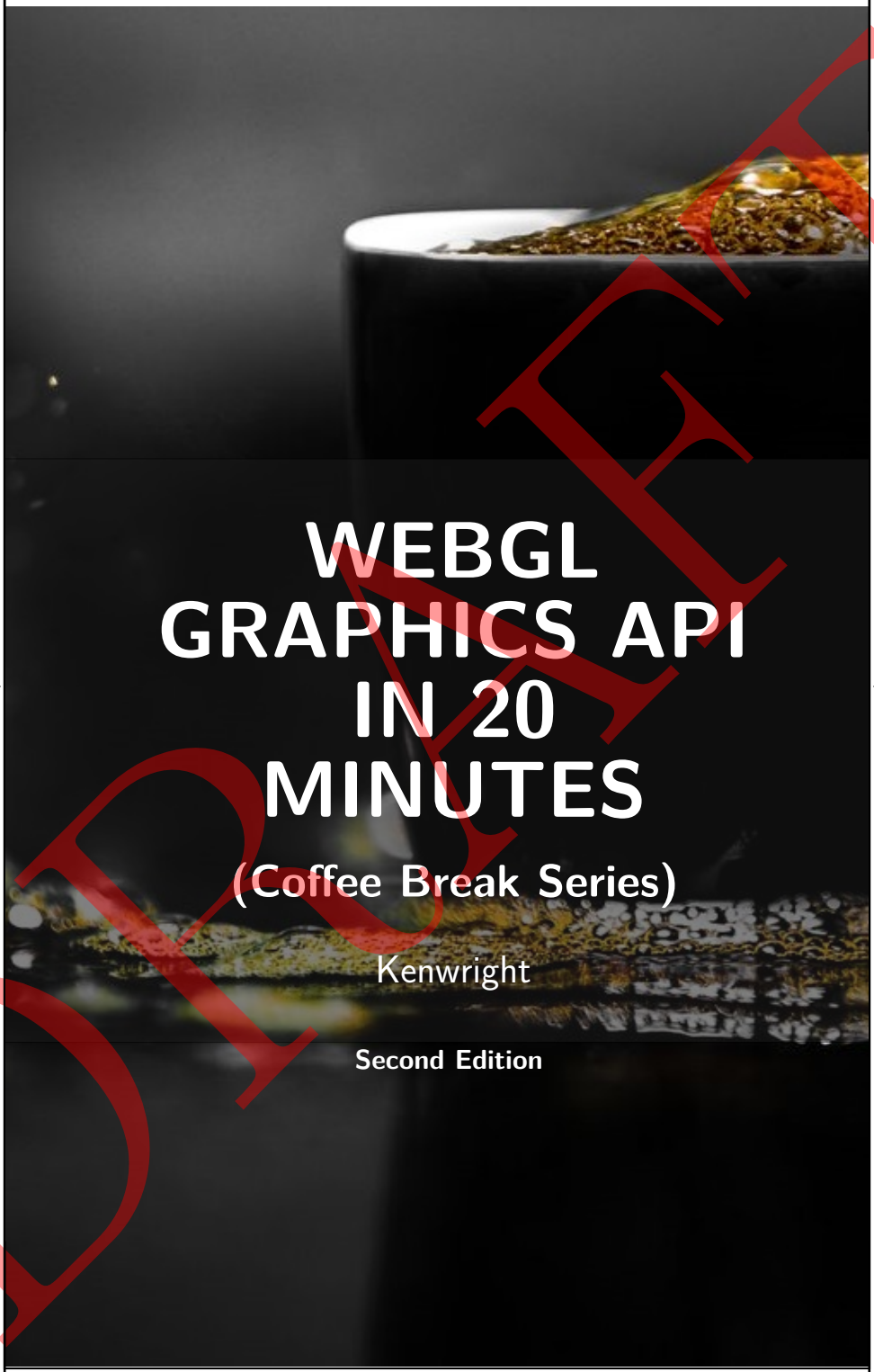

WEBGL GRAPHICS API IN 20 MINUTES

(Coffee Break Series)

Kenwright

Second Edition

BOOK TITLE:
WebGL Graphics API in 20 Minutes
Second Edition
ISBN-13: 979-8-78-834482-9

(Second Edition)

Preface

WebGL (Web Graphics Library) is a powerful JavaScript API for rendering interactive 3D and 2D computer graphics within any compatible web browser without the use of plug-ins. This incredible API is integrated into nearly every standard browser today - allowing GPU accelerated graphics, image processing effects, physical simulations and more. WebGL is so powerful and light on its feet that it's easy to forget it's a web-based API. WebGL is a must-learn for its flexibility, power and beauty. Works in a bold and poignant way, providing you the kind of results that will leave you amazed. WebGL twists the browser's abilities into remarkably uncomfortable places to take you to a higher level (spin-tingling visual effects). Sometimes it's difficult to believe the amazing things WebGL can do these days, compared to even a few decades ago. But WebGL just keep getting stronger. Sometimes, words just don't do the trick. As you learn WebGL, you'll adjust your expectations - from interested to impressed and eventually awe of what is possible with WebGL. Learn WebGL in 20 Minutes is an ambitious guide that helps you get started on the road of WebGL development. Simplified examples with no-nonsense explanations, this book guides you through a series of chapters, progressively introducing you to the concepts and applications of WebGL (including tips, code listings and visual illustrations). This introductory text is a practical guide to finding your way to getting started with WebGL in a world where we never have enough time. The book explains simple yet important concepts and practices that you can incorporate into your own real-world programs.

Table of Contents

1 **Introduction** 11

1.1 **What is WebGL?** 11

1.2 **What is special about WebGL?** 12

1.3 **Why Learn WebGL?** 12

1.4 **What are the differences between WebGL and OpenGL?** 13

1.5 **Can you implement high performance solutions with WebGL? (e.g., Games)** 14

1.6 **Is WebGL well supported? (Limitations)** 15

1.6.1 Limitations 16

1.6.2 Test if your Browser supports WebGL 18

1.7 **WebGL 1.0 vs WebGL 2.0** 18

1.7.1 Is WebGL dead with the arrival of the WebGPU API? . 21

1.8 **What this book is NOT** 21

1.9 **Checklist** 23

1.10 **Summary** 25

2 Getting Started 27

2.1 Tools (Editor) 27

2.1.1 Using your Browser's Development Tools 28

2.2 Browser Languages 29

2.2.1 HTML, CSS and JavaScript 29

2.2.2 JavaScript 29

2.2.3 'strict mode' 30

2.2.4 Inspecting the Code 31

2.2.5 Example 32

2.3 WebGL API 33

2.4 WebGL vs Canvas 33

2.5 WebGL Helpers 34

2.6 Checking Errors 35

2.7 What version of WebGL do you have? 36

2.8 Roadmap 37

2.9 Summary 37

3 Initializing WebGL 38

3.1 Your First WebGL Program (Hello WebGL) 38

3.2 Drawing Squares 40

3.3 Graphical Concepts 41

3.4 Summary 42

4 Shaders 43

4.1 What are Shaders? (GPU Language) 43

4.2 Do we need Shaders? 44

4.3 What can you do with Shaders? 45

4.4 What is the Graphics Pipeline? 45

4.5 What language are Shaders written in? 48

4.6 Vertex Shaders 48

4.7 Fragement (Pixel) Shaders 50

4.8 Uniforms and Attributes 51

4.9 Shader Tools 52

4.10 Summary 52

5 Primitives 54

5.1 Points, Lines and Triangles 54

5.2 Points 55

5.2.1 Single Point 55

5.2.2 Multiple Points 58

5.3 Lines 60

5.4 Triangles 61

5.4.1 Front and Back 63

5.5 Square (2 x Triangles) 65

5.6 Cube (12 x Triangles) 70

5.7 Summary 74

6 Texture Mapping 76

6.1 Texture Coordinates 76

6.2 Procedural Texture Applied to a Triangle 78

6.3 Load Texture from File (test.png) 81

6.4 Debugging/Rendering to Texture 85

6.5 Summary 86

7 Loading Models 87

7.1 Why Load Geometry? 87

7.2 JSON File Format 88

7.3 Summary 91

8 Lighting 92

8.1 Why is Lighting Important? 93

8.2 Cube with Lighting 94

8.3 Summary 100

9 User Input 102

9.1 Mouse Input 103

9.2 Keyboard Input 103

9.3 HTML Elements 104
9.4 Summary 104
10 Debugging WebGL 105
10.1 Debugging WebGL 105
10.2 WebGL code working (no errors) but only showing a blank page 106
10.3 WebGL unavailable error when opening in a browser 106
10.4 Is your Browser driver up to date? 107
10.5 Blacklist and Whitelist of Supported WebGL Hardware/Systems 107
10.6 Summary 107

1. Introduction

This book introducing the reader (you) to the WebGL cross platform 2D/3D graphics API (a taste of WebGL) including simple examples and tutorials. You'll address questions, such as, how do you get started with WebGL? what is special about WebGL? how is WebGL different from other graphical libraries (DirectX or OpenGL)? and how do you initialize and setup basic WebGL programs in JavaScript?

1.1 What is WebGL?

To begin with, let's talk about what WebGL is and what it has to do with computer graphics.

WebGL is a graphics application programming interface (API) created for use in web applications. It is based off the open graphics language (OpenGL) embedded standard (ES). WebGL is used by developers to provide a platform-independent

means of creating interactive graphical applications on the web. WebGL is not only used to draw the graphics of 2D and 3D games, but also to accelerate the functions of web based image editors and their effects, as well as physics simulations.

Although WebGL is functionally based off OpenGL ES, it is partly written in JavaScript. WebGL is used to render interactive 2D and 3D graphics in compatible web browsers. The API allows users to experience interactive content on webpages, with GPU acceleration, without having to first download or install any plug-ins. For developers, WebGL provides low-level access to hardware with the familiar code structure of OpenGL ES.

1.2 What is special about WebGL?

Unlike previous graphical predecessors (OpenGL and DirectX), WebGL is designed to run in your browser. Making it one of the most accessible graphical API out their. The underlying design of the API is layered (to be more structured and modular), so it enables the creation of common, yet extensible architectures for code validation, debugging, and profiling, without impacting performance.

1.3 Why Learn WebGL?

WebGL is like DirectX and OpenGL which targets high-performance real-time 3D graphics applications such as games and interactive media across all platforms - offering high performance and low CPU usage. In addition to WebGL's performance factors, WebGL can be distribute across multiple systems easily (runs in the browser).

WebGL can allow you to perform tasks that are simply not possible with other technologies. This makes it suitable for cross-platform complex visualizations and games. My hope is that you'll take a look at WebGL, even if it's just for fun

(WebGL is sexy).

This short text helps you get started right away (give you some pointers on where to start, what is important, pitfalls, essential facts and where to go next). While you might find WebGL challenging initially if you don't have a graphics or programming background, you will find it extremely rewarding in the long term.

While some people might think that WebGL means you're need to master other front-end concepts, such as, HTML and CSS, this is not true, WebGL runs independently within its own 'window'. You'll start your browser, create a 'window' for your WebGL application, and off you go.

Having knowledge of Maya or 3ds Max helps (as you'll be able to connect with concepts such as vectors, matrices and lighting types) however, this is not essential. There are a number of libraries built on top of WebGL (e.g., three.js), however, going the pure WebGL route is fun, educational and exciting.

1.4 What are the differences between WebGL and OpenGL?

WebGL is made for browsers, whereas OpenGL needs native drivers, and is oriented to installed software. OpenGL is used in many video games. WebGL also lacks certain capabilities that OpenGL has, as it is based on OpenGL ES. WebGL lacks things like 3D textures, geometry shaders, etc. However, WebGL can fake a 3D texture using a 2D texture.

One of the major differences between WebGL and OpenGL is that WebGL is based on OpenGL ES which lacks many of the features that regular OpenGL has. For example WebGL only supports vertex and fragment shaders whereas OpenGL has those plus geometry shaders, tessellation shaders, and compute shaders. There are a number of other features that

OpenGL has that WebGL does not have such as 3D textures, vertex array objects, and instanced drawing (available by extension in some browsers). Another difference is that in WebGL there is no fixed function pipeline. This is a good thing because in OpenGL many people still learn the fixed function pipeline even though it has been deprecated for many years now so in learning WebGL you are forced to learn to use shaders and buffers right from the start. Basically if you learn WebGL you will likely be able to pick up OpenGL easily since the function names and parameters are very similar and pretty much all of the features supported in WebGL are supported in OpenGL.

1.5 Can you implement high performance solutions with WebGL? (e.g., Games)

WebGL 2.0 (current, in Chrome and Firefox) implements OpenGL ES 3.0, including GLES 3.0 for shaders. So it can do the same as 3D graphics with those capabilities in general, what you have in games and other native 3D apps - ES as in embedded systems, subset of OpenGL for mobile devices. So that's the power in terms of features.

Compared to native code, WebGL is slower because the execution engine in the Web browser does extra security checks (array bounds and such) so the code that deals with the graphics memory can not break into the host operating system. It would be interesting to know these bottlenecks better,

So what does that mean in practice? Good question, and hard to summarize briefly. One way to think simply is that compare with some fancy 3D iPad or Android mobile game, maybe it uses the same OpenGL ES 3.0, and a native app on mobile hardware is perhaps similar in performance to what you get on a more powerful laptop or PC in the browser, with the security overhead - supposing you have a good gfx chip there, like mobile devices nowadays do. Or maybe think of a

5–10 year old top PC 3D game.

Besides WebGL vs native GL (or DirectX), another difference is having your application code in JavaScript. Now that has changed with WebAssembly (WASM) which runs code compiled from e.g. C++ for the browser basically the same speed as native code runs. These are exciting times.

1.6 Is WebGL well supported? (Limitations)

As it happens, not all browsers are created equal. Historically, WebGL was shipped by only two browser vendors, Mozilla and Google, while other vendors were reluctant to implement then state-of-the-art technology. Things drastically changed after Apple and, finally, Microsoft joined the 3D Web club. Today, almost all Internet traffic goes through WebGL-enabled client software, making this technology a robust and widespread foundation for delivering interactive 3D content to 4+ billion of users.

Figure 1.1: ***All Browsers*** *- WebGL Browser Support - Any web browser worth its salt should support the WebGL specification.*

- **Google Chrome** was the first browser that introduced 3D support back in 2011. Being the most popular user agent, it comes with the most advanced and fast WebGL implementation. Prefer using it for the best experience with 3D web applications.
- **Google Chrome Mobile** which is now widespread across Android devices, also offers a good WebGL implementation. The only exception, however, is Chrome for iOS, which is