

OpenGL[®]

Programming Guide

Ninth Edition



*The Official Guide to Learning
OpenGL[®], Version 4.5 with SPIR-V*



John Kessenich • Graham Sellers • Dave Shreiner

The Khronos OpenGL ARB Working Group

Praise for previous editions of OpenGL[®] Programming Guide

“Wow! This book is basically one-stop shopping for OpenGL information. It is the kind of book that I will be reaching for a lot. Thanks to Dave, Graham, John, and Bill for an amazing effort.”

—Mike Bailey, professor, Oregon State University

“The most recent Red Book parallels the grand tradition of OpenGL; continuous evolution towards ever-greater power and efficiency. The eighth edition contains up-to-the minute information about the latest standard and new features, along with a solid grounding in modern OpenGL techniques that will work anywhere. The Red Book continues to be an essential reference for all new employees at my simulation company. What else can be said about this essential guide? I laughed, I cried, it was much better than Cats—I’ll read it again and again.”

—Bob Kuehne, president, Blue Newt Software

“OpenGL has undergone enormous changes since its inception twenty years ago. This new edition is your practical guide to using the OpenGL of today. Modern OpenGL is centered on the use of shaders, and this edition of the Programming Guide jumps right in, with shaders covered in depth in Chapter 2. It continues in later chapters with even more specifics on everything from texturing to compute shaders. No matter how well you know it or how long you’ve been doing it, if you are going to write an OpenGL program, you want to have a copy of the *OpenGL[®] Programming Guide* handy.”

—Marc Olano, associate professor, UMBC

“If you are looking for the definitive guide to programming with the very latest version of OpenGL, look no further. The authors of this book have been deeply involved in the creation of OpenGL 4.3, and everything you need to know about the cutting edge of this industry-leading API is laid out here in a clear, logical, and insightful manner.”

—Neil Trevett, president, Khronos Group

OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 4.5 with SPIR-V

Table of Contents

Cover

Title Page

Copyright Page

Contents

Figures

Tables

Examples

About This Guide

- What This Guide Contains

- Whats New in This Edition

- What You Should Know Before Reading This Guide

- How to Obtain the Sample Code

- Errata

- Style Conventions

- About the OpenGL Series

Acknowledgments

1. Introduction to OpenGL

- What Is OpenGL?

- Your First Look at an OpenGL Program

- OpenGL Syntax

Table of Contents

OpenGLs Rendering Pipeline

- Preparing to Send Data to OpenGL

- Sending Data to OpenGL

- Vertex Shading

- Tessellation Shading

- Geometry Shading

- Primitive Assembly

- Clipping

- Rasterization

- Fragment Shading

- Per-Fragment Operations

Our First Program: A Detailed Discussion

- Entering main()

- OpenGL Initialization

- Our First OpenGL Drawing

2. Shader Fundamentals

- Shaders and OpenGL

- OpenGLs Programmable Pipeline

- An Overview of the OpenGL Shading Language

- Creating Shaders with GLSL

- Storage Qualifiers

- Statements

- Computational Invariance

- Shader Preprocessor

- Compiler Control

- Global Shader-Compilation Option

- Interface Blocks

- Uniform Blocks

- Specifying Uniform Blocks in Shaders

Table of Contents

Accessing Uniform Blocks from Your Application

Buffer Blocks

In/Out Blocks, Locations, and Components

Compiling Shaders

Shader Subroutines

GLSL Subroutine Setup

Selecting Shader Subroutines

Separate Shader Objects

SPIR-V

Reasons to Choose SPIR-V

Using SPIR-V with OpenGL

Using GLSL to Generate SPIR-V for OpenGL

Glslang

Whats Inside SPIR-V?

3. Drawing with OpenGL

OpenGL Graphics Primitives

Points

Lines, Strips, and Loops

Triangles, Strips, and Fans

Data in OpenGL Buffers

Creating and Allocating Buffers

Getting Data into and out of Buffers

Accessing the Content of Buffers

Discarding Buffer Data

Vertex Specification

VertexAttribPointer in Depth

Static Vertex-Attribute Specification

OpenGL Drawing Commands

Table of Contents

Restarting Primitives

Instanced Rendering

4. Color, Pixels, and Fragments

Basic Color Theory

Buffers and Their Uses

Clearing Buffers

Masking Buffers

Color and OpenGL

Color Representation and OpenGL

Smoothly Interpolating Data

Testing and Operating on Fragments

Scissor Test

Multisample Fragment Operations

Stencil Test

Stencil Examples

Depth Test

Blending

Logical Operations

Occlusion Query

Conditional Rendering

Multisampling

Sample Shading

Per-Primitive Antialiasing

Antialiasing Lines

Antialiasing Polygons

Reading and Copying Pixel Data

Copying Pixel Rectangles

5. Viewing Transformations, Culling, Clipping, and Feedback

Table of Contents

Viewing

- Viewing Model

- Camera Model

- Orthographic Viewing Model

User Transformations

- Matrix Multiply Refresher

- Homogeneous Coordinates

- Linear Transformations and Matrices

- Transforming Normals

- OpenGL Matrices

OpenGL Transformations

- Advanced: User Culling and Clipping

- Controlling OpenGL Transformations

Transform Feedback

- Transform Feedback Objects

- Transform Feedback Buffers

- Configuring Transform Feedback Varyings

- Starting and Stopping Transform Feedback

- Transform Feedback ExampleParticle System

6. Textures and Framebuffers

- Introduction to Texturing

- Basic Texture Types

- Creating and Initializing Textures

- Proxy Textures

- Specifying Texture Data

- Explicitly Setting Texture Data

- Loading Textures from Buffers

- Loading Images from Files

- Retrieving Texture Data

Table of Contents

Texture Data Layout

Texture Formats

Internal Formats

External Formats

Compressed Textures

Sampler Objects

Sampler Parameters

Using Textures

Texture Coordinates

Arranging Texture Data

Using Multiple Textures

Complex Texture Types

3D Textures

Array Textures

Cube-Map Textures

Shadow Samplers

Depth-Stencil Textures

Buffer Textures

Texture Views

Filtering

Linear Filtering

Using and Generating Mipmaps

Calculating the Mipmap Level

Mipmap Level-of-Detail Control

Advanced Texture Lookup Functions

Explicit Level of Detail

Explicit Gradient Specification

Texture Fetch with Offsets

Projective Texturing

Table of Contents

- Texture Queries in Shaders
- Gathering Texels
- Combining Special Functions

Bindless Textures

- Texture Handles
- Texture Residency
- Sampling Bindless Textures

Sparse Textures

- Sparse Texture Commitment
- Sparse Texture Pages

Point Sprites

- Textured Point Sprites
- Controlling the Appearance of Points

Framebuffer Objects

Rendering to Texture Maps

- Discarding Rendered Data
- Renderbuffers
- Creating Renderbuffer Storage
- Framebuffer Attachments
- Framebuffer Completeness
- Invalidating Framebuffers
- Writing to Multiple Renderbuffers Simultaneously
- Selecting Color Buffers for Writing and Reading
- Dual-Source Blending

Chapter Summary

- Texture Redux
- Texture Best Practices

7. Light and Shadow

Table of Contents

Lighting Introduction

Classic Lighting Model

- Fragment Shaders for Different Light Styles

- Moving Calculations to the Vertex Shader

- Multiple Lights and Materials

- Lighting Coordinate Systems

- Limitations of the Classic Lighting Model

Advanced Lighting Models

- Hemisphere Lighting

- Image-Based Lighting

- Lighting with Spherical Harmonics

Shadow Mapping

- Creating a Shadow Map

- Using a Shadow Map

8. Procedural Texturing

Procedural Texturing

- Regular Patterns

- Toy Ball

- Lattice

- Procedural Shading Summary

Bump Mapping

- Application Setup

- Vertex Shader

- Fragment Shader

- Normal Maps

Antialiasing Procedural Textures

- Sources of Aliasing

- Avoiding Aliasing

- Increasing Resolution

Table of Contents

Antialiasing High Frequencies

Frequency Clamping

Procedural Antialiasing Summary

Noise

Definition of Noise

Noise Textures

Trade-Offs

A Simple Noise Shader

Turbulence

Marble

Granite

Wood

Noise Summary

Further Information

9. Tessellation Shaders

Tessellation Shaders

Tessellation Patches

Tessellation Control Shaders

Generating Output-Patch Vertices

Tessellation Control Shader Variables

Controlling Tessellation

Tessellation Evaluation Shaders

Specifying the Primitive Generation Domain

Specifying the Face Winding for Generated Primitives

Specifying the Spacing of Tessellation Coordinates

Additional Tessellation Evaluation Shaderlayout Options

Specifying a Vertexs Position

Tessellation Evaluation Shader Variables

A Tessellation Example: The Teapot

Table of Contents

Processing Patch Input Vertices

Evaluating Tessellation Coordinates for the Teapot

Additional Tessellation Techniques

View-Dependent Tessellation

Shared Tessellated Edges and Cracking

Displacement Mapping

10. Geometry Shaders

Creating a Geometry Shader

Geometry Shader Inputs and Outputs

Geometry Shader Inputs

Special Geometry Shader Primitives

Geometry Shader Outputs

Producing Primitives

Culling Geometry

Geometry Amplification

Advanced Transform Feedback

Multiple Output Streams

Primitive Queries

Using Transform Feedback Results

Geometry Shader Instancing

Multiple Viewports and Layered Rendering

Viewport Index

Layered Rendering

Chapter Summary

Geometry Shader Redux

Geometry Shader Best Practices

11. Memory

Using Textures for Generic Data Storage

Table of Contents

Binding Textures to Image Units

Reading and Writing to Images

Shader Storage Buffer Objects

Writing Structured Data

Atomic Operations and Synchronization

Atomic Operations on Images

Atomic Operations on Buffers

Sync Objects

Image Qualifiers and Barriers

High-Performance Atomic Counters

Example: Order-Independent Transparency

Principles of Operation

Initialization

Rendering

Sorting and Blending

Results

12. Compute Shaders

Overview

Workgroups and Dispatch

Knowing Where You Are

Communication and Synchronization

Communication

Synchronization

Examples

Physical Simulation

Image Processing

Chapter Summary

Compute Shader Redux

Compute Shader Best Practices

Table of Contents

A. Support Libraries

Basics of GLFW: The OpenGL Utility Framework

Initializing and Creating a Window

Handling User Input

Controlling the Window

Shutting Down Cleanly

GL3W: OpenGL Glue

B. OpenGL ES and WebGL

OpenGL ES

WebGL

Setting Up WebGL Within an HTML5 Page

Initializing Shaders in WebGL

Initializing Vertex Data in WebGL

Using Texture Maps in WebGL

C. Built-in GLSL Variables and Functions

Built-in Variables

Built-in Variable Declarations

Built-in Variable Descriptions

Built-in Constants

Built-in Functions

Angle and Trigonometry Functions

Exponential Functions

Common Functions

Floating-Point Pack and Unpack Functions

Geometric Functions

Matrix Functions

Vector Relational Functions

Integer Functions

Table of Contents

- Texture Functions
- Atomic-Counter Functions
- Atomic Memory Functions
- Image Functions
- Fragment Processing Functions
- Geometry Shader Functions
- Shader Invocation Control Functions
- Shader Memory Control Functions

D. State Variables

- The Query Commands

- OpenGL State Variables

- Current Values and Associated Data
 - Vertex Array Object State
 - Vertex Array Data
 - Buffer Object State
 - Transformation State
 - Coloring State
 - Rasterization State
 - Multisampling
 - Textures
 - Pixel Operations
 - Framebuffer Controls
 - Framebuffer State
 - Renderbuffer State
 - Pixel State
 - Shader Object State
 - Shader Program Pipeline Object State
 - Shader Program Object State
 - Program Interface State

Table of Contents

Program Object Resource State
Vertex and Geometry Shader State
Query Object State
Image State
Transform Feedback State
Atomic Counter State
Shader Storage Buffer State
Sync Object State
Hints
Compute Dispatch State
Implementation-Dependent Values
Tessellation Shader Implementation-Dependent Limits
Geometry Shader Implementation-Dependent Limits
Fragment Shader Implementation-Dependent Limits
Implementation-Dependent Compute Shader Limits
Implementation-Dependent Shader Limits
Implementation-Dependent Debug Output State
Implementation-Dependent Values
Internal Format-Dependent Values
Implementation-Dependent Transform Feedback Limits
Framebuffer-Dependent Values
Miscellaneous

E. Homogeneous Coordinates and Transformation Matrices

Homogeneous Coordinates

Transforming Vertices

Transforming Normals

Transformation Matrices

Translation

Scaling

Table of Contents

Rotation

Perspective Projection

Orthographic Projection

F. Floating-Point Formats for Textures, Framebuffers, and Renderbuffers

Reduced-Precision Floating-Point Values

16-Bit Floating-Point Values

10- and 11-Bit Unsigned Floating-Point Values

G. Debugging and Profiling OpenGL

Creating a Debug Context

Debug Output

Debug Messages

Filtering Messages

Application-Generated Messages

Debug Groups

Naming Objects

Profiling

Profiling Tools

In-Application Profiling

H. Buffer Object Layouts

Using Standard Layout Qualifiers

The std140 Layout Rules

The std430 Layout Rules

Glossary

A

B

C

Table of Contents

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Index