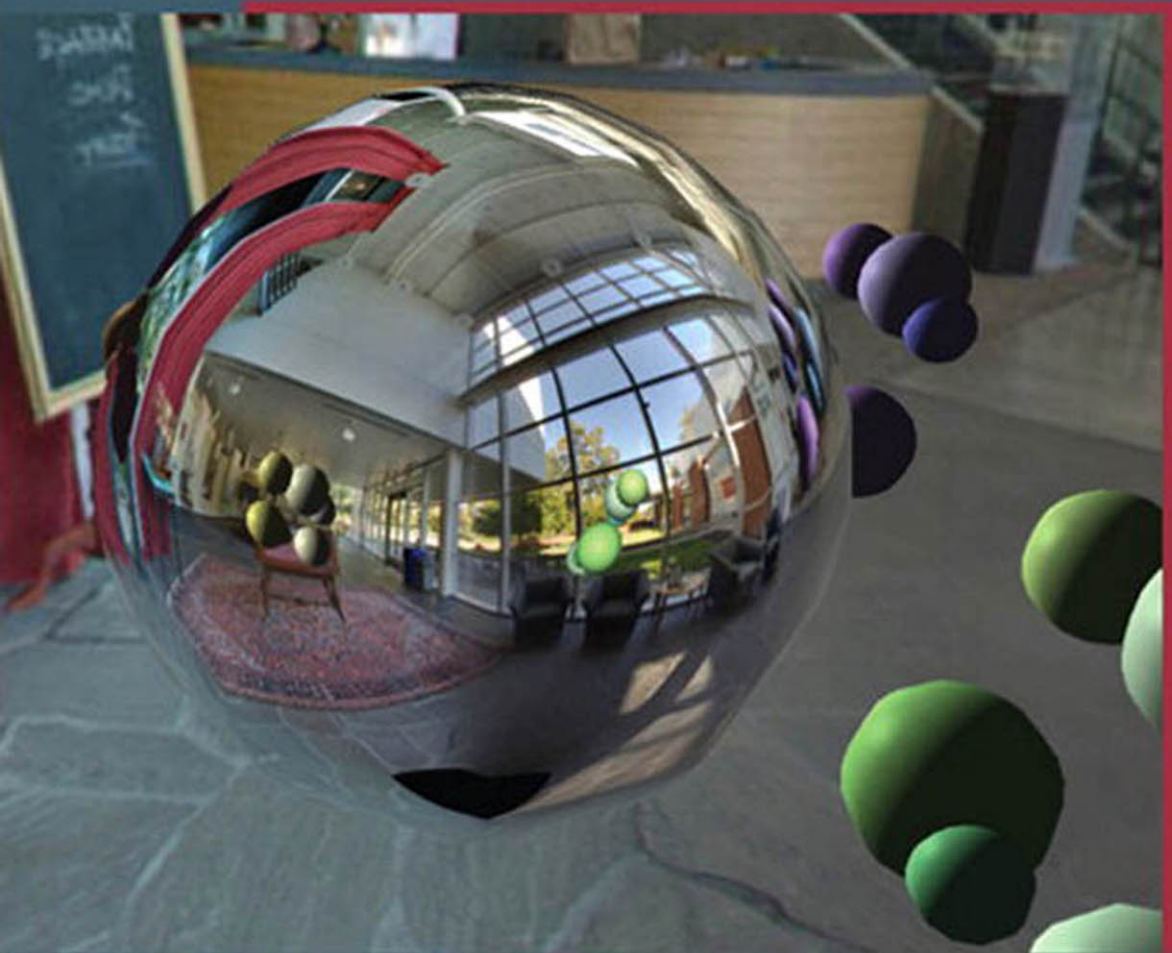




WebGL[®] Programming Guide

*Interactive 3D Graphics Programming
with WebGL*



Kouichi Matsuda ■ Rodger Lea

Praise for *WebGL Programming Guide*

“WebGL provides one of the final features for creating applications that deliver ‘the desktop application experience’ in a web browser, and the *WebGL Programming Guide* leads the way in creating those applications. Its coverage of all aspects of using WebGL—JavaScript, OpenGL ES, and fundamental graphics techniques—delivers a thorough education on everything you need to get going. Web-based applications are the wave of the future, and this book will get you ahead of the curve!”

Dave Shreiner, Coauthor of *The OpenGL Programming Guide, Eighth Edition*; Series Editor, *OpenGL Library* (Addison Wesley)

“HTML5 is evolving the Web into a highly capable application platform supporting beautiful, engaging, and fully interactive applications that run portably across many diverse systems. WebGL is a vital part of HTML5, as it enables web programmers to access the full power and functionality of state-of-the-art 3D graphics acceleration. WebGL has been designed to run securely on any web-capable system and will unleash a new wave of developer innovation in connected 3D web-content, applications, and user interfaces. This book will enable web developers to fully understand this new wave of web functionality and leverage the exciting opportunities it creates.”

Neil Trevett, Vice President Mobile Content, NVIDIA; President, The Khronos Group

“With clear explanations supported by beautiful 3D renderings, this book does wonders in transforming a complex topic into something approachable and appealing. Even without denying the sophistication of WebGL, it is an accessible resource that beginners should consider picking up before anything else.”

Evan Burchard, Author, *Web Game Developer's Cookbook* (Addison Wesley)

“Both authors have a strong OpenGL background and transfer this knowledge nicely over to WebGL, resulting in an excellent guide for beginners as well as advanced readers.”

Daniel Haehn, Research Software Developer, Boston Children's Hospital

“*WebGL Programming Guide* provides a straightforward and easy-to-follow look at the mechanics of building 3D applications for the Web without relying on bulky libraries or wrappers. A great resource for developers seeking an introduction to 3D development concepts mixed with cutting-edge web technology.”

Brandon Jones, Software Engineer, Google

WebGL Programming Guide: Interactive 3D Graphics Programming with WebGL

Table of Contents

Contents

Preface

1. Overview of WebGL

Advantages of WebGL

You Can Start Developing 3D Graphics Applications Using Only a
Text Editor

Publishing Your 3D Graphics Applications Is Easy

You Can Leverage the Full Functionality of the Browser

Learning and Using WebGL Is Easy

Origins of WebGL

Structure of WebGL Applications

Summary

2. Your First Step with WebGL

What Is a Canvas?

Using the <canvas> Tag

DrawRectangle.js

The Worlds Shortest WebGL Program: Clear Drawing Area

The HTML File (HelloCanvas.html)

JavaScript Program (HelloCanvas.js)

Experimenting with the Sample Program

Table of Contents

Draw a Point (Version 1)

HelloPoint1.html

HelloPoint1.js

What Is a Shader?

The Structure of a WebGL Program that Uses Shaders

Initializing Shaders

Vertex Shader

Fragment Shader

The Draw Operation

The WebGL Coordinate System

Experimenting with the Sample Program

Draw a Point (Version 2)

Using Attribute Variables

Sample Program (HelloPoint2.js)

Getting the Storage Location of an Attribute Variable

Assigning a Value to an Attribute Variable

Family Methods of `gl.vertexAttrib3f()`

Experimenting with the Sample Program

Draw a Point with a Mouse Click

Sample Program (ClickedPoints.js)

Register Event Handlers

Handling Mouse Click Events

Experimenting with the Sample Program

Change the Point Color

Sample Program (ColoredPoints.js)

Uniform Variables

Retrieving the Storage Location of a Uniform Variable

Assigning a Value to a Uniform Variable

Table of Contents

Family Methods of `gl.uniform4f()`

Summary

3. Drawing and Transforming Triangles

Drawing Multiple Points

Sample Program (MultiPoint.js)

Using Buffer Objects

Create a Buffer Object (`gl.createBuffer()`)

Bind a Buffer Object to a Target (`gl.bindBuffer()`)

Write Data into a Buffer Object (`gl.bufferData()`)

Typed Arrays

Assign the Buffer Object to an Attribute Variable (`gl.vertexAttribPointer()`)

Enable the Assignment to an Attribute Variable (`gl.enableVertexAttribArray()`)

The Second and Third Parameters of `gl.drawArrays()`

Experimenting with the Sample Program

Hello Triangle

Sample Program (HelloTriangle.js)

Basic Shapes

Experimenting with the Sample Program

Hello Rectangle (HelloQuad)

Experimenting with the Sample Program

Moving, Rotating, and Scaling

Translation

Sample Program (TranslatedTriangle.js)

Rotation

Sample Program (RotatedTriangle.js)

Transformation Matrix: Rotation

Transformation Matrix: Translation

Rotation Matrix, Again

Table of Contents

Sample Program (RotatedTriangle_Matrix.js)

Reusing the Same Approach for Translation

Transformation Matrix: Scaling

Summary

4. More Transformations and Basic Animation

Translate and Then Rotate

Transformation Matrix Library: cuon-matrix.js

Sample Program (RotatedTriangle_Matrix4.js)

Combining Multiple Transformation

Sample Program (RotatedTranslatedTriangle.js)

Experimenting with the Sample Program

Animation

The Basics of Animation

Sample Program (RotatingTriangle.js)

Repeatedly Call the Drawing Function (tick())

Draw a Triangle with the Specified Rotation Angle (draw())

Request to Be Called Again (requestAnimationFrame())

Update the Rotation Angle (animate())

Experimenting with the Sample Program

Summary

5. Using Colors and Texture Images

Passing Other Types of Information to Vertex Shaders

Sample Program (MultiAttributeSize.js)

Create Multiple Buffer Objects

The gl.vertexAttribPointer() Stride and Offset Parameters

Sample Program (MultiAttributeSize_Interleaved.js)

Modifying the Color (Varying Variable)

Table of Contents

Sample Program (MultiAttributeColor.js)

Experimenting with the Sample Program

Color Triangle (ColoredTriangle.js)

Geometric Shape Assembly and Rasterization

Fragment Shader Invocations

Experimenting with the Sample Program

Functionality of Varying Variables and the Interpolation Process

Pasting an Image onto a Rectangle

Texture Coordinates

Pasting Texture Images onto the Geometric Shape

Sample Program (TexturedQuad.js)

Using Texture Coordinates (initVertexBuffers())

Setting Up and Loading Images (initTextures())

Make the Texture Ready to Use in the WebGL System (loadTexture())

Flip an Images Y-Axis

Making a Texture Unit Active (gl.activeTexture())

Binding a Texture Object to a Target (gl.bindTexture())

Set the Texture Parameters of a Texture Object (gl.texParameteri())

Assigning a Texture Image to a Texture Object (gl.texImage2D())

Pass the Texture Unit to the Fragment Shader (gl.uniform1i())

Passing Texture Coordinates from the Vertex Shader to the Fragment Shader

Retrieve the Texel Color in a Fragment Shader (texture2D())

Experimenting with the Sample Program

Pasting Multiple Textures to a Shape

Sample Program (MultiTexture.js)

Summary

6. The OpenGL ES Shading Language (GLSL ES)

Table of Contents

Recap of Basic Shader Programs

Overview of GLSL ES

Hello Shader!

- Basics

- Order of Execution

- Comments

Data (Numerical and Boolean Values)

Variables

GLSL ES Is a Type Sensitive Language

Basic Types

- Assignment and Type Conversion

- Operations

Vector Types and Matrix Types

- Assignments and Constructors

- Access to Components

- Operations

Structures

- Assignments and Constructors

- Access to Members

- Operations

Arrays

Samplers

Precedence of Operators

Conditional Control Flow and Iteration

- if Statement and if-else Statement

- for Statement

- continue, break, discard Statements

Table of Contents

Functions

- Prototype Declarations

- Parameter Qualifiers

Built-In Functions

Global Variables and Local Variables

Storage Qualifiers

- const Variables

- Attribute Variables

- Uniform Variables

- Varying Variables

Precision Qualifiers

Preprocessor Directives

Summary

7. Toward the 3D World

Whats Good for Triangles Is Good for Cubes

Specifying the Viewing Direction

- Eye Point, Look-At Point, and Up Direction

- Sample Program (LookAtTriangles.js)

- Comparing LookAtTriangles.js with RotatedTriangle_Matrix4.js

- Looking at Rotated Triangles from a Specified Position

- Sample Program (LookAtRotatedTriangles.js)

- Experimenting with the Sample Program

- Changing the Eye Point Using the Keyboard

- Sample Program (LookAtTrianglesWithKeys.js)

- Missing Parts

Specifying the Visible Range (Box Type)

- Specify the Viewing Volume

Table of Contents

Defining a Box-Shaped Viewing Volume

Sample Program (OrthoView.html)

Sample Program (OrthoView.js)

Modifying an HTML Element Using JavaScript

The Processing Flow of the Vertex Shader

Changing Near or Far

Restoring the Clipped Parts of the Triangles

(LookAtTrianglesWithKeys_ViewVolume.js)

Experimenting with the Sample Program

Specifying the Visible Range Using a Quadrangular Pyramid

Setting the Quadrangular Pyramid Viewing Volume

Sample Program (PerspectiveView.js)

The Role of the Projection Matrix

Using All the Matrices (Model Matrix, View Matrix, and Projection Matrix)

Sample Program (PerspectiveView_mvp.js)

Experimenting with the Sample Program

Correctly Handling Foreground and Background Objects

Hidden Surface Removal

Sample Program (DepthBuffer.js)

Z Fighting

Hello Cube

Drawing the Object with Indices and Vertices Coordinates

Sample Program (HelloCube.js)

Writing Vertex Coordinates, Colors, and Indices to the Buffer Object

Adding Color to Each Face of a Cube

Sample Program (ColoredCube.js)

Experimenting with the Sample Program

Summary

Table of Contents

8. Lighting Objects

Lighting 3D Objects

Types of Light Source

Types of Reflected Light

Shading Due to Directional Light and Its Diffuse Reflection

Calculating Diffuse Reflection Using the Light Direction and the Orientation of a Surface

The Orientation of a Surface: What Is the Normal?

Sample Program (LightedCube.js)

Add Shading Due to Ambient Light

Sample Program (LightedCube_ambient.js)

Lighting the Translated-Rotated Object

The Magic Matrix: Inverse Transpose Matrix

Sample Program (LightedTranslatedRotatedCube.js)

Using a Point Light Object

Sample Program (PointLightedCube.js)

More Realistic Shading: Calculating the Color per Fragment

Sample Program (PointLightedCube_perFragment.js)

Summary

9. Hierarchical Objects

Drawing and Manipulating Objects Composed of Other Objects

Hierarchical Structure

Single Joint Model

Sample Program (JointModel.js)

Draw the Hierarchical Structure (draw())

A Multijoint Model

Sample Program (MultiJointModel.js)

Draw Segments (drawBox())

Table of Contents

Draw Segments (`drawSegment()`)

Shader and Program Objects: The Role of `initShaders()`

Create Shader Objects (`gl.createShader()`)

Store the Shader Source Code in the Shader Objects
(`gl.shaderSource()`)

Compile Shader Objects (`gl.compileShader()`)

Create a Program Object (`gl.createProgram()`)

Attach the Shader Objects to the Program Object
(`gl.attachShader()`)

Link the Program Object (`gl.linkProgram()`)

Tell the WebGL System Which Program Object to Use
(`gl.useProgram()`)

The Program Flow of `initShaders()`

Summary

10. Advanced Techniques

Rotate an Object with the Mouse

How to Implement Object Rotation

Sample Program (`RotateObject.js`)

Select an Object

How to Implement Object Selection

Sample Program (`PickObject.js`)

Select the Face of the Object

Sample Program (`PickFace.js`)

HUD (Head Up Display)

How to Implement a HUD

Sample Program (`HUD.html`)

Sample Program (`HUD.js`)

Display a 3D Object on a Web Page (`3DoverWeb`)

Table of Contents

Fog (Atmospheric Effect)

- How to Implement Fog

- Sample Program (Fog.js)

- Use the w Value (Fog_w.js)

Make a Rounded Point

- How to Implement a Rounded Point

- Sample Program (RoundedPoints.js)

Alpha Blending

- How to Implement Alpha Blending

- Sample Program (LookAtBlendedTriangles.js)

- Blending Function

- Alpha Blend 3D Objects (BlendedCube.js)

- How to Draw When Alpha Values Coexist

Switching Shaders

- How to Implement Switching Shaders

- Sample Program (ProgramObject.js)

Use What Youve Drawn as a Texture Image

- Framebuffer Object and Renderbuffer Object

- How to Implement Using a Drawn Object as a Texture

- Sample Program (FramebufferObjectj.js)

- Create Frame Buffer Object (gl.createFramebuffer())

- Create Texture Object and Set Its Size and Parameters

- Create Renderbuffer Object (gl.createRenderbuffer())

- Bind Renderbuffer Object to Target and Set Size (gl.bindRenderbuffer(),
gl.renderbufferStorage())

- Set Texture Object to Framebuffer Object (gl.bindFramebuffer(),
gl.framebufferTexture2D())

- Set Renderbuffer Object to Framebuffer Object

Table of Contents

`(gl.framebufferRenderbuffer())`

Check Configuration of Framebuffer Object

`(gl.checkFramebufferStatus())`

Draw Using the Framebuffer Object

Display Shadows

How to Implement Shadows

Sample Program (Shadow.js)

Increasing Precision

Sample Program (Shadow_highp.js)

Load and Display 3D Models

The OBJ File Format

The MTL File Format

Sample Program (OBJViewer.js)

User-Defined Object

Sample Program (Parser Code in OBJViewer.js)

Handling Lost Context

How to Implement Handling Lost Context

Sample Program (RotatingTriangle_contextLost.js)

Summary

A. No Need to Swap Buffers in WebGL

B. Built-in Functions of GLSL ES 1.0

Angle and Trigonometry Functions

Exponential Functions

Common Functions

Geometric Functions

Matrix Functions

Vector Functions

Table of Contents

Texture Lookup Functions

C. Projection Matrices

Orthogonal Projection Matrix

Perspective Projection Matrix

D. WebGL/OpenGL: Left or Right Handed?

Sample Program CoordinateSystem.js

Hidden Surface Removal and the Clip Coordinate System

The Clip Coordinate System and the Viewing Volume

What Is Correct?

Summary

E. The Inverse Transpose Matrix

F. Load Shader Programs from Files

G. World Coordinate System Versus Local Coordinate System

The Local Coordinate System

The World Coordinate System

Transformations and the Coordinate Systems

H. Web Browser Settings for WebGL

Glossary

A

B

C

D

F

G

Table of Contents

H

I

L

M

N

O

P

R

S

T

U

V

W

References

Index