPUs</b>	<b>133</b>
	Introduction	133
	Background	133
	Math to the Rescue	134
	Applying It in the Real World	135
	And Now the Bad News	137
	Implementation, the Sequel	138
	Results	140
	Future Work	140
<b>2.2</b>	<b>Efficient Soft Particles</b>	<b>143</b>
	Introduction	143
	Hard Particles vs. Soft Particles	144
	Implementing Soft Particles the Standard Way	145
	Optimizing Soft Particles	145
	Results	146
	Conclusion	147
<b>2.3</b>	<b>Simplified High-Quality Anti-Aliased Lines</b>	<b>149</b>
	Abstract	149
	Introduction	149
	Method	150

	Texture Creation	150
	Vertex Setup	152
	Variations on the Theme	154
	Conclusion	155
	Appendix A: The Shader Code	155
<b>2.4</b>	<b>Fast Skin Shading</b>	<b>161</b>
	Introduction	161
	Background and Existing Art	162
	Specular and Diffuse	163
	Data Preparation	171
	Conclusion	172
<b>2.5</b>	<b>An Efficient and Physically Plausible Real-Time Shading Model</b>	<b>175</b>
	Introduction	175
	Review: Blinn-Phong and Cook-Torrance	176
	Some Physics of Light-Surface Interaction	178
	Toward an Improved Shading Model	179
	Mathematical Formulation	180
	Appendix	185
<b>2.6</b>	<b>Graphics Techniques in Crackdown</b>	<b>189</b>
	Introduction	189
	Sky	190
	Implementation Notes	192
	Clutter	193
	Outlines	198
	Deferred Rendering	201
	Vehicle Reflections	205
	Implementation Notes	208
	Texture Map Setup	209
	Conclusion	214
	Endnotes	214
<b>2.7</b>	<b>Deferred Rendering Transparency</b>	<b>217</b>
	Introduction	217
	Transparency	218
	Overview	219
	Rendering	221
	Results Discussion	223
	Summary and Future Work	224

<b>2.8</b>	<b>Deferred Shading with Multisampling Anti-Aliasing in DirectX 10</b>	<b>225</b>
	Introduction	225
	Deferred Shading Principles	226
	MSAA Requirements for Deferred Shading	227
	Implementation	231
	Assessment of Alternative Implementation	240
	Conclusion	242
<b>2.9</b>	<b>Light-Indexed Deferred Rendering</b>	<b>243</b>
	Introduction	243
	Rendering Concept	243
	Light-Indexed Deferred Rendering	244
	Combining with Other Rendering Techniques	250
	Multi-Sample Anti-Aliasing	250
	Transparency	252
	Shadows	252
	Constraining Lights to Surfaces	253
	Multi-Light Type Support	254
	Lighting Technique Comparison	254
	Future Work	255
	Conclusion	255
<b>Part III</b>	<b>Image Space</b>	<b>257</b>
<b>3.1</b>	<b>Efficient Post-Processing with Importance Sampling</b>	<b>259</b>
	Introduction	259
	Problem Statement	259
	The Approach of Importance Sampling	260
	Tone Mapping with Glow	263
	Depth of Field	266
	Comparisons to Uniform Sampling	275
	Conclusion	276
<b>3.2</b>	<b>Efficient Real-Time Motion Blur for Multiple Rigid Objects</b>	<b>277</b>
	Introduction	277
	Overview	278
	CPU-Side Work	278
	GPU-Side Work	278
	Blurring and Halo Fixing	279
	Integration with a Post-Processing Pipeline	282
	Coping with Hardware Limitations	282
	Conclusion	282

<b>3.3</b>	<b>Real-Time Image Abstraction by Directed Filtering</b>	<b>285</b>
	Introduction	285
	Color Space Conversion	287
	Flow Field Construction	287
	Orientation-Aligned Bilateral Filter	291
	Separable Flow-Based Difference-of-Gaussians	295
	Color Quantization	301
	Conclusions	302

## **Part IV    Shadows    303**

<b>4.1</b>	<b>Practical Cascaded Shadow Maps</b>	<b>305</b>
	Introduction	305
	Flickering of Shadow Quality	307
	Exact Solution	309
	Approximated Solution	311
	Storage Strategy	313
	Non-Optimized Split Selection	315
	Correct Computation of Texture Coordinates	317
	Filtering Across Splits	321
	Method Used in PSVSMs	324
	Analytic Method	325
	Conclusion	328
<b>4.2</b>	<b>A Hybrid Method for Interactive Shadows in Homogeneous Media</b>	<b>331</b>
	Introduction	331
	Participating Media Review	332
	Hybrid Approach	335
	Adding Textured Light Sources	336
	Implementation Details	338
	Results	342
	Conclusions	342
<b>4.3</b>	<b>Real-Time Dynamic Shadows for Image-Based Lighting</b>	<b>345</b>
	Introduction	345
	Related Work	346
	Algorithm Overview	346
	Environment Map Importance Sampling	346
	Visibility Map Generation	349
	Rendering Shadows on Diffuse Surfaces	357
	Rendering Shadows on Glossy Surfaces	358
	Results	359

	Conclusion	361
	Endnotes	361
<b>4.4</b>	<b>Facetted Shadow Mapping for Large Dynamic Game Environments</b>	<b>363</b>
	Introduction	363
	The Challenges	363
	Existing Shadow Map Approaches	364
	Facetted Shadow Map Approach	365
	Creating and Using Facetted Shadow Maps	367
	Results	370
	Conclusion	370
<b>Part V</b>	<b>Environmental Effects</b>	<b>373</b>
<b>5.1</b>	<b>Dynamic Weather Effects</b>	<b>375</b>
	Introduction	375
	Particle Simulation and Rendering	376
	Rendering Motion-Blurred Particles	378
	Occlusion	382
	Dynamic, Artist-Controlled Weather	384
	Additional Effects	385
	Conclusion	386
<b>5.2</b>	<b>Interactive Hydraulic Erosion on the GPU</b>	<b>389</b>
	Introduction	389
	Data Structures	390
	Water Movement	392
	Erosion	394
	Boundaries	400
	Rendering	401
	Results and Conclusion	402
<b>5.3</b>	<b>Advanced Geometry for Complex Sky Representation</b>	<b>405</b>
	Introduction	405
	Geometry Generation	406
	Conclusion	408

<b>Part VI</b>	<b>Global Illumination Effects</b>	<b>411</b>
6.1	Screen-Space Ambient Occlusion	413
	Introduction	413
	The Problems	414
	Previous Work	414
	Overview of the Approach	415
	Implementation	420
	Future Improvements	423
	Results and Conclusion	423
6.2	Image-Space Horizon-Based Ambient Occlusion	425
	Introduction	425
	Input Buffers for Image Space Ambient Occlusion	429
	Image Space Ambient Occlusion with Ray Marching	429
	Our Algorithm	431
	Reformulating the Ambient Occlusion Integral	431
	Implementation Considerations	435
	Results	440
6.3	Deferred Occlusion from Analytic Surfaces	445
	Introduction	445
	Method	446
	Analytic Occlusion from other Surfaces	451
	Optimization	451
	Conclusion	453
6.4	Fast Fake Global Illumination	455
	Introduction	455
	Ambient Occlusion Probes	455
	Screen-Space Ambient Occlusion	458
	Screen-Space Radiosity	462
	Fake Radiosity	462
	Conclusion	466
6.5	Real-Time Subsurface Scattering Using Shadow Maps	467
	Introduction	467
	Related Work	468
	Theory	469
	Algorithm	471
	Results	476
	Conclusion	477



<b>6.6</b>	<b>Instant Radiosity with GPU Photon Tracing and Approximate Indirect Shadows</b>	<b>479</b>
	Introduction	479
	Techniques We Build On	480
	Algorithm Overview	481
	Scene Representation for Ray Tracing	483
	Ray-Triangle Intersection	484
	BIH Traversal	487
	Light Source Sampling	489
	Photon Shooting	489
	VPL Management	490
	Rendering the Distance Impostor Cube Map	490
	Rendering Deferring Textures	490
	Building Pyramidal Occlusion Maps	490
	Lighting	491
	Adaptive Geometry-Sensitive Box Filtering	493
	Performance	493
	Conclusion	494
<b>6.7</b>	<b>Variance Methods for Screen-Space Ambient Occlusion</b>	<b>495</b>
	Ambient Lighting	495
	Ambient Occlusion	496
<b>6.8</b>	<b>Per-Pixel Ambient Occlusion Using Geometry Shaders</b>	<b>501</b>
	Introduction	501
	Background	502
	Our Approach	502
	Results	506
	Conclusion	509
<b>Part VII</b>	<b>Handheld Devices</b>	<b>511</b>
<b>7.1</b>	<b>Optimizing Your First OpenGL ES Application</b>	<b>513</b>
	Introduction	513
	Mobile Development	514
	Graphics Development Guidelines	518
	A Developer's Experience: Insight from Jadestone into KODO Evolved	539
	Conclusion	541

<b>7.2</b>	<b>Optimized Shaders for Advanced Graphical User Interfaces</b>	<b>543</b>
	Introduction	543
	Handheld GUI Requirements	543
	Optimizing for Power Consumption	544
	Optimizing Blurs	546
	Optimizing Other Popular Effects	549
	Background and Post-Processing	552
	Transitions	555
	Conclusion	560
<b>7.3</b>	<b>Facial Animation for Mobile GPUs</b>	<b>561</b>
	Introduction	561
	Facial Animation Components	561
	Current Approaches and Efficiency	562
	Mobile Approach: Maximal Efficiency Is Critical	563
	Conclusion	569
<b>7.4</b>	<b>Augmented Reality on Mobile Phones</b>	<b>571</b>
	Introduction	571
	Developing Augmented Reality Applications	572
	Platform Considerations	581
	Application Initialization	583
	Graphics API Abstraction	586
	Hardware vs. Software Rendering	596
	Scene-Graph Rendering	599
	Video and Image Processing	601
	Fixed Point vs. Floating Point	603
	Conclusion	604
<b>Part VIII</b>	<b>3D Engine Design Overview</b>	<b>605</b>
<b>8.1</b>	<b>Cross-Platform Rendering Thread: Design and Implementation</b>	<b>607</b>
	Motivation	607
	Overview	607
	Implementation	612
	Results	617
	Going Further: Add-Ons and Features	618
	Conclusion	620

<b>8.2</b>	<b>Advanced GUI System for Games</b>	<b>621</b>
	Introduction	621
	Architecture	622
	Rendering	624
	Conclusion	625
<b>8.3</b>	<b>Automatic Load-Balancing Shader Framework</b>	<b>627</b>
	Introduction	627
	The Problems	627
	User Inconvenience	628
	Performance	629
	The Approach	629
	Workflow	632
	Discussion of Results	632
	GPU Requirements	633
	Conclusion	633
<b>8.4</b>	<b>Game-Engine-Friendly Occlusion Culling</b>	<b>637</b>
	Introduction	637
	Coherent Hierarchical Culling	638
	Reducing State Changes	641
	Game Engine Integration	643
	Skipping Tests for Visible Nodes	645
	Further Optimizations	646
	Multiqueries	647
	Putting It All Together	650
	Conclusion	653
<b>8.5</b>	<b>Designing a Renderer for Multiple Lights: The Light Pre-Pass Renderer</b>	<b>655</b>
	Z Pre-Pass Renderer	655
	Deferred Renderer	657
	Light Pre-Pass Renderer	660
	Storing an Additional Diffuse Term	662
	Converting the Diffuse Term to Luminance	663
	Bending the Specular Reflection Rules	663
	Comparison and Conclusion	664
	Appendix: Applying Different Materials with a Light Pre-Pass Renderer	665

<b>8.6</b>	<b>Light Pre-Pass Renderer: Using the CIE Luv Color Space</b>	<b>667</b>
	Introduction	667
	Why CIE Luv?	668
	Working with Luv Colors	668
	Luv Light Buffer Format	671
	Grouping and Rendering Lights	672
	Integrating Luv into Light Accumulation	675
	Conclusion	676
<b>8.7</b>	<b>Elemental Engine II</b>	<b>679</b>
<b>Part IX</b>	<b>Beyond Pixels and Triangles</b>	<b>683</b>
<b>9.1</b>	<b>Sliced Grid: A Memory and Computationally Efficient Data Structure for Particle-Based Simulation on the GPU</b>	<b>685</b>
	Introduction	685
	Sliced Grid	686
	Implementing a Sliced Grid on the GPU	689
	Results	693
	Conclusion	697
<b>9.2</b>	<b>Free-Viewpoint Video on the GPU</b>	<b>699</b>
	Introduction	699
	Background Subtraction	700
	Shape from Silhouette	703
	Surface Extraction	705
	Texture Mapping	709
	Conclusion	713
<b>9.3</b>	<b>A Volume Shader for Quantum Voronoi Diagrams Inside the 3D Bloch Ball</b>	<b>715</b>
	Introduction and Preliminaries	715
	Von Neumann Quantum Entropy and Its Relative Entropy Divergence	716
	Quantum Voronoi Diagrams	717
	Quantum Voronoi Diagrams for Pure States	719
	Quantum Voronoi Diagrams for Mixed States	723
	Quantum Channel and Holevo's Capacity	728
	Concluding Remarks	729

<b>9.4</b>	<b>Packing Arbitrary Bit Fields into 16-Bit Floating-Point Render Targets in DirectX 10</b>	<b>731</b>
	Introduction	731
	16-Bit and 32-Bit Floating-Point Formats	732
	Writing a Valid 32-Bit Floating-Point Output	733
	Converting Between Single- and Half-Precision	733
	Packing and Unpacking Code	736
	Performance Considerations	741
	Conclusion	741
<b>9.5</b>	<b>Interactive Image Morphing Using Thin-Plate Spline</b>	<b>743</b>
	Introduction	743
	Thin-Plate Spline-Based Warping	744
	GPU Implementation of TPS Warping	747
	Interactive Image Morphing	750
	Conclusion	752
	<b>Index</b>	<b>753</b>