

Table of Contents

1. Introduction	1
1.1 A visual hierarchy: color, light, shape, space, image	3
1.2 What you need to know to write mental ray shaders	5
1.3 How to read this book	6
1.4 Web-based resources	7
1.5 Acknowledgments	7
 Part 1: Structure	 9
2. The structure of the scene	11
2.1 The scene in mental ray	11
2.2 A block of statements with a name	11
2.3 A block that includes another block	12
2.4 A block containing a shader	13
2.5 A block that refers to another block	14
2.6 Grouping the elements in the scene	16
2.7 Scene file commands	16
2.8 Putting the parts together	17
2.9 A complete scene file	18
2.10 The scene file and 3D applications	19
 3. The structure of a shader	 21
3.1 Defining a color with a shader	21
3.2 Providing input to a shader	22
3.3 Telling the shader about its context	23
3.4 Accumulating the effect of a series of shaders	23
3.5 Combining the previous and current results	24
 4. Shaders in the scene	 25
4.1 Describing parameters: shader declarations	25
4.2 Missing parameter values: defining defaults	27
4.3 Defining the material: anonymous and named shaders	29
4.3.1 Anonymous shaders	29
4.3.2 Named shaders	30
4.4 Chaining shaders of the same result type: shader lists	30
4.4.1 Using the result of the previous shader	31
4.4.2 Modifying the rendering state in a shader list	32

4.4.3 Named and anonymous shaders in the shader list 35

4.5 Connecting shaders in a network: shader graphs 37

4.6 Named shader graphs: Phenomena 41

4.6.1 The components of a Phenomenon 42

4.6.2 Defining a material as a Phenomenon 44

4.7 The five shader usage types 46

Part 2: Color 47

5. A single color 49

5.1 The simplest shader model 49

5.2 The shader function in C 50

5.3 The shader source file 51

5.4 Using a shader in a material 54

5.5 Shader programming style 54

5.6 Basic issues in writing shaders 55

6. Color from orientation 57

6.1 The representation of rendering state: miState 57

6.2 A shader based on surface orientation 58

6.3 Passing parameters to a shader 59

6.4 Other fields in miState 59

6.5 Calculation of surface orientation 61

6.6 Shaders as analysis tools 61

7. Color from position 63

7.1 Intensity based on distance 63

7.2 Interpolating between colors based on distance 64

7.3 Clarifying the shader with auxiliary functions 66

8. The transparency of a surface 69

8.1 Using the API library to trace a ray 69

8.2 Functions as a description of a process 73

9. Color from functions 77

9.1 The texture coordinate system 78

9.2 Quantizing the texture coordinates 80

9.3 Using the texture coordinates for texture mapping 82

9.4 Manipulating the texture coordinates 84

9.5 Defining texture maps that tile well 86

9.6 Multiple texture spaces 87

9.7 Noise functions 89

10. The color of edges 93

10.1 The four contour shader types 94

10.2 Location of the contour shaders in the scene file 95

10.3 The Store shader: defining and saving contour data 96

10.4 The Contrast shader: determining the location of contours 98

10.5 The Contour shader: defining the properties of contours 100

10.6 The Output shader: writing the contour image 102

10.7 Using the four contour shaders 104

10.8 Compositing contours with the color from the material shader 104

10.9 Displaying tessellation with contour shaders 106

10.9.1 The barycentric coordinates of a triangle 109

10.10 Including color in the determination of contour lines 112

10.10.1 Contour lines at edges 112

10.10.2 Contour lines at color boundaries 113

10.10.3 Converting illumination shading into regions of single color 118

Part 3: Light 125

11. Lights 127

11.1 Light from a single point 127

11.2 A point light with shadows 130

11.3 A simple spotlight 131

11.3.1 Acquiring light parameter values from the scene database 134

11.3.2 Using light parameter values in the shader 137

11.4 A spotlight with a soft edge 139

11.5 A spotlight with a better soft edge 141

11.6 Soft shadows using area lights 143

11.6.1 Area light primitives 145

11.6.2 Using area lights in spotlights 146

11.7 Modifying light intensity based on distance 146

11.8 Defining good default values for parameters 148

12. Light on a surface 149

12.1 Diffuse reflection: Lambert shading 149

12.2 Specular reflections 154

12.2.1 The Phong specular model 154

12.2.2 The Blinn specular model 156

12.2.3 The Cook-Torrance specular model 158

12.2.4 The Ward specular model	160
12.3 Faking the specular component	162
12.4 Adding arbitrary effects to the Lambert model	164
13. Shadows	167
13.1 Color shadows without full transparency	168
13.2 Midrange transparency control	171
13.3 Other parameters for the breakpoint function	174
13.4 A shadow shader with continuous transparency change	176
13.5 Combining the material and shadow shader	179
14. Reflection	183
14.1 Specular reflections	183
14.2 Reflections and trace depth	185
14.3 Glossy reflections	187
14.4 Glossy reflection with multiple samples in the shader	189
14.5 Glossy reflection with varying sample counts for reflections	192
14.6 Glossy reflections and object geometry	195
15. Refraction	197
15.1 Specular refraction of non-intersecting objects	198
15.2 Improving the determination of the indices of refraction	201
15.3 Using different indices of refraction	204
15.4 Glossy refraction	206
15.4.1 Using a single glossy refraction ray	206
15.4.2 Using multiple glossy refraction rays	207
15.4.3 Controlling the number of rays per refraction	209
15.5 Combining reflection and refraction with Phenomena	212
16. Light from other surfaces	219
16.1 Using the photon map for global illumination	220
16.1.1 Adding global illumination to the Lambert shader	221
16.1.2 The photon shader	222
16.1.3 Global illumination statements in the options block	224
16.2 Global illumination as the ambient parameter	224
16.2.1 Overriding global illumination options in a shader	226
16.3 Final gathering	230
16.4 Caustics	232
16.5 Visualizing the photon map	235
16.6 Ambient occlusion	238
16.6.1 Sampling the environment by tracing rays to detect occlusion	239

16.6.2 Restricting ray length in occlusion detection	241
Part 4: Shape	245
17. Modifying surface geometry	247
17.1 Surface position and the normal vector	247
17.2 A simple displacement shader	248
17.2.1 Using texture coordinates	250
17.3 Shader lists and displacement mapping	252
17.4 Using texture images to control displacement mapping	253
17.5 Combining displacement mapping with color	255
17.6 Using noise functions with displacement mapping	257
18. Modifying surface orientation	259
18.1 Simulating surface orientation	259
18.2 Modifying the normal in a shader	260
18.3 Changing the normal based on texture coordinates	261
18.4 Evaluating a function in the sample neighborhood	263
18.5 Shader lists and bump mapping	266
18.6 Using images to control bump mapping	266
18.7 Combining bump mapping with color	269
19. Creating geometric objects	271
19.1 Creating a square polygon	271
19.2 Procedural use of the geometry API	277
19.3 Placeholder objects	282
19.3.1 Creating an object from a file specified in the scene	282
19.3.2 Creating an object from a file within a shader	284
19.3.3 Creating an object instance from file within a shader	286
20. Modeling hair	289
20.1 Placeholder objects and callback functions	289
20.2 Structure of the shader and its callback	289
20.3 Basic principles of the hair geometric primitive	290
20.4 A geometry shader for a single hair	292
20.5 Visualizing the hair's barycentric coordinates	298
20.6 Higher order curves in the hair primitive	298
20.7 Additional data attached to the hair primitive	303
20.8 Multiple hairs	309
20.9 A basic lighting model for hair	315
20.10 The hair primitive as a general modeling tool	320

20.11 The hair primitive and particle systems	326
20.12 Particle system data and dynamic rendering effects	332
Part 5: Space	337
21. The environment of the scene	339
21.1 A single color for the environment	339
21.2 Defining a color ramp	340
21.3 Efficient shader access to a color ramp	342
21.4 Using parameter arrays for color channels in the ramp	345
21.4.1 Allocating memory in the init function	346
21.4.2 Releasing the memory allocated through the user pointer	349
21.4.3 Using channel_ramp in a named shader	350
21.5 A color array as a shader parameter	351
21.6 Environment shaders for cameras and objects	354
21.7 Encapsulating constant data in a Phenomenon	355
22. A visible atmosphere	357
22.1 Volume shaders and the camera	357
22.2 Changing the fog density	360
22.3 Adding a debugging mode to a shader	364
22.4 Varying the ground fog layer positions	367
23. Volumetric effects	369
23.1 Rays in volume shaders	369
23.2 Using a threshold value in the volume	371
23.3 Cumulative density for a ray in the volume	374
23.4 Illumination of samples in the volume	377
23.4.1 Fractional occlusion	379
23.4.2 Total light at a point in the volume	379
23.4.3 Accumulating and blending volume colors	380
23.4.4 The illuminated volume	381
23.5 Defining the density function with a shader	383
23.6 Using voxel data sets for the density function	389
23.7 Summary of the volume shaders	395
Part 6: Image	397
24. Changing the lens	399
24.1 A conceptual model of a camera	399
24.2 Shifting the lens position	401

24.3 Mapping the scene around the camera to a rectangle	403
24.4 A fisheye lens	407
24.5 Depth of field	412
24.6 Multiple lens shaders	418
25. Rendering image components	421
25.1 Definition and use of frame buffers	421
25.2 Separating illumination components into frame buffers	423
25.3 Compositing illumination components	426
25.4 Rendering shadows separately	427
25.5 Saving components from standard shaders	430
25.6 Using a material Phenomenon with framebuffer components	434
26. Modifying the final image	439
26.1 An image processing vocabulary	439
26.2 Adding a letterbox mask	440
26.3 A median filter in a shader	443
26.4 Compositing text over a rendered image	447
26.5 Multiple output shaders	451
26.6 Late binding and beyond	452
Appendices	455
A. Rendered scene files	457
Chapter 5 – A single color	457
Chapter 6 – Color from orientation	457
Chapter 7 – Color from position	457
Chapter 8 – The transparency of a surface	458
Chapter 9 – Color from functions	458
Chapter 10 – The color of edges	459
Chapter 11 – Lights	460
Chapter 12 – Light on a surface	461
Chapter 13 – Shadows	462
Chapter 14 – Reflection	463
Chapter 15 – Refraction	464
Chapter 16 – Light from other surfaces	465
Chapter 17 – Modifying surface geometry	466
Chapter 18 – Modifying surface orientation	467
Chapter 19 – Creating geometric objects	467
Chapter 20 – Modeling hair	468

Chapter 21 – The environment of the scene	469
Chapter 22 – A visible atmosphere	470
Chapter 23 – Volumetric effects	471
Chapter 24 – Changing the lens	471
Chapter 25 – Rendering image components	472
Chapter 26 – Modifying the final image	474
B. Shader source code	475
Shader definitions	476
The miaux utility library	585
Utility header file miaux.h	585
Utility source file miaux.c	589
C. Creating fontimage files	609
C.1 Python script fontimage.py	610
C.2 PostScript file fontimage_components.ps	612
Bibliography	617
References to Volumes I and II	619
Index	623



<http://www.springer.com/978-3-211-48964-2>

Writing mental ray® Shaders

A Perceptual Introduction

Kopra, A.

2008, XIV, 635 p. 171 illus. in color., Softcover

ISBN: 978-3-211-48964-2