



## Programming with OpenGL: Advanced Techniques

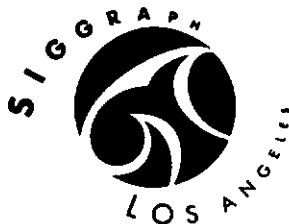
ORGANIZER

Tom McReynolds  
Silicon Graphics, Inc.

LECTURERS

David Blythe  
Celeste Fowle  
Brad Grantham  
Simon Hui  
Tom McReynolds  
Paula Womack  
Silicon Graphics, Inc.

Conference 3-8 August 1997  
Exhibition 5-7 August 1997



Los Angeles Convention Center  
Los Angeles, California USA

# Programming with OpenGL: Advanced Techniques

Organizer:  
Tom McReynolds  
Silicon Graphics

May 1, 1997

## SIGGRAPH '97 Course

### Abstract

This course moves beyond the straightforward images generated by the novice, demonstrating the more sophisticated and novel techniques possible using the OpenGL library.

By explaining the concepts and demonstrating the techniques required to generate images of greater realism and utility, the course helps students achieve two goals: they gain a deeper insight into OpenGL functionality and computer graphics concepts, while expanding their “toolbox” of useful OpenGL techniques.

## Speakers

### David Blythe

David Blythe is a Principal Engineer with the Advanced Systems Division at Silicon Graphics. David joined SGI in 1991 and has contributed to the development of RealityEngine and InfiniteReality graphics. He has contributed extensively to implementations of the OpenGL graphics library and OpenGL extension specifications.

Prior to joining SGI, David was a visualization scientist at the Ontario Centre for Large Scale Computation. David received both a B.S. and M.S. degree in computer science from the University of Toronto.

Email: [blythe@asd.sgi.com](mailto:blythe@asd.sgi.com)

### Celeste Fowler

Celeste Fowler is a software engineer in the Advanced Systems Division at Silicon Graphics. She worked on the OpenGL imaging pipeline for the InfiniteReality graphics system and on the OpenGL display list implementation for InfiniteReality and RealityEngine.

Before coming to SGI, Celeste attended Princeton University where she did research on radiosity techniques and TA'd courses in computer graphics and programming systems.

Email: [celeste@asd.sgi.com](mailto:celeste@asd.sgi.com)

### Brad Grantham

Brad Grantham currently contributes to the design and implementation of Silicon Graphics' high-level graphics toolkits, including OpenGL++, a scene graph toolkit for OpenGL. Brad previously worked on the Windows 95 port and Java bindings for Cosmo 3D and, before that, worked in the IRIS Performer group.

Before joining SGI, Brad wrote UNIX kernel code and imaging codecs. He received a B.S. from Virginia Tech in 1992, and his previous claim to fame is MacBSD, BSD UNIX for the Macintosh.

Email: [grantham@sgi.com](mailto:grantham@sgi.com)

### Simon Hui

Simon Hui is a software engineer at 3Dfx Interactive, Inc. He currently works on OpenGL and other graphics libraries for PC and consumer platforms.

Prior to joining 3Dfx, Simon worked on IRIS Performer, a realtime graphics

toolkit, in the Advanced Systems Division at Silicon Graphics. He has also worked on OpenGL implementations for the RealityEngine and InfiniteReality. Simon received a B.A. in Computer Science from the University of California at Berkeley.  
Email: [simon@3dfx.com](mailto:simon@3dfx.com)

### **Tom McReynolds**

Tom McReynolds is a software engineer in the Core Rendering group at Silicon Graphics. He's implemented OpenGL extensions and done OpenGL performance work. He currently works on IRIS Performer, a real-time visualization library that uses OpenGL.

Prior to SGI, he worked at Sun Microsystems, where he developed graphics hardware support software and graphics libraries, including XGL.

Tom is also an adjunct professor at Santa Clara University, where he teaches courses in computer graphics using the OpenGL library. He has also presented at the X Technical Conference, SIGGRAPH '96, and SGI's 1996 Developer Forum.  
Email: [tomcat@asd.sgi.com](mailto:tomcat@asd.sgi.com)

### **Paula Womack**

Paula Womack manages the OpenGL group at Silicon Graphics. She is also a member of the OpenGL Architectural Review Board (the OpenGL ARB) which is responsible for defining and enhancing OpenGL.

Prior to joining Silicon Graphics, Paula worked on OpenGL at Kubota and Digital Equipment. She has a B.S. in Computer Engineering from the University of California at San Diego.  
Email: [womack@asd.sgi.com](mailto:womack@asd.sgi.com)

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Acknowledgments . . . . .	1
1.2	Course Notes Web Site . . . . .	2
<b>2</b>	<b>About OpenGL</b>	<b>2</b>
<b>3</b>	<b>Modelling</b>	<b>3</b>
3.1	Modelling Considerations . . . . .	3
3.2	Decomposition and Tessellation . . . . .	5
3.3	Capping Clipped Solids with the Stencil Buffer . . . . .	7
3.4	Constructive Solid Geometry with the Stencil Buffer . . . . .	8
<b>4</b>	<b>Geometry and Transformations</b>	<b>17</b>
4.1	Stereo Viewing . . . . .	17
4.1.1	Fusion Distance . . . . .	18
4.1.2	Computing the Transforms . . . . .	19
4.1.3	Rotate vs. Shear . . . . .	20
4.2	Depth of Field . . . . .	21
4.3	The Z Coordinate and Perspective Projection . . . . .	21
4.3.1	Depth Buffering . . . . .	22
4.4	Image Tiling . . . . .	26
4.5	Moving the Current Raster Position . . . . .	28
<b>5</b>	<b>Texture Mapping</b>	<b>28</b>
5.1	Review . . . . .	29
5.1.1	Filtering . . . . .	29
5.1.2	Texture Environment . . . . .	30
5.2	MIPmap Generation . . . . .	32
5.3	View Dependent Filtering . . . . .	34
5.4	Fine Tuning . . . . .	36
5.5	Paging Textures . . . . .	36
5.6	Transparency Mapping and Trimming with Alpha . . . . .	38
5.7	Billboards . . . . .	39
5.8	Rendering Text . . . . .	41
5.9	Projective Textures . . . . .	42
5.10	Environment Mapping . . . . .	42
5.11	Image Warping and Dewarping . . . . .	43
5.12	3D Textures . . . . .	43

5.12.1	Using 3D Textures . . . . .	43
5.12.2	3D Texture Portability . . . . .	44
5.12.3	3D Textures to Render Solid Materials . . . . .	45
5.12.4	3D Textures as Multidimensional Functions . . . . .	46
5.13	Procedural Texture Generation . . . . .	46
5.13.1	Filtered Noise Functions . . . . .	47
5.13.2	Generating Noise Functions . . . . .	47
5.13.3	High Resolution Filtering . . . . .	48
5.13.4	Spectral Synthesis . . . . .	49
5.13.5	Other Noise Functions . . . . .	50
5.13.6	Turbulence . . . . .	50
5.13.7	Example: Image Warping . . . . .	51
5.13.8	Generating 3D Noise . . . . .	52
5.13.9	Generating 2D Noise to Simulate 3D Noise . . . . .	52
5.13.10	Trade-offs Between 3D and 2D Techniques . . . . .	53
<b>6</b>	<b>Blending</b> . . . . .	<b>53</b>
6.1	Compositing . . . . .	53
6.2	Advanced Blending . . . . .	54
6.3	Painting . . . . .	54
6.4	Blending with the Accumulation Buffer . . . . .	55
<b>7</b>	<b>Antialiasing</b> . . . . .	<b>57</b>
7.1	Antialiasing Points and Lines . . . . .	57
7.2	Polygon Antialiasing . . . . .	58
7.3	Multisampling . . . . .	59
7.4	Antialiasing With Textures . . . . .	59
7.5	Antialiasing with Accumulation Buffer . . . . .	60
<b>8</b>	<b>Lighting</b> . . . . .	<b>63</b>
8.1	Phong Shading . . . . .	63
8.1.1	Phong Highlights with Texture . . . . .	63
8.1.2	Spotlight Effects using Projective Textures . . . . .	64
8.1.3	Phong shading by Adaptive Tessellation . . . . .	67
8.2	Light Maps . . . . .	67
8.2.1	2D Texture Light Maps . . . . .	68
8.2.2	3D Texture Light Maps . . . . .	70
8.3	Bump Mapping with Textures . . . . .	71
8.3.1	Tangent Space . . . . .	72
8.3.2	Going for higher quality . . . . .	76

8.4	Blending . . . . .	76
8.4.1	Why does this work? . . . . .	77
8.4.2	Limitations . . . . .	77
8.5	Choosing Material Properties . . . . .	78
8.5.1	Modeling Material Type . . . . .	78
8.5.2	Modeling Material Smoothness . . . . .	80
<b>9</b>	<b>Scene Realism</b>	<b>83</b>
9.1	Motion Blur . . . . .	83
9.2	Depth of Field . . . . .	83
9.3	Reflections and Refractions . . . . .	85
9.3.1	Planar Reflectors . . . . .	88
9.3.2	Sphere Mapping . . . . .	93
9.4	Creating Shadows . . . . .	102
9.4.1	Projection Shadows . . . . .	103
9.4.2	Shadow Volumes . . . . .	105
9.4.3	Shadow Maps . . . . .	108
9.4.4	Soft Shadows by Jittering Lights . . . . .	110
9.4.5	Soft Shadows Using Textures . . . . .	110
<b>10</b>	<b>Transparency</b>	<b>111</b>
10.1	Screen-Door Transparency . . . . .	111
10.2	Alpha Blending . . . . .	112
10.3	Sorting . . . . .	113
10.4	Using the Alpha Function . . . . .	114
10.5	Using Multisampling . . . . .	114
<b>11</b>	<b>Natural Phenomena</b>	<b>115</b>
11.1	Smoke . . . . .	115
11.2	Vapor Trails . . . . .	115
11.3	Fire . . . . .	116
11.4	Clouds . . . . .	117
11.5	Water . . . . .	118
11.6	Light Points . . . . .	118
11.7	Other Atmospheric Effects . . . . .	119
<b>12</b>	<b>Image Processing</b>	<b>121</b>
12.1	Introduction . . . . .	121
12.1.1	The Pixel Transfer Pipeline . . . . .	121
12.1.2	Geometric Drawing and Texturing . . . . .	122

12.1.3	The Frame Buffer and Per-Fragment Operations . . . . .	122
12.2	Colors and Color Spaces . . . . .	123
12.2.1	The Accumulation Buffer: Interpolation and Extrapolation . . . . .	123
12.2.2	Pixel Scale and Bias Operations . . . . .	125
12.2.3	Look-Up Tables . . . . .	125
12.2.4	The Color Matrix Extension . . . . .	128
12.3	Convolutions . . . . .	132
12.3.1	Introduction . . . . .	132
12.3.2	The Convolution Operation . . . . .	132
12.3.3	Convolutions Using the Accumulation Buffer . . . . .	134
12.3.4	The Convolution Extension . . . . .	137
12.3.5	Useful Convolution Filters . . . . .	137
12.4	Image Warping . . . . .	141
12.4.1	The Pixel Zoom Operation . . . . .	141
12.4.2	Warps Using Texture Mapping . . . . .	141
<b>13</b>	<b>Volume Visualization with Texture</b>	<b>144</b>
13.1	Overview of the Technique . . . . .	145
13.2	3D Texture Volume Rendering . . . . .	146
13.3	2D Texture Volume Rendering . . . . .	147
13.4	Blending Operators . . . . .	148
13.4.1	Over . . . . .	148
13.4.2	Attenuate . . . . .	148
13.4.3	MIP . . . . .	149
13.4.4	Under . . . . .	149
13.5	Sampling Frequency . . . . .	149
13.6	Shrinking the Volume Image . . . . .	151
13.7	Virtualizing Texture Memory . . . . .	151
13.8	Mixing Volumetric and Geometric Objects . . . . .	151
13.9	Transfer Functions . . . . .	152
13.10	Volume Cutting Planes . . . . .	152
13.11	Shading the Volume . . . . .	152
13.12	Warped Volumes . . . . .	153
<b>14</b>	<b>Using the Stencil Buffer</b>	<b>153</b>
14.1	Dissolves with Stencil . . . . .	156
14.2	Decaling with Stencil . . . . .	158
14.3	Finding Depth Complexity with the Stencil Buffer . . . . .	160
14.4	Compositing Images with Depth . . . . .	161



<b>15</b>	<b>Line Rendering Techniques</b>	<b>162</b>
15.1	Hidden Lines . . . . .	162
15.2	Haloed Lines . . . . .	164
15.3	Silhouette Edges . . . . .	166
<b>16</b>	<b>Tuning Your OpenGL Application</b>	<b>167</b>
16.1	What Is Pipeline Tuning? . . . . .	167
16.1.1	Three-Stage Model of the Graphics Pipeline . . . . .	168
16.1.2	Finding Bottlenecks in Your Application . . . . .	169
16.1.3	Factors Influencing Performance . . . . .	170
16.2	Optimizing Your Application Code . . . . .	170
16.2.1	Optimize Cache and Memory Usage . . . . .	170
16.2.2	Store Data in a Format That is Efficient for Rendering . . . . .	171
16.2.3	Per-Platform Tuning . . . . .	172
16.3	Tuning the Geometry Subsystem . . . . .	173
16.3.1	Use Expensive Modes Efficiently . . . . .	173
16.3.2	Optimizing Transformations . . . . .	173
16.3.3	Optimizing Lighting Performance . . . . .	174
16.3.4	Advanced Geometry-Limited Tuning Techniques . . . . .	176
16.4	Tuning the Raster Subsystem . . . . .	177
16.4.1	Using Backface/Frontface Removal . . . . .	177
16.4.2	Minimizing Per-Pixel Calculations . . . . .	177
16.4.3	Optimizing Texture Mapping . . . . .	178
16.4.4	Clearing the Color and Depth Buffers Simultaneously . . . . .	179
16.5	Rendering Geometry Efficiently . . . . .	179
16.5.1	Using Peak-Performance Primitives . . . . .	179
16.5.2	Using Vertex Arrays . . . . .	180
16.5.3	Using Display Lists . . . . .	181
16.5.4	Balancing Polygon Size and Pixel Operations . . . . .	182
16.6	Rendering Images Efficiently . . . . .	182
16.7	Tuning Animation . . . . .	183
16.7.1	Factors Contributing to Animation Speed . . . . .	183
16.7.2	Optimizing Frame Rate Performance . . . . .	184
16.8	Taking Timing Measurements . . . . .	184
16.8.1	Benchmarking Basics . . . . .	184
16.8.2	Achieving Accurate Timing Measurements . . . . .	185
16.8.3	Achieving Accurate Benchmarking Results . . . . .	186
<b>17</b>	<b>List of Demo Programs</b>	<b>187</b>

<b>18 Equation Appendix</b>	<b>190</b>
18.1 Projection Matrices . . . . .	190
18.1.1 Perspective Projection . . . . .	190
18.1.2 Orthographic Projection . . . . .	191
18.2 Lighting Equations . . . . .	191
18.2.1 Attenuation Factor . . . . .	191
18.2.2 Spotlight Effect . . . . .	191
18.2.3 Ambient Term . . . . .	192
18.2.4 Diffuse Term . . . . .	192
18.2.5 Specular Term . . . . .	192
18.2.6 Putting It All Together . . . . .	193
<b>19 References</b>	<b>193</b>

## List of Figures

1	T-intersection . . . . .	4
2	Quadrilateral decomposition . . . . .	6
3	Octahedron with triangle subdivision . . . . .	7
4	An Example Of Constructive Solid Geometry . . . . .	8
5	A CSG tree in normal form . . . . .	9
6	Thinking of a CSG tree as a sum of products . . . . .	12
7	Examples of $n$ -convex solids . . . . .	13
8	Stereo Viewing Geometry . . . . .	19
9	The relationship of window $z$ (depth) to eye $z$ for different far/near ratios . . . . .	22
10	Polygon and Outline Slopes . . . . .	25
11	Texture Tiling . . . . .	31
12	Footprint in full height texture . . . . .	34
13	Footprint in half height texture . . . . .	34
14	2D Image Roam . . . . .	37
15	Billboard with cylindrical symmetry . . . . .	39
16	3D Textures as 2D Textures varying with R . . . . .	46
17	Input Image . . . . .	49
18	Output Image . . . . .	49
19	Rasterization of a wide point. . . . .	57
20	Tangent Space Defined at Polygon Vertices . . . . .	73
21	Shifting Bump Mapping to Create Normal Components . . . . .	74
22	Jittered Eye Points . . . . .	84
23	Reflection and refraction. The image on the top shows transmission from a medium with a lower to a higher index of refraction; the image on the bottom shows transmission from higher to lower. . . . .	85
24	Total Internal Reflection . . . . .	85
25	Mirror reflection of the viewpoint . . . . .	88
26	Mirror reflection of the scene . . . . .	88
27	Creating a sphere map . . . . .	93
28	Sphere map coordinate generation . . . . .	94
29	Reflection map created using a reflective sphere . . . . .	95
30	Image cube faces captured at a cafe in Palo Alto, CA . . . . .	98
31	Sphere map generated from image cube faces in Figure 30 . . . . .	98
32	Shadow Volume . . . . .	105
33	Vapor Trail . . . . .	116
34	Slicing a 3D Texture to Render Volume . . . . .	145
35	Slicing a 3D Texture with Spheres . . . . .	146

36	Using stencil to dissolve between images . . . . .	156
37	Using stencil to render coplanar polygons . . . . .	158
38	Haloed Line . . . . .	165