

OpenGL[®]

Shading Language

Third Edition



Randi J. Rost • Bill Licea-Kane

With Contributions by Dan Ginsburg, John M. Kessenich, Barthold Lichtenbelt,
Hugh Malan, and Mike Weiblen

Praise for *OpenGL® Shading Language*

“As the ‘Red Book’ is known to be the gold standard for OpenGL, the ‘Orange Book’ is considered to be the gold standard for the OpenGL Shading Language. With Randi’s extensive knowledge of OpenGL and GLSL, you can be assured you will be learning from a graphics industry veteran. Within the pages of the second edition you can find topics from beginning shader development to advanced topics such as the spherical harmonic lighting model and more.”

—David Tommeraasen
CEO/Programmer
Plasma Software

“This will be the definitive guide for OpenGL shaders; no other book goes into this detail. Rost has done an excellent job at setting the stage for shader development, what the purpose is, how to do it, and how it all fits together. The book includes great examples and details, as well as good additional coverage of 2.0 changes!”

—Jeffery Galinovsky
Director of Emerging Market
Platform Development
Intel Corporation

“The coverage in this new edition of the book is pitched just right to help many new shader-writers get started, but with enough deep information for the ‘old hands.’”

—Marc Olano
Assistant Professor
University of Maryland

“This is a really great book on GLSL—well written and organized, very accessible, and with good real-world examples and sample code. The topics flow naturally and easily, explanatory code fragments are inserted in very logical places to illustrate concepts, and, all in all, this book makes an excellent tutorial as well as a reference.”

—John Carey
Chief Technology Officer
C.O.R.E. Feature Animation

OpenGL Shading Language

Table of Contents

Contents

Foreword to the Second Edition

Foreword to the First Edition

Preface

About the Authors

About the Contributors

Acknowledgments

Chapter 1. Review of OpenGL Basics

1.1 OpenGL History

1.2 OpenGL Evolution

1.3 Execution Model

1.4 The Framebuffer

1.5 State

1.6 Processing Pipeline

1.7 Drawing Geometry

1.7.1 Geometry Specification

1.7.2 Per-Vertex Operations

1.7.3 Primitive Assembly

1.7.4 Primitive Processing

1.7.5 Rasterization

1.7.6 Fragment Processing

1.7.7 Per-Fragment Operations

1.7.8 Framebuffer Operations

Table of Contents

1.8 Drawing Images

1.8.1 Pixel Unpacking

1.8.2 Pixel Transfer

1.8.3 Rasterization and Back-End Processing

1.8.4 Read Control

1.9 Coordinate Transforms

1.10 Texturing

1.11 Summary

1.12 Further Information

Chapter 2. Basics

2.1 Introduction to the OpenGL Shading Language

2.2 Why Write Shaders?

2.3 OpenGL Programmable Processors

2.3.1 Vertex Processor

2.3.2 Fragment Processor

2.4 Language Overview

2.4.1 Language Design Considerations

2.4.2 C Basis

2.4.3 Additions to C

2.4.4 Additions from C++

2.4.5 C Features Not Supported

2.4.6 Other Differences

2.5 System Overview

2.5.1 Driver Model

2.5.2 OpenGL Shading Language Compiler/Linker

2.5.3 OpenGL Shading Language API

2.6 Key Benefits

2.7 Summary

Table of Contents

2.8 Further Information

Chapter 3. Language Definition

3.1 Example Shader Pair

3.2 Data Types

3.2.1 Scalars

3.2.2 Vectors

3.2.3 Matrices

3.2.4 Samplers

3.2.5 Structures

3.2.6 Arrays

3.2.7 Void

3.2.8 Declarations and Scope

3.2.9 Type Matching and Promotion

3.3 Initializers and Constructors

3.4 Type Conversions

3.5 Qualifiers and Interface to a Shader

3.5.1 Uniform Qualifiers

3.5.2 Uniform Blocks

3.5.3 In Qualifiers (Vertex Shader)

3.5.4 Out Qualifiers (Vertex Shader)

3.5.5 In Qualifiers (Fragment Shader)

3.5.6 Out Qualifiers (Fragment Shader)

3.5.7 Constant Qualifiers

3.5.8 Absent Qualifier

3.6 Flow Control

3.6.1 Functions

3.6.2 Calling Conventions

3.6.3 Built-in Functions

3.7 Operations

Table of Contents

- 3.7.1 Indexing
- 3.7.2 Swizzling
- 3.7.3 Component-wise Operation

3.8 Preprocessor

3.9 Preprocessor Expressions

3.10 Error Handling

3.11 Summary

3.12 Further Information

Chapter 4. The OpenGL Programmable Pipeline

4.1 The Vertex Processor

- 4.1.1 Vertex Attributes
- 4.1.2 Special Input Variables
- 4.1.3 Uniform Variables
- 4.1.4 User-Defined Out Variables
- 4.1.5 Special Output Variables

4.2 The Fragment Processor

- 4.2.1 User-Defined In Variables
- 4.2.2 Special Input Variables
- 4.2.3 Uniform Variables
- 4.2.4 User-Defined Out Variables
- 4.2.5 Special Output Variables

4.3 Built-in Uniform Variables

4.4 Built-in Constants

4.5 Interaction with OpenGL Fixed Functionality

- 4.5.1 Point Size Mode
- 4.5.2 Clipping
- 4.5.3 Position Invariance
- 4.5.4 Texturing

Table of Contents

4.6 Summary

4.7 Further Information

Chapter 5. Built-in Functions

5.1 Angle and Trigonometry Functions

5.2 Exponential Functions

5.3 Common Functions

5.4 Geometric Functions

5.5 Matrix Functions

5.6 Vector Relational Functions

5.7 Texture Access Functions

5.8 Fragment Processing Functions

5.9 Noise Functions

5.10 Summary

5.11 Further Information

Chapter 6. Simple Shading Example

6.1 Brick Shader Overview

6.2 Vertex Shader

6.3 Fragment Shader

6.4 Observations

6.5 Summary

6.6 Further Information

Chapter 7. OpenGL Shading Language API

7.1 Obtaining Version Information

7.2 Creating Shader Objects

7.3 Compiling Shader Objects

7.4 Linking and Using Shaders

Table of Contents

- 7.5 Cleaning Up
- 7.6 Query Functions
- 7.7 Specifying Vertex Attributes
- 7.8 Specifying Uniform Variables
 - 7.8.1 Default Uniform Block
 - 7.8.2 Named Uniform Blocks
- 7.9 Samplers
- 7.10 Multiple Render Targets
- 7.11 Development Aids
- 7.12 Implementation-Dependent API Values
- 7.13 Application Code for Brick Shaders
- 7.14 Summary
- 7.15 Further Information

Chapter 8. Shader Development

- 8.1 General Principles
 - 8.1.1 Understand the Problem
 - 8.1.2 Add Complexity Progressively
 - 8.1.3 Test and Iterate
 - 8.1.4 Strive for Simplicity
 - 8.1.5 Exploit Modularity
- 8.2 Performance Considerations
 - 8.2.1 Consider Computational Frequency
 - 8.2.2 Analyze Your Algorithm
 - 8.2.3 Use the Built-in Functions
 - 8.2.4 Use Vectors
 - 8.2.5 Use Textures to Encode Complex Functions
 - 8.2.6 Review the Information Logs
- 8.3 Shader Debugging

Table of Contents

8.3.1 Use the Vertex Shader Output

8.3.2 Use the Fragment Shader Output

8.3.3 Use Simple Geometry

8.4 Shader Development Tools

8.4.1 RenderMonkey

8.4.2 Apple GLSLEditorSample

8.4.3 Graphic Remedy gDEDebugger

8.4.4 OpenGL Shading Language Compiler Front End

8.5 Scene Graphs

8.6 Summary

8.7 Further Information

Chapter 9. Emulating OpenGL Fixed Functionality

9.1 Transformation

9.2 Light Sources

9.2.1 Directional Lights

9.2.2 Point Lights

9.2.3 Spotlights

9.3 Material Properties and Lighting

9.4 Two-Sided Lighting

9.5 No Lighting

9.6 Fog

9.7 Texture Coordinate Generation

9.8 User Clipping

9.9 Texture Application

9.10 Matrices

9.10.1 Identity Matrix

9.10.2 Scale

9.10.3 Translate

Table of Contents

9.10.4 Rotate

9.10.5 Ortho

9.10.6 Frustum

9.11 Operating on the Current Matrices

9.11.1 A Simple Matrix Transformation Example

9.12 Summary

9.13 Further Information

Chapter 10. Stored Texture Shaders

10.1 Access to Texture Maps from a Shader

10.2 Simple Texturing Example

10.2.1 Application Setup

10.2.2 Vertex Shader

10.2.3 Fragment Shader

10.3 Multitexturing Example

10.3.1 Application Setup

10.3.2 Vertex Shader

10.3.3 Fragment Shader

10.4 Cube Mapping Example

10.4.1 Application Setup

10.4.2 Vertex Shader

10.4.3 Fragment Shader

10.5 Another Environment Mapping Example

10.5.1 Vertex Shader

10.5.2 Fragment Shader

10.6 Glyph Bombing

10.6.1 Application Setup

10.6.2 Vertex Shader

10.6.3 Fragment Shader

Table of Contents

10.7 Summary

10.8 Further Information

Chapter 11. Procedural Texture Shaders

11.1 Regular Patterns

11.1.1 Stripes Vertex Shader

11.1.2 Stripes Fragment Shader

11.2 Toy Ball

11.2.1 Application Setup

11.2.2 Vertex Shader

11.2.3 Fragment Shader

11.3 Lattice

11.4 Bump Mapping

11.4.1 Application Setup

11.4.2 Vertex Shader

11.4.3 Fragment Shader

11.4.4 Normal Maps

11.5 Summary

11.6 Further Information

Chapter 12. Lighting

12.1 Hemisphere Lighting

12.2 Image-Based Lighting

12.3 Lighting with Spherical Harmonics

12.4 The Überlight Shader

12.4.1 Überlight Controls

12.4.2 Vertex Shader

12.4.3 Fragment Shader

12.5 Summary

Table of Contents

12.6 Further Information

Chapter 13. Shadows

13.1 Ambient Occlusion

13.2 Shadow Maps

13.2.1 Application Setup

13.2.2 Vertex Shader

13.2.3 Fragment Shader

13.3 Deferred Shading for Volume Shadows

13.3.1 Shaders for First Pass

13.3.2 Shaders for Second Pass

13.4 Summary

13.5 Further Information

Chapter 14. Surface Characteristics

14.1 Refraction

14.2 Diffraction

14.3 BRDF Models

14.4 Polynomial Texture Mapping with BRDF Data

14.4.1 Application Setup

14.4.2 Vertex Shader

14.4.3 Fragment Shader

14.5 Summary

14.6 Further Information

Chapter 15. Noise

15.1 Noise Defined

15.1.1 2D Noise

15.1.2 Higher Dimensions of Noise

15.1.3 Using Noise in OpenGL Shaders

Table of Contents

15.2 Noise Textures

15.3 Trade-offs

15.4 A Simple Noise Shader

15.4.1 Application Setup

15.4.2 Vertex Shader

15.4.3 Fragment Shader

15.5 Turbulence

15.5.1 Sun Surface Shader

15.5.2 Marble

15.6 Granite

15.7 Wood

15.7.1 Application Setup

15.7.2 Fragment Shader

15.8 Summary

15.9 Further Information

Chapter 16. Animation

16.1 On/Off

16.2 Threshold

16.3 Translation

16.4 Morphing

16.4.1 Sphere Morph Vertex Shader

16.5 Other Blending Effects

16.6 Vertex Noise

16.7 Particle Systems

16.7.1 Application Setup

16.7.2 Confetti Cannon Vertex Shader

16.7.3 Further Enhancements

16.8 Wobble

Table of Contents

16.9 Animating Once per Frame

16.9.1 Application Setup

16.9.2 Updating Matrices Once per Frame

16.10 Summary

16.11 Further Information

Chapter 17. Antialiasing Procedural Textures

17.1 Sources of Aliasing

17.2 Avoiding Aliasing

17.3 Increasing Resolution

17.4 Antialiased Stripe Example

17.4.1 Generating Stripes

17.4.2 Analytic Prefiltering

17.4.3 Adaptive Analytic Prefiltering

17.4.4 Analytic Integration

17.4.5 Antialiased Brick Fragment Shader

17.5 Frequency Clamping

17.5.1 Antialiased Checkerboard Fragment Shader

17.6 Summary

17.7 Further Information

Chapter 18. Non-photorealistic Shaders

18.1 Hatching Example

18.1.1 Application Setup

18.1.2 Vertex Shader

18.1.3 Generating Hatching Strokes

18.1.4 Obtaining Uniform Line Density

18.1.5 Simulating Lighting

18.1.6 Adding Character

18.1.7 Hatching Fragment Shader

Table of Contents

18.2 Technical Illustration Example

18.2.1 Application Setup

18.2.2 Vertex Shader

18.2.3 Fragment Shader

18.3 Mandelbrot Example

18.3.1 About the Mandelbrot Set

18.3.2 Vertex Shader

18.3.3 Fragment Shader

18.3.4 Julia Sets

18.4 Summary

18.5 Further Information

Chapter 19. Shaders for Imaging

19.1 Geometric Image Transforms

19.2 Mathematical Mappings

19.3 Lookup Table Operations

19.4 Color Space Conversions

19.5 Image Interpolation and Extrapolation

19.5.1 Brightness

19.5.2 Contrast

19.5.3 Saturation

19.5.4 Sharpness

19.6 Blend Modes

19.6.1 Normal

19.6.2 Average

19.6.3 Dissolve

19.6.4 Behind

19.6.5 Clear

19.6.6 Darken

Table of Contents

- 19.6.7 Lighten
- 19.6.8 Multiply
- 19.6.9 Screen
- 19.6.10 Color Burn
- 19.6.11 Color Dodge
- 19.6.12 Overlay
- 19.6.13 Soft Light
- 19.6.14 Hard Light
- 19.6.15 Add
- 19.6.16 Subtract
- 19.6.17 Difference
- 19.6.18 Inverse Difference
- 19.6.19 Exclusion
- 19.6.20 Opacity

19.7 Convolution

- 19.7.1 Smoothing
- 19.7.2 Edge Detection
- 19.7.3 Sharpening

19.8 Summary

19.9 Further Information

Chapter 20. Language Comparison

- 20.1 Chronology of Shading Languages
- 20.2 RenderMan
- 20.3 OpenGL Shader (ISL)
- 20.4 HLSL
- 20.5 Cg
- 20.6 Summary
- 20.7 Further Information

Table of Contents

Appendix A: Language Grammar

Appendix B: API Function Reference

Implementation-Dependent API Values for GLSL

Other Queriable Values for GLSL

glAttachShader

glBindAttribLocation

glCompileShader

glCreateProgram

glCreateShader

glDeleteProgram

glDeleteShader

glDetachShader

glDrawBuffers

glEnableVertexAttribArray

glGetActiveAttrib

glGetActiveUniform

glGetAttachedShaders

glGetAttribLocation

glGetProgram

glGetProgramInfoLog

glGetShader

glGetShaderInfoLog

glGetShaderSource

glGetUniform

glGetUniformLocation

glGetVertexAttrib

Table of Contents

glGetVertexAttribPointer

glIsProgram

glIsShader

glLinkProgram

glShaderSource

glUniform

glUseProgram

glValidateProgram

glVertexAttrib

glVertexAttribPointer

OpenGL 1.5 to OpenGL 2.0 GLSL Migration Guide

Afterword

Glossary

A

B

C

D

E

F

G

H

I

K

L

M

N

Table of Contents

O

P

R

S

T

U

V

W

Further Reading

Index