



Learning Virtual Reality

DEVELOPING IMMERSIVE EXPERIENCES AND APPLICATIONS
FOR DESKTOP, WEB, AND MOBILE

Tony Parisi

Learning Virtual Reality

*Developing Immersive Experiences and
Applications for Desktop, Web, and Mobile*

Tony Parisi

Beijing • Boston • Farnham • Sebastopol • Tokyo

O'REILLY®

Learning Virtual Reality

by Tony Parisi

Copyright © 2016 Tony Parisi. All rights reserved.

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (<http://safaribooksonline.com>). For more information, contact our corporate/institutional sales department: 800-998-9938 or corporate@oreilly.com.

Editors: Simon St.Laurent and Meg Foley

Production Editor: Colleen Lobner

Copyeditor: Rachel Head

Proofreader: Charles Roumeliotis

Indexer: Ellen Troutman-Zaig

Interior Designer: David Futato

Cover Designer: Karen Montgomery

Illustrator: Rebecca Demarest

November 2015: First Edition

Revision History for the First Edition

2015-10-22: First Release

2016-08-31: Second Release

See <http://oreilly.com/catalog/errata.csp?isbn=9781491922835> for release details.

The O'Reilly logo is a registered trademark of O'Reilly Media, Inc. *Learning Virtual Reality*, the cover image of a Salvin's prion, and related trade dress are trademarks of O'Reilly Media, Inc.

While the publisher and the author have used good faith efforts to ensure that the information and instructions contained in this work are accurate, the publisher and the author disclaim all responsibility for errors or omissions, including without limitation responsibility for damages resulting from the use of or reliance on this work. Use of the information and instructions contained in this work is at your own risk. If any code samples or other technology this work contains or describes is subject to open source licenses or the intellectual property rights of others, it is your responsibility to ensure that your use thereof complies with such licenses and/or rights.

978-1-491-92283-5

[LSI]

Table of Contents

Preface	vii
1. Introduction	1
What Is Virtual Reality?	2
Stereoscopic Displays	3
Motion Tracking Hardware	5
Input Devices	5
Computing Platforms	6
Virtual Reality Applications	8
Chapter Summary	9
2. Virtual Reality Hardware	11
Oculus Rift	11
The DK1	12
The DK2	13
The CV1	14
Setting Up Your Oculus Rift	16
Other High-End Head-Mounted Displays	17
Samsung Gear VR: Deluxe, Portable Virtual Reality	18
The Oculus Mobile SDK	20
Google Cardboard: Low-Cost VR for Smartphones	20
Stereo Rendering and Head Tracking with Cardboard VR	22
Cardboard VR Input	22
Developing for Google Cardboard	23
VR Input Devices	23
Chapter Summary	25

3. Going Native: Developing for Oculus Rift on the Desktop	27
3D Graphics Basics	28
3D Coordinate Systems	28
Meshes, Polygons, and Vertices	29
Materials, Textures, and Lights	30
Transforms and Matrices	31
Cameras, Perspective, Viewports, and Projections	32
Stereoscopic Rendering	34
Unity3D: The Game Engine for the Common Man	34
Setting Up the Oculus SDK	37
Setting Up Your Unity Environment for Oculus Development	38
Building Your First VR Example	39
Building and Running the Application	41
Walking Through the Code	43
Chapter Summary	46
 4. Going Mobile: Developing for Gear VR.	 49
The Gear VR User Interface and Oculus Home	50
Using the Oculus Mobile SDK	51
Setting Up the Android SDK	52
Generating an Oculus Signature File	52
Setting Up Your Device for USB Debugging	52
Developing for Gear VR Using Unity3D	53
Setting Up Your Unity3D Environment	53
A Simple Unity3D Sample	54
Handling Touchpad Events	59
Deploying Applications for Gear VR	62
Chapter Summary	62
 5. WebVR: Browser-Based Virtual Reality in HTML5.	 63
The Story of WebVR	65
The WebVR API	66
Supported Browsers and Devices	67
Querying for VR Displays	68
Presenting Content to the VR Display	69
Refreshing the VR Display	70
Head Tracking	72
Creating a WebVR Application	73
Three.js: A JavaScript 3D Engine	73
A Full Example	74
Tools and Techniques for Creating Web VR	85
WebVR Engines and Development Tools	85

Using Unity3D and Unreal for WebVR Development	87
Open Source Libraries and Frameworks	88
WebVR and the Future of Web Browsing	89
Chapter Summary	90
6. VR Everywhere: Google Cardboard for Low-Cost Mobile Virtual Reality.....	93
Cardboard Basics	95
Supported Devices and Operating Systems	95
Headset Manufacturers	95
Cardboard Applications	97
Input Devices for Cardboard	99
Cardboard Stereo Rendering and Head Tracking	100
Developing with the Google VR SDK for Android	101
Setting Up the Environment	102
Walking Through the Code	103
Developing with the Google VR SDK for Unity	107
Setting Up the SDK	108
Building Treasure Hunt for Unity	109
A Walkthrough of the Unity Code	111
Developing Cardboard Applications Using HTML5 and a Mobile Browser	114
Setting Up the WebVR Project	114
The JavaScript Cardboard Code	115
Chapter Summary	118
7. Your First VR Application.....	119
About 360-Degree Panoramas	120
Setting Up the Project	121
Getting the Software, Hardware, and Sample Code	121
Creating the Unity Scene and Project	122
Adding Cardboard VR Support	128
Creating a Gaze-and-Tap User Interface	131
Creating a 2D Plane	132
Adding Input Support to the Project	132
Creating an Event Handler Script	134
Handling Switch Events	136
Creating the Functioning User Interface	137
Where to Take the Project from Here	138
Chapter Summary	139
A. Resources.....	141
Index.....	149

Preface

Virtual reality has captured the world's imagination. Over the last few years, developers and enthusiasts in the thousands have devoted countless hours to coding, designing, and speculating about the possibilities of this exciting new medium. Affordable hardware systems like the *Oculus Rift*, *Samsung Gear VR*, and *Google Cardboard VR* allow consumers to experience virtual reality firsthand in the comfort of their homes, in a cafe, or on the train to work. Press coverage has reached beyond the trade press and blogosphere to mainstream publications extolling the virtues of VR, and the possibilities seem endless: from gaming and cinema to architecture, education, training, and medicine. Even though it has a long way to go, virtual reality appears poised to become the next major entertainment medium, and perhaps even the computing platform of the future.

I first tried an Oculus Rift in the summer of 2013. I was, to put it bluntly, underwhelmed. The graphics of the original development kit (the so-called “DK1”) were low-res by today's standards, and I lasted about 10 minutes before the nausea kicked in—which, according to my friend Dave, was an impressive display of endurance. I put the device aside, grumbled “not ready,” staggered back to work, and didn't give the matter much thought again until nine months later, when the tech industry was set on its ear by the announcement that social media giant Facebook had acquired Oculus VR, the makers of the Rift. Like many of my peers, I was stunned. Also, like many of my peers, I decided to jump right in and start developing something in VR.

Since the watershed Oculus acquisition, the industry has invested millions in developing applications, content, tools, display hardware, video capture systems, and peripherals. Big tech players are already staking out their turf, making bets on what the future will look like with respect to distribution channels, “killer apps,” and such, and creating platforms and ecosystems to align with their visions. Developers of all shapes and sizes are flocking to VR, some out of genuine excitement for a new medium with great potential, and others preparing to take advantage of what could be the next tech boom after mobile. However it all ultimately shakes out, being a VR developer promises to be an exciting ride.

Every journey begins with a first step, and this book is here to set you on your path. It's not deep; the goal is to familiarize you with core programming concepts, and the innovations in hardware and software that have made VR possible. It is broad: we cover three of the major platforms, using three different development environments and as many programming languages. By the end, you should come away with the feeling that you understand the basics, and a desire to learn more.

Audience

This book was written for programmers and designers looking for an introduction to virtual reality development. It assumes at least entry-level programming experience, but you don't have to be a professional developer to read it. Creative coders, producers with a programming background, and technically savvy artists should also be able to follow along. I want anyone who likes to make things on a computer to be able to walk away having learned something from this book.

Readers should know the basics of JavaScript, Java, C#, or another C-family programming language. Experience with 3D graphics is also helpful. If you don't have 3D experience, Chapter 3 contains a primer that you may find useful.

If you are a professional developer, a lot of this material will seem basic to you. But go through it carefully: interleaved with the how-tos and 101s are essential nuggets of technical information for putting together desktop, mobile, and web VR apps. It represents many hours of carefully working through the details of various tools, SDKs, APIs, and operating systems, including suffering through a few major upgrades that were sprung on me halfway through writing the manuscript. Maybe my effort will help save you from hitting those same land mines along the way.

If you are a native developer of mobile and desktop apps, and know Unity3D or another game engine, then this book should help you extend your skills into VR. If you are comfortable developing for WebGL, then you should have an easy time creating VR for the Web with the introduction of just a few new concepts. If you are a newbie at both, no worries; I like to think I wrote the book in a way that will get you going, no matter your starting point.

How This Book Is Organized

This book is divided into seven chapters, as follows:

- Chapters 1 and 2 provide an introduction to virtual reality concepts and survey the new hardware systems coming to market.
- Chapters 3 through 6 cover virtual reality development in detail. This is the heart of the book, in which we look at developing for three of the major virtual reality hardware systems: the Oculus Rift, Gear VR, and Cardboard VR. These chapters

explore different tools for creating VR applications, including Unity3D and Android Studio, as well as several programming languages. There is also a chapter about creating web-based VR applications using WebVR, a new JavaScript API supported in development versions of popular browsers like Firefox and Chrome.

- Chapter 7 brings the concepts introduced in Chapters 3 through 6 together in a simple working application: a 360-degree panoramic VR photo viewer for Google Cardboard, built in Unity3D.

Updates for the Revised Edition

Consumer virtual reality is still new, and like many emerging technologies, it's moving fast. Since its original publication, several major changes have been made to the VR headsets and SDK software described in this book. The Revised Edition incorporates those changes. We can hope that the information in here will change much less frequently, now that commercial versions of the hardware and software are out in the market.

The Revised Edition has the following major changes:

Changes to the hardware

Oculus, Samsung, Sony, and HTC have shipped consumer versions of their hardware. The names of these products and any associated descriptions and screenshots have been updated to reflect this.

Changes to the Oculus SDK for Windows

The setup details and the API for the Oculus SDK for Windows have changed since the original publication; Chapter 3 has been updated to reflect these changes.

Changes to the Oculus Mobile SDK

The setup details and the API for the Oculus Mobile SDK for Gear VR have changed since the original publication; Chapter 4 has been updated to reflect these changes.

Changes to the WebVR API

The WebVR API went through a major revision since the original publication; those details are now reflected in the Revised Edition (see Chapter 5).

Changes to the Google VR SDK

The “Cardboard SDK” is now the “Google VR SDK.” In addition to renaming, some details of the SDK changed. Chapters 6 and 7 of the Revised Edition contain updated information on both the Android native and Unity versions of the SDK.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic

Indicates new terms, URLs, email addresses, filenames, and file extensions.

Constant width

Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, objects, parameters, and values.

Constant width bold

Shows commands or other text that should be typed literally by the user.

Constant width italic

Shows text that should be replaced with user-supplied values or by values determined by context.



This element signifies a general note.

Using Code Examples

Supplemental material for this book is available for download at <https://github.com/tparisi/LearningVirtualReality> and <https://github.com/tparisi/WebVR>.

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Learning Virtual Reality* by Tony Parisi (O'Reilly). Copyright 2016 Tony Parisi, 978-1-4919-2283-5."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at permissions@oreilly.com.



About Third-Party Copyrighted Material

Note that some of the content assets used in this book are subject to copyright. Their creators have kindly granted me permission to redistribute them for use with the book for the *sole* purpose of supporting the programming examples included. For any other purpose, including and especially use in your applications, you must obtain your own copies of those assets, which may include purchasing a license.

Safari® Books Online



Safari Books Online is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business.

Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training.

Safari Books Online offers a range of plans and pricing for enterprise, government, education, and individuals.

Members have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and hundreds more. For more information about Safari Books Online, please visit us online.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at <http://bit.ly/learning-virtual-reality>.

To comment or ask technical questions about this book, send email to bookquestions@oreilly.com.

For more information about our books, courses, conferences, and news, see our website at <http://www.oreilly.com>.

Find us on Facebook: <http://facebook.com/oreilly>

Follow us on Twitter: <http://twitter.com/oreillymedia>

Watch us on YouTube: <http://www.youtube.com/oreillymedia>

Acknowledgments

It takes a village to make a good VR book, and I was able to lean on some great developers to get my job done. “That VR Guy” Dave Arendash and Alex Sink of RIVER Studios provided much-needed expert information on using Unity3D. Patrick Chen and Andrew Dickerson from Samsung lent early moral support (and a Gear VR) to get me going with mobile development. The WebVR brain trust, including Josh Carpenter, Vlad Vukićević, Diego Marcos, and Brandon Jones, have been stalwart, always ready with advice and answers to questions, and always willing to consider an API design change.

I am grateful for the excellent technical reviews done for the book, especially the review by my friend and startup collaborator Jason Marsh. Jason worked through the code examples in painstaking detail, and identified several critical issues. Jason also contributed the beautiful panoramic photos featured in the example in Chapter 7.

I would like to thank the team at O’Reilly, especially my editor, Meg Foley. Meg was exceedingly patient with my halting progress due to conflicting startup commitments. She also gave me much-needed encouragement when I hit a snag and had to rewrite several chapters due to massive changes in the Oculus SDK software. That’s life on the bleeding edge, I guess, but nevertheless, Meg has been a champ.

Finally, thanks to my family. Marina and Lucian are used to my book-writing antics by now, but each one is a new exercise in forbearance. Guys, you’re the best.

Introduction

Virtual reality is a medium with tremendous potential. The ability to be transported to other places, to be fully immersed in experiences, and to feel like you're really there—*present*—opens up previously unimagined ways to interact and communicate. Until recently, virtual reality was out of reach for the average consumer due to cost and other factors. However, advances in the technology over the last few years have set the stage for a mass-market revolution that could be as influential as the introduction of television, the Internet, or the smartphone.

Virtual reality—VR for short—comprises a collection of technologies: 3D displays, motion tracking hardware, input devices, software frameworks, and development tools. While consumer-grade VR hardware is young and evolving, a handful of platforms have emerged as go-to choices, including the Oculus Rift, Samsung Gear VR, and Google Cardboard. Each delivers a different level of VR experience, at a different price point, with varying degrees of in-your-hands portability.

Software to create and display consumer virtual reality is also coming together rapidly. The Unity3D and Unreal game engines, popular for making desktop and mobile games, have become tools of choice for native VR development. And the Web is not far behind: WebGL and 3D JavaScript frameworks like Three.js and Babylon.js are providing a path for creating open source, browser-based virtual reality experiences for desktop and mobile operating systems.

It's an exciting time! With so much energy going into development, and so much consumer interest, VR just might be the next big wave of computer technology. In this book, we explore the hardware, software, application techniques, and interface design challenges encountered by today's virtual reality creator. Virtual reality is still in its early stages. It's a lot like the Wild West, and you are a pioneer. The landscape may be fraught with unknowns, even dangers—but we push on, driven by the promise of a better life. Let's take a peek at this new frontier.

Figure 1-1 shows a screenshot of the now-famous Tuscany VR demo, created by the team at Oculus VR to show off their hardware. Put on the Oculus Rift and launch the demo. You are on the grounds of a Tuscan estate, looking at a beautiful villa. Clouds drift lazily across the sky. You hear birds chirping, and the sound of waves lapping gently against a shore.

You move through the scene, video game-style, using the W, A, S, and D keys on your keyboard (known to gamers as the “WASD keys”). If you play a lot of PC games, this is nothing new. But now, turn your head: looking up, down, and behind, you can see the entire estate. You are there, immersed in a virtual world that completely surrounds you. Walk forward, into the villa, and take a look around. Walk out, up to the edge of the property, and see the lake below. For a few moments at least, you forget that you are not actually in this other place. You’re *present*.

This feeling of total immersion—of being somewhere else, experiencing something else entirely—is what we are striving for with virtual reality. And this is where our journey begins.



Figure 1-1. Tuscany VR demo by the Oculus VR team

What Is Virtual Reality?

Reality is merely an illusion, albeit a very persistent one.

—Albert Einstein

Virtual reality has one goal: to convince you that you are somewhere else. It does this by tricking the human brain—in particular, the visual cortex and parts of the brain that perceive motion. A variety of technologies conspire to create this illusion, including:

Stereoscopic displays

Also known as *3D displays* or *head-mounted displays* (HMDs), these displays use a combination of multiple images, realistic optical distortion, and special lenses to produce a stereo image that our eyes interpret as having three-dimensional depth.

Motion tracking hardware

Gyroscopes, accelerometers, and other low-cost components are used in virtual reality hardware to sense when our bodies move and our heads turn, so that the application can update our view into the 3D scene.

Input devices

Virtual reality is creating the need for new types of input devices beyond the keyboard and mouse, including game controllers and hand- and body-tracking sensors that can recognize motion and gestures.

Desktop and mobile platforms

This includes the computer hardware, operating systems, software to interface to the devices, frameworks and engines that run applications, and software tools for building them.

Without all four of these components, it is hard to achieve a fully immersive virtual reality experience. We will dive further into the details throughout the book; for now, let's take a quick look at each.

Stereoscopic Displays

The main ingredient in virtual reality is a persistent 3D visual representation of the experience that conveys a sense of depth. To create this depth, virtual reality hardware systems employ a 3D display, also known as a stereoscopic display or head-mounted display.

For years, one of the biggest impediments to consumer-grade virtual reality was an affordable stereoscopic display that is light and comfortable enough to be worn for an extended period. This situation changed dramatically when the team from Oculus VR created the Oculus Rift. First introduced in 2012, the Rift was a breakthrough in VR hardware, featuring a stereoscopic display and a head-tracking sensor built into a lightweight headset that could be purchased as a development kit for a few hundred dollars. While the original development kit, known as the DK1, was fairly low-resolution, it was enough to get the entire industry excited and unleash a storm of VR development. Newer Rift development kit versions, such as the DK2 depicted in Figure 1-2, feature higher display resolution, position as well as orientation tracking, and better performance.



Figure 1-2. The Oculus Rift head-mounted display, Development Kit 2

So what does the Oculus Rift actually do? To create the illusion of depth, we need to generate a separate image for each eye, one slightly offset from the other, to simulate *parallax*—the visual phenomenon where our brains perceive depth based on the difference in the apparent position of objects (due to our eyes being slightly apart from each other). To create a really good illusion, we also want to distort the image to better emulate the spherical shape of the eye, using a technique known as *barrel distortion*. The Oculus Rift does both.

A more accurate screenshot of the Tuscany VR demo, a capture of the entire display that renders this scene in virtual reality using these two techniques, would look like Figure 1-3. This shows the rendering as it appears on a computer screen connected to the Oculus Rift's head-mounted display.

From a software point of view, an Oculus Rift application's job is to render an image like the one in Figure 1-3 a minimum of 60 and ideally 120 times per second, to avoid any perceived lag, or *latency*, that might break the illusion or, worse, lead to the nausea that is often associated with poorly performing VR. Exactly how we do this in our applications is the subject of another chapter.

Note that the Oculus Rift is not the only game in town. As we will see in the next chapter, there are several head-mounted displays to choose from. Some of these work only with desktop computers, others are just for smartphones, and still others can only be used with game consoles. The HMDs come in a variety of styles and a range of prices. But as consumer-ready virtual reality displays go, Oculus is the first and—as of this writing—still the best.



Figure 1-3. *Tuscany VR demo rendered in stereo for the Oculus Rift*

Motion Tracking Hardware

The second essential trick for making the brain believe it is in another place is to track movements of the head and update the rendered scene in real time. This mimics what happens when we look around in the real world.

One of the innovations in the Oculus Rift is its rapid head motion tracking using a high-speed *inertial measurement unit* (IMU). Head-tracking IMUs combine gyroscope, accelerometer, and/or magnetometer hardware, similar to that found in smartphones today, to precisely measure changes in rotation. The VR hardware systems covered in the next chapter employ a variety of IMU configurations.

Motion tracking of the head is as important as quality stereo rendering, if not more so. Our perceptual systems are very sensitive to motion, and like lag in stereo rendering, high latency in head tracking can break the feeling of immersion and/or cause nausea. Virtual reality IMU hardware must track head movement as rapidly as possible, and the software must keep up. When stereo rendering and head motion tracking are properly combined, and updated with enough frequency, we can achieve a true feeling of being immersed in the experience.

Input Devices

To create a convincing feeling of immersion, head-mounted displays completely enclose the user's eyes, cutting them off from seeing the outside world. This makes for an interesting situation with respect to input: users have to "fly blind"; they cannot see their mouse or keyboard when using the VR display. To address this, VR is mak-

ing use of alternate types of input devices, including game controllers, hand-tracking motion sensors, and wireless hand and body trackers.

There is currently no one standard way to interact, no “mouse of virtual reality,” if you will. VR is driving a lot of experimentation and innovation with respect to input devices, and we are only at the beginning of that process. Over the next few years, we may see a complete transformation of how we interact with our computers, thanks to VR.

Computing Platforms

Many VR applications will run on a majority of existing computers and mobile phones. A relatively modern desktop or high-powered laptop can do the trick with Oculus Rift; smartphones can also offer a good VR experience, provided they have enough CPU and graphics power. For most of us, this means that our existing computers and devices can be turned into virtual reality boxes simply by adding a few peripherals. But for those seeking super high production value experiences, the latest desktop PC with a best-of-breed graphics processor and the fastest available CPU might be the top item on the wish list for next Christmas.

As VR matures and gains in popularity, we may also begin to see dedicated computers, phones, and consoles—dedicated, that is, to the sole purpose of enabling amazing virtual reality.

Software to create VR applications comes in several flavors: native software development kits, game engines and frameworks, and even the latest versions of modern web browsers. VR video is another area of exploration. Let’s take a closer look at each of these development options.

Native software development kits (SDKs)

These are the device drivers and software libraries used in conjunction with the computer’s host operating system. On Windows they would be Win32 libraries used in C++ applications; on Android, they would be Java libraries; and so on. You can build a native application simply using the SDKs, and “roll your own” with respect to everything else (such as 3D graphics and game behaviors). But most developers use engines or frameworks.

Game engines and frameworks

Unless you are a game engine developer, you probably won’t want to deal with the native SDKs directly; you will more than likely turn to a game engine such as Unity3D (described in detail in Chapter 3). Libraries like Unity3D, also known as *middleware*, take care of the low-level details of 3D rendering, physics, game behaviors, and interfacing to devices. Most VR developers today build their apps using game middleware like Unity3D.

Many middleware engines have strong cross-platform support, allowing you to write your code once (most of it, at least) and target multiple platforms, including desktop and mobile. They also usually come with a powerful set of tools known as *level editors* or *integrated development environments* (IDEs).

Web browsers

Much in the way that HTML5 added mobile features over a short number of years, nearly achieving parity with native mobile capability, browser makers are fast following the development of virtual reality. In the case of VR, the adoption of features into web browsers is looking like it will only take one or two years, not four to five.

The upshot of this is twofold: first, it means that we can use web technologies like HTML5, WebGL, and JavaScript to create our applications, making it potentially faster to code and more cross-platform; second, it affords us access to all of the existing infrastructure the Web has to offer, such as hyperlinking between VR experiences, hosting content in the cloud, developing multiuser shared experiences, and integrating web data directly into our virtual reality applications.

Video players

Stereoscopic video represents a whole class of virtual reality technology by itself. Unlike with game engines, where the application's graphics are completely synthetic, based on handcrafted 3D models, animations, backgrounds, and so on, stereo video is captured from the real world. This makes for truly realistic and often stunning experiences: imagine taking a virtual helicopter tour of the Grand Canyon in VR, being able to look around as you fly over the canyon. Video is not fully interactive in the way that a 3D virtual environment can be though, so the use of this media type has some limitations.

Stereo video recording requires multiple cameras. The minimum is two, but if we want the video to be *panoramic*—that is, capture a 360-degree view of an entire scene for use in virtual reality—then we need even more cameras. Pioneers in the field, such as California-based Jaunt VR, are experimenting with setups that use dozens of cameras to make the first VR feature-length films.

The capture and production of VR video is a nascent field. Several companies and research projects are devoting their efforts to it. There are also a variety of VR video players in development. Some players work only in native environments, while others are only for mobile or the Web, but some developers are creating full cross-platform players. One of the biggest issues in this young endeavor is that there are as yet no standard formats for storing and playing back of the videos, so if you want to produce video content, you may have to choose a single vendor for the hardware, the production tools, and the playback software.

Virtual Reality Applications

Even though consumer virtual reality is just a few years old, we are already seeing a staggering range of applications. To say that VR has captured people's imaginations would be an understatement. Developers are trying to build virtually *everything* in VR, understandably with mixed success so far.

While it is way too early to pick winners and identify "killer apps" for VR, there are several domains that show great promise, including:

Video games

This is an obvious candidate, the one that most people immediately imagine when you tell them about virtual reality. The potential for deep immersion, higher production value, and stickier engagement has developers, console makers, and peripheral manufacturers salivating in anticipation. It is fair to assume that the majority of skilled, independent, large development shops creating VR are currently doing so for games.

Virtual worlds

Social, user-generated persistent virtual worlds could be a powerful combination with virtual reality immersion. Companies like High Fidelity, created by *Second Life* founder Philip Rosedale, and AltSpace VR, a new San Francisco Bay Area startup, are leading this charge.

Education

For years, 3D visualization has been a great tool for interactive learning; VR immersion could make learning even more approachable and effective.

Productivity

Some researchers and small companies are exploring using VR as a replacement for the decades-old desktop computer metaphor. Imagine a virtual reality, 360-degree workspace that holds your personal information, contacts, work projects, etc.

Tourism

Stereoscopic 360-degree panoramas in VR are really compelling. They represent a simple, effective way to convey the experience of being elsewhere without having to get on a plane.

Architecture and real estate

Architecture and real estate firms are already experimenting with virtual reality, using both video and interactive graphics. Video can be great for showing existing properties; interactive graphics work well for visualization of buildings and complexes in the planning stages (i.e., using 3D CAD models).

Live events

VR video is promising to be quite popular for concerts, news reporting, and other live events. Musicians Paul McCartney and Jack White are among several rock stars who have already broadcast virtual reality versions of their live shows.

Web browsing

Mozilla is leading the charge in experimenting with adding VR support to its browser, and Google is not far behind in creating the same features for Chrome. Beyond building the technology plumbing, the research team at Mozilla is exploring visual and interface designs for how to navigate a universe of information in virtual reality.

Enterprise applications

There are countless potential VR applications for the enterprise, including simulation and training for military use, medical diagnostics and training, and engineering and design.

This list is just a sampling. We have no way of knowing which applications will be most successful in a new medium like virtual reality, or what will become popular. We also can't predict what other ideas will spring from the minds of clever folks—but if the last few decades of technology have taught us anything, it's that whatever people come up with, it will probably be something we never could have imagined before.

Chapter Summary

Consumer virtual reality is upon us. As we have seen in this chapter, VR brings together several technologies, including 3D stereoscopic displays, motion tracking hardware, new input devices, computers, and mobile phones. The key innovations that enable virtual reality are stereoscopic rendering and motion tracking. When these two are properly combined, we feel immersed, or *present*, and the illusion of VR is compelling enough to transport us to another place.

There are a variety of development options for virtual reality: native SDKs that access raw platform features; game engines that give us more power and cross-platform reach; web browsers, for creating shared, connected virtual reality apps; and video players and tools for creating stereoscopic panoramic videos based on the real world.

Though the technology is still immature, virtual reality is already being used in a number of applications, from games to education, from real estate to rock concerts. Developers are also creating productivity and enterprise applications, looking to the power of VR immersion to enhance business value.

In the next chapter we will explore the wide variety of virtual reality hardware systems in use today. Let's get to it.

Virtual Reality Hardware

In this chapter we will take a look at popular consumer virtual reality hardware in use today, including VR headsets and the computers or phones they work with.

Even though this is a rapidly evolving industry, some manufacturers are already becoming established as leaders. Most notable is Facebook with the Oculus Rift, but there are several other head-mounted displays (HMDs) to choose from. Some HMDs require using desktop computers, others are just for smartphones, and still others are for use with game consoles. To further muddy the picture, new HMDs are coming out all the time.

Let's examine three headsets that represent a wide range of HMDs on the market today, and that will form the focus of the rest of the chapters in the book: the Oculus Rift for desktop virtual reality; Samsung's Gear VR (based on Oculus technology) for a high-end mobile VR experience; and Google's Cardboard VR, a simple, low-cost way of adapting your existing smartphone to become a VR device.

Oculus Rift

While there have been several attempts to market consumer virtual reality headsets over the past few decades, nothing ever pushed it over the top until the Oculus Rift came along. Buoyed by an extremely successful Kickstarter campaign to fund its original development, Oculus VR raised significant venture funding, and the company was ultimately sold to Facebook in a blockbuster US\$2B acquisition that the industry is still talking about. Without that watershed event, it is unlikely that virtual reality would have gotten as popular as quickly, or captured the public's interest in the way it has. But it did, and it has, and here we are.

The Oculus Rift has spawned an entire VR industry. This includes companies large and small creating applications, tools, peripheral hardware, and development serv-

ices. Numerous applications are being built for the Oculus Rift, spanning gaming, architecture, medicine, real estate, tourism, entertainment, and education. With its voracious appetite for CPU and graphics processor cycles, the Rift has even given a shot in the arm to the declining desktop PC industry.

The Oculus Rift is a stereoscopic display with built-in head motion tracking sensors. It straps to the head, allowing hands-free operation. The Rift is a peripheral that attaches to a computer: Mac, Linux, or Windows (desktop or laptop). The Rift is tethered, with a cable running to the computer. At the moment it is quite bulky—but that will most certainly change with the newer models being designed as we speak.

The following is a brief history of the Oculus Rift, by way of exploring the evolution of this groundbreaking VR headset.



The Gold Standard of Virtual Reality

By any reckoning, as of this writing the Oculus Rift is the virtual reality headset against which all others will be measured in coming years. While the previous development kit versions felt clunky in some ways, each iteration is noticeably better than the last. Oculus VR began shipping retail consumer headsets in March 2016 and they are affordable, comfortable, and amazing.

The DK1

The original Oculus Rift development kit, known as the *DK1*, is a big affair—weighing in at over 13 ounces (.33 kg) and measuring 7 inches (18 cm) diagonal—and therefore not very comfortable for extended use. It has an external control box in addition to two sets of wires: one for the USB connection to the head tracker, and one for the video signal coming from the computer. The DK1 is depicted in Figure 2-1.

The DK1 has a resolution of 1280x800 pixels. This screen real estate is divided between the two eyes, so the effective resolution is 640x800; in other words, the DK1 is a low-resolution device by today's standards for computer monitors. The display has a horizontal field of view of well over 90 degrees, which is important for emulating the visual field of view experienced in the real world. The head tracking of the inertial measurement unit (IMU) is quite fast at 250 Hz; this is critical to enable a true sense of immersion and to reduce motion sickness.

Despite its low visual resolution, the wide field of view and rapid head tracking response time of the DK1 made it the first practical low-cost VR headset, at least for short sessions. And it was promising enough to get Oculus VR off the ground as a company. Oculus initially made DK1 units available to its Kickstarter backers in the summer of 2012, then sold them online to the general public in late 2012. Then,

infused with a large round of venture capital, the company went straight to work on the next generation, the DK2.

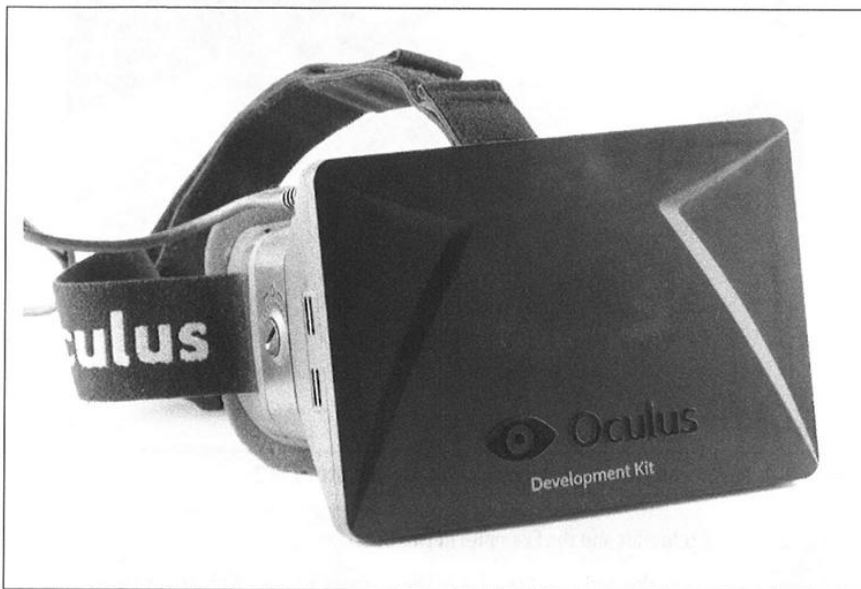


Figure 2-1. The original Oculus development kit, or DK1

The DK2

In the spring of 2014, Oculus released the second major incarnation of its headset. In contrast with the unwieldy DK1, the Oculus Rift DK2 is a sleek device. It is still quite large, covering a good portion of the user's face, but it is much lighter and features a more pleasing form factor than its predecessor. It also has only one cable coming out of the HMD, which then splits into the HDMI and USB cables that connect to the computer's video and USB ports, respectively.

The DK2 improved upon its predecessor with a lower-persistence, 1920x1080 display (960x1080 per eye). Beyond the display improvements, the DK2 incorporated a major advancement: positional head tracking. This allows the user to not only look around, but move around within a virtual scene—forward, back, side to side, and up and down. However, use of the positional tracker requires being in front of a tracking camera mounted on the computer, which effectively means the user needs to stay within a small zone near the computer. The positional tracking also only works when the user is facing the camera.

The DK2 was depicted in Figure 1-2 (see Chapter 1). The tracking camera for the DK2 is pictured in Figure 2-2.

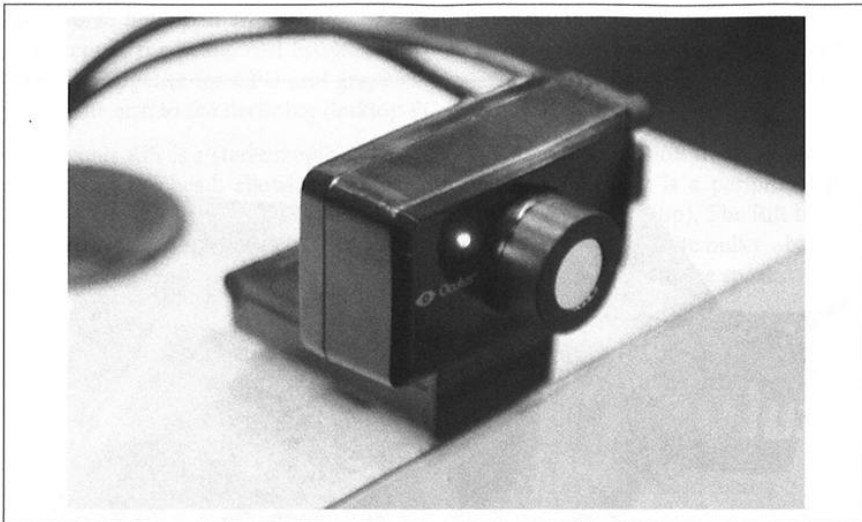


Figure 2-2. Positional tracking camera for the Oculus Rift DK2



Oculus Rift and the Examples in This Book

As of this writing, the CV1 is the primary Oculus Rift device being used in development. The examples throughout the book have been developed for the CV1, using the associated software development kit (SDK).

The CV1

The third major Oculus Rift prototype was code-named *Crescent Bay* and became the first consumer version. Oculus Rift CV1 (known as “CV1” for Consumer Version 1) features greater capabilities and big improvements in design. It has a higher resolution display and 360-degree head tracking, meaning that you don’t have to be looking at the tracking camera; there are position sensors mounted at the rear of the device as well. Oculus Rift CV1 is depicted in Figure 2-3.

Great content has already been created for the consumer version, including *Lucky’s tale*, the game shown in Figure 2-4. *Lucky’s Tale* is a 3D third-person platformer experience that was created exclusively for Oculus by Texas-based studio Playful Corp.

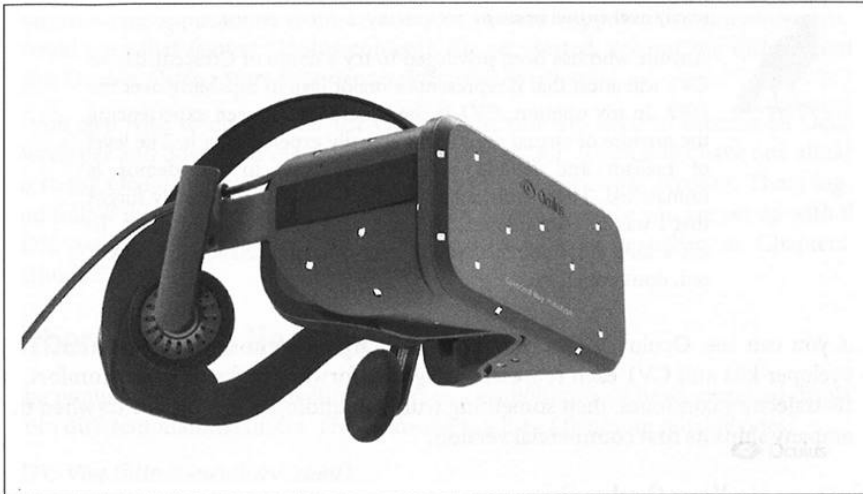


Figure 2-3. The Oculus Rift CV1, the newest Oculus headset

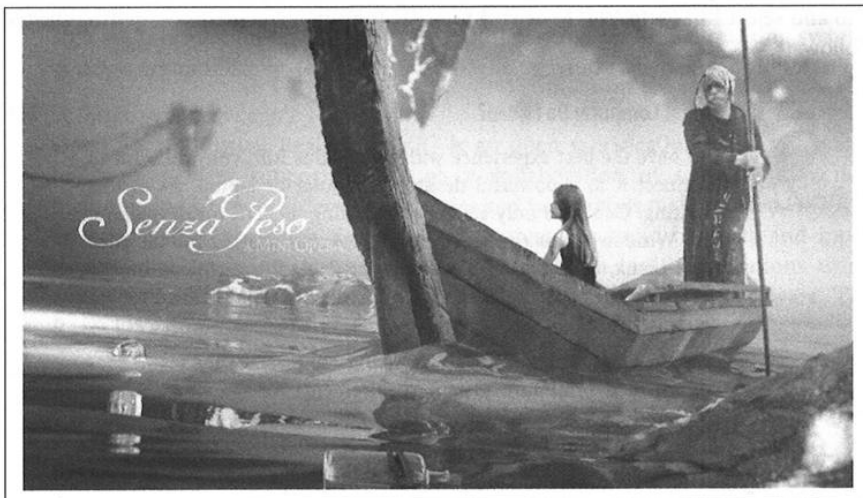


Figure 2-4. Lucky's Tale, a VR platformer game by Playful Corp.



Really Real Virtual Reality

Anyone who has been privileged to try a demo of Crescent Bay or CV1 will attest that it represents a major leap in capability over the DK2. In my opinion, CV1 is the difference between experiencing the *promise* of virtual reality and actually experiencing it. The level of realism and feeling of presence while in the demos is unmatched. There were many moments when I completely forgot that I was in a simulation... which I believe is the entire point. To cut a long story short: if you're ever offered a chance to check it out, don't pass it up.

As you can see, Oculus VR is continually striving to improve on its creation. The developer kits and CV1 each represent a big step forward in features and comfort. If this trajectory continues, then something truly astounding is in store for us when the company ships its first commercial version.

Setting Up Your Oculus Rift

If you are ready to take the plunge and buy a Oculus Rift, go to the Oculus Products tab and select Rift (<http://bit.ly/oc-rift>) to order it online. A few tips on getting started follow.



What Computer Do I Need?

To have the best experience with the Oculus Rift, you will want to connect it to a powerful desktop computer or laptop. As of this writing, Oculus is only actively supporting its SDK for developing on Windows. You can still find applications written for the Mac and Linux, and early SDK versions available for download on the Oculus developer site if you hunt for them. But for now, PCs are your best bet.

See the Oculus blog (<http://bit.ly/oculus-rec-spec>) for the team's recommended minimum system requirements for machines to run VR on the CV1 and newer models.

Downloading the Oculus runtime, examples, and SDK

Setting up the hardware and software can be tricky and requires a modicum of technical knowledge and a lot of patience. In order to experience content using the Rift, you must at a minimum download the Oculus setup for your platform. The setup provides the runtime software necessary for the Rift to communicate with your computer's operating system.

Oculus setup installers can be found on the Rift Setup site (<http://bit.ly/rift-setup>). Once you have the runtime installed, you can try out the Oculus Rift by downloading

and installing applications from a variety of sources (Appendix A contains a list of several sites that feature Oculus content). To get started, try out the official portal from Oculus, *Oculus Store Experiences* (<http://bit.ly/rift-ex>).

If you also wish to develop for the Oculus Rift, you will need to become an Oculus Developer and download the software development kit. If you don't have one already, go to the Oculus developer site to sign up for a free developer account. Then, log in and follow the instructions for your particular platform. Once you are set up with the SDK, you can develop desktop Oculus applications as described in Chapters 3 through 5.

Other High-End Head-Mounted Displays

The popularity of the Oculus Rift has led to a proliferation of head-mounted displays from different manufacturers. The following is a list of the more notable ones:

HTC Vive (<http://www.htcvr.com/>)

The result of a partnership between hardware manufacturer HTC and gaming titan Valve (the makers of the *Steam* downloadable game service), the Vive represents another major step forward in virtual reality systems. It features a comfortable, high-resolution HMD, a 15x15-foot room-sized positional tracking system, and two ergonomically designed motion-sensing hand controllers for input.

OSVR (<http://www.osvr.org/>)

Open Source Virtual Reality (OSVR) is an open specification or software and hardware devices supported through a collaboration of several companies, including long-time VR developer Sensics and peripherals maker Razer. OSVR is an ecosystem with which device manufacturers can build headsets and input devices that work interoperably, and developers can build applications using common APIs, without knowing the specifics of the underlying hardware. The Razer OSVR headset is depicted in Figure 2-5.

FOVE (<http://www.getfove.com/>)

FOVE is the next generation of HMD, able to track not only head movements but *eye movements*, opening up whole new possibilities for VR applications.

PlayStation VR (<http://bit.ly/proj-morph>)

Sony has created a virtual reality system for use with its PlayStation 4 console. The PlayStation VR features a comfortable HMD and two motion-sensing hand controllers for input.

Unfortunately for us as developers, none of these systems are compatible. Each requires custom programming using a different SDK. OSVR is attempting to provide an open, vendor-neutral way to talk to VR hardware, and this is promising; how-

ever, the software is too early in its adoption cycle to know whether it will become a standard API for programming VR.



Figure 2-5. Razer's headset, based on the OSVR open specification

That's how it goes on the bleeding edge of technology. While we may someday see a universally adopted software stack for creating VR, for the foreseeable future we will have to choose among several conflicting options.

Samsung Gear VR: Deluxe, Portable Virtual Reality

The Oculus Rift may be the flagship VR headset, but it is not without its issues. First, it requires a very powerful computer with a fast graphics processor. If you try to run Rift applications with a garden-variety PC laptop, MacBook Air, or older desktop computer, you won't have much fun. The frame rate will be slow, which is not only dissatisfying but can actually produce nausea. Second, the Rift is bulky and connected to the computer with a wire. You will need to install the position tracking camera that is shipped with the Rift and sit yourself down right in front of it for the tracking to work. Finally, it is not very mobile. Yes, you can pack up a six-pound laptop and your Rift to take to a demo party, but it's not the kind of thing you will want to bring to Starbucks (believe me, I've seen it done and it's not pretty), or on your train commute. This all makes for a cumbersome, stationary experience—and a potentially expensive one if you also need to buy a new computer.

Anticipating these issues, Oculus has also produced a much more lightweight mobile solution. Through a partnership with Samsung, Oculus technology has been incorporated into the *Gear VR*, a revolutionary headset that combines Oculus optics (those barrel distortion lenses) with new head-tracking technology, placed in a custom headset that houses a mobile phone with a high-resolution display.

Samsung released the Gear VR as an “Innovator Edition” in late 2014 and consumer versions in late 2015. The Gear VR unit costs US\$99 and can be purchased through Samsung’s online store. It currently only works with the Samsung Galaxy Note 4 (innovator), Note 5, S6, and S7 phones, so if you don’t already own one of these phones, this can be a pricey option. But it is very high quality, and may be the killer mobile option for the next few years. The new Gear VR is depicted in Figure 2-6.



Figure 2-6. The Samsung Gear VR

The Gear VR’s graphics are stunning, thanks to the Samsung phones’ display resolution of 2560x1440 pixels (1280x1440 per eye). The headset contains a custom IMU based on Oculus technology. The Gear VR IMU tracks much faster and with less latency (delay) than the IMU built into the phone itself.

The Gear VR housing has several intuitive input controls, including an eye adjustment diopter wheel (for matching the device’s interpupillary distance to your own), as well as controls that let you access functions of the phone within: a volume control, a headphone jack, and, most importantly, a trackpad for touch input.

The Gear VR is a fantastic consumer VR headset. A quick trip through the available demos and apps might even convince you that it's the best one on the market today. If money were no object, this could be it—however, including a newly purchased phone, the price may be prohibitive for some. Google has also recently announced a new mobile VR platform called Daydream which promises to certify upcoming phones and software with features similar to Gear VR.

The Oculus Mobile SDK

Developing applications for the Gear VR's Android-based operating system requires a different Oculus SDK, called the Mobile SDK, which is also available on the Oculus developer website (<https://developer.oculus.com>).

We will cover the basics of Gear VR development in Chapter 4.

Google Cardboard: Low-Cost VR for Smartphones

If they were automobiles, the Oculus Rift and Gear VR would be the Tesla and Lamborghini of VR headsets: best-of-breed but out of reach for the average consumer. The rough user experience, aggressive machine specs, and price tag in the hundreds of dollars mean that using a Rift is a major commitment. The Gear VR provides a sublime experience, but for now it is priced out of reach for anyone but a serious enthusiast or early adopter. Over time, the hope is that these higher-end consumer VR systems will become more affordable and accessible, and truly mass-consumer.

In the meantime, there is Google's *Cardboard VR*: a simple, low-cost way of adapting your existing smartphone to become a VR device. In 2014 Google introduced Cardboard VR, which allows users to experience virtual reality using pretty much any mobile phone, without the need for expensive additional hardware.

Google's original Cardboard VR unit, which debuted at the I/O conference in May 2014, is pictured in Figure 2-7.

To experience Cardboard VR, simply launch a Cardboard-ready application and place your mobile phone into the box. You will be immersed in a VR experience: a stereo-rendered 3D scene that you can look around and move around in by turning your head. Not bad for two bucks!

Cardboard is actually a reference specification. Google does offer it as a product; and you can get the specifications (<http://bit.ly/manufacture-cardboard>) from them and build one of your own.

If you don't have the time or the inclination to build a Cardboard viewer from scratch, you can also purchase a ready-to-assemble kit from one of several manufacturers, including DODOcase (<http://www.dodocase.com/>), I Am Cardboard (<http://www.imcardboard.com/>), Knox Labs (<http://www.knoxlabs.com/>), and Unofficial

Cardboard (<https://www.unofficialcardboard.com/>). Google also has approved Google Cardboard VR viewers on its Get Cardboard (<http://bit.ly/gvr-cardboard>) site. In addition to selling kits, each of these manufacturers also provides a mobile app, available to download from the Google Play Store and/or the iTunes Store, that provides a handy list of Cardboard-aware VR applications.

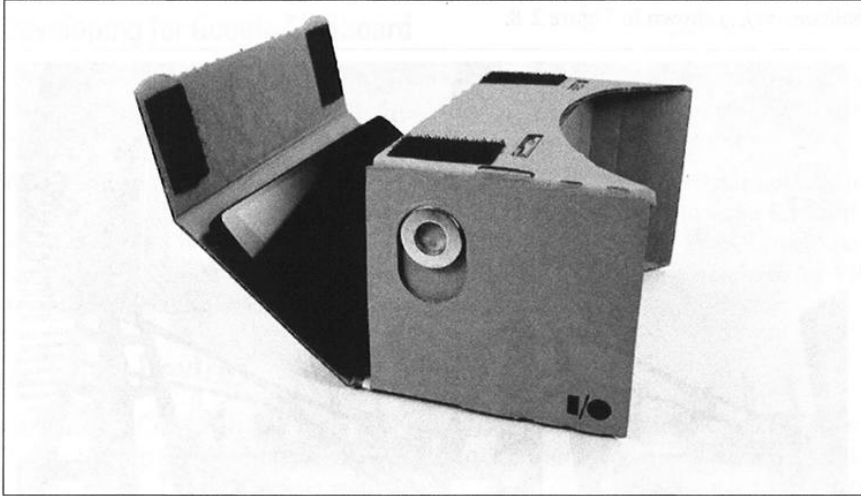


Figure 2-7. The original Google Cardboard VR viewer, shown at Google I/O in 2014

According to Google, over one million Cardboard viewers had shipped as of the first quarter of 2015. This number dwarfs the installed bases of the Oculus Rift and Gear VR, and on that basis alone, Cardboard is a force to be reckoned with. There are already hundreds of applications available for Cardboard, including games, 360-degree video and photo viewers, and educational simulations. Cardboard has powered VR experiences with big-name entertainers like Sir Paul McCartney and Jack White, and brands such as Volvo. So while insiders will tell you that the Cardboard VR experience isn't as deep or immersive as that of the Oculus Rift, for many people, it's going to be their first VR experience.



But Does It Have to Be Made of Cardboard?

No! Cardboard simply refers to the technique of dropping a smartphone into a box with lenses and implementing the appropriate rendering and head tracking in software. There are durable drop-in VR viewers made of other materials, including plastic and foam rubber. Chapter 6 describes several of these in detail.

Stereo Rendering and Head Tracking with Cardboard VR

The Cardboard VR approach to stereo rendering is similar to the Rift's—it is an distorted, 90-degree horizontal field of view. This allows applications to do simple side-by-side rendering in two viewports, one for each eye. An example of a 3D scene rendered side-by-side for Cardboard, from the app *Balloons VR* (<http://bit.ly/balloons-vr>), is shown in Figure 2-8.



Figure 2-8. *Balloons VR* for Google Cardboard

Head tracking for Cardboard is also straightforward. It simply uses the existing operating system orientation events generated by the phone's compass and accelerometer.

Cardboard VR Input

For Cardboard VR, input is still a work in progress. Because the phone is encased in the box, you can't touch the screen to tap or swipe. To get around this, the original Google unit featured a magnet mounted on the outside of the box that will trigger compass events that can be detected by an application. However, that combination works only with certain phone models.

Other makers of Cardboard units are developing innovations that will enable more universal input and work with most phones. For example, DODOcase, a San Francisco-based company that manufactures tablet and phone cases, is now making a Cardboard unit that uses a wooden lever to trigger a piece of plastic that touches the

surface of the phone, emulating the touch of a finger. With this mechanism, developers can use standard touch events in their VR applications. This method has been adopted for the official 2.0 version of Cardboard. Several Cardboard makers are also experimenting with creating peripherals, such as Bluetooth-connected devices that provide simple click or swipe input.

Developing for Google Cardboard

Google provides SDKs for Cardboard developers in two forms: an Android native SDK, and an add-in for the popular Unity3D game development system. Those can be downloaded from the Android developer site (<http://bit.ly/cardboard-dev>).

We will cover the details of building a Cardboard app using the Android SDK in Chapter 6. Note that you don't need to be an Android developer to write for Cardboard. Because most Android operating systems now support WebGL, you can actually write mobile web VR applications too! This is an exciting new area for VR development that we will cover in Chapter 5.

VR Input Devices

As we saw in Chapter 1, user input poses a new set of challenges in VR system design. With a head-mounted display completely enclosing your eyes, you can't see the physical environment around you—including your mouse or keyboard. Anyone who has tried a demo of the Oculus Rift knows all too well the feeling of your friend guiding your hands to the WASD or arrow keys so that you can use them to navigate your body through a virtual scene.

Most of the time, blind operation of the computer keyboard and mouse does not offer a good experience from a human factors point of view. With visual stimulation from the outside world cut off, we need to employ different types of input devices beyond the keyboard and mouse. A lot of experimentation is happening in this area, with the following types of input devices being used to provide a greater feeling of immersion:

- Game controllers, such as the controllers for the Microsoft Xbox One and Sony PS4 consoles. These can also be connected to desktop computers to interact with PC-based or mobile virtual reality scenes.
- Hand-tracking motion input sensors. Over the last few years, low-cost motion input devices have become available, including the Leap Motion controller (shown in Figure 2-9, mounted on the front of a DK2). The Leap Motion uses a combination of cameras and infrared LEDs to track hand motions and recognize gestures, similar to the Xbox Kinect.

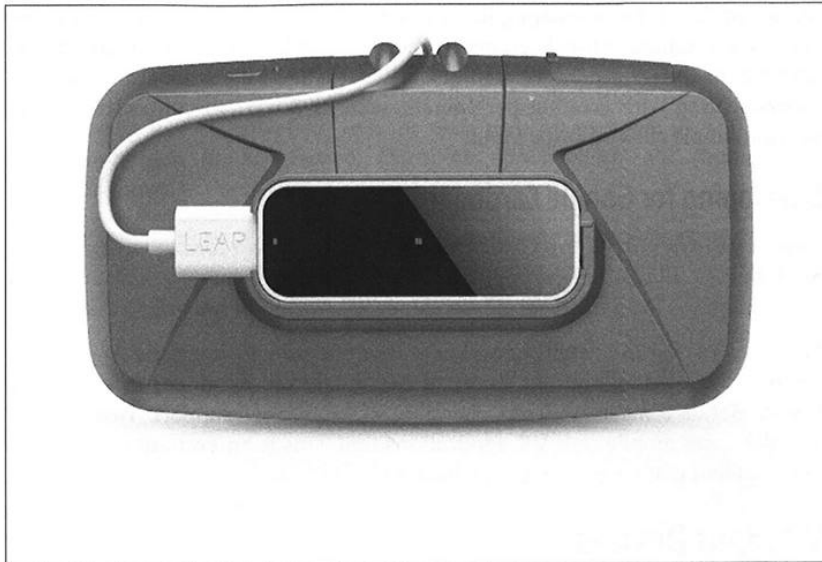


Figure 2-9. Leap Motion controller, front-mounted on the Oculus Rift DK2

- Wireless hand and body trackers, such as the full body motion STEM system by Sixense and the Hydra by Razer. These devices combine wand-like hand motion sensing with control buttons similar to those found in game controllers. Oculus is about to introduce its own such device into the market, called the Oculus Touch: a pair of lightweight wireless controllers that you hold in your hands. The Rift's positional tracking camera can locate the controllers' positions in space, so that you can see a representation of your hands in the VR world (if the app developer builds in that feature). The Oculus Touch controllers are depicted in Figure 2-10.

At this point in time, there is no one way to interact in VR using these new input devices. Much depends on the type of application being created, and the system to which the devices are connected. While it is too early to say what the “mouse of virtual reality” is going to be, it will likely emerge from products in the set described here, or devices like them. In the meantime, all this research is a big part of the fun. Expect to see a lot of innovation in this area over the next few years.

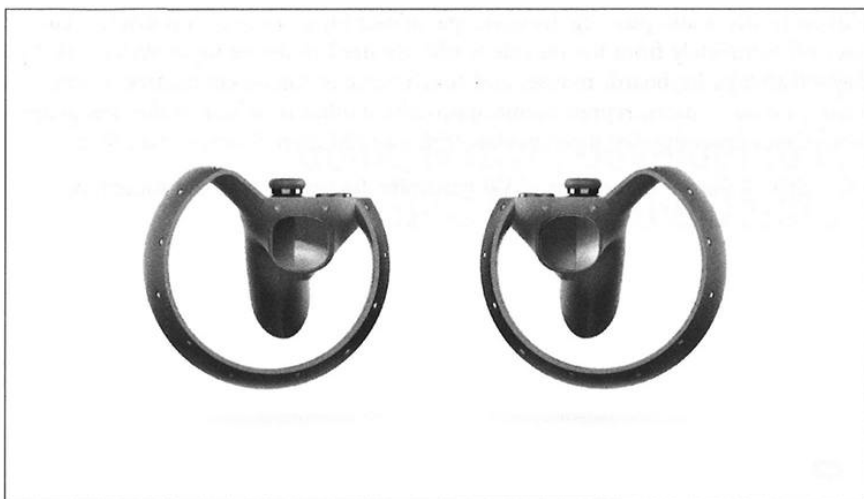


Figure 2-10. Oculus Touch controllers

Chapter Summary

Consumer VR hardware is a young and evolving field. The Oculus Rift leads this evolution with breakthrough display and head-tracking technology. Each version of the Rift gets better, more ergonomic, and more consumer-friendly.

While the Oculus Rift is the standard-bearer for virtual reality, it's not the only game in town. Other manufacturers are making high-end VR headsets for desktop computers and game consoles, including the HTC Vive, PlayStation VR for PlayStation 4, and HMDs based on OSVR, an open specification that allows manufacturers to build their own VR device.

For mobile virtual reality, Samsung's Gear VR, based on Oculus technology, is an outstanding product. Its high-resolution display, lightweight headset with intuitive controls, and Oculus Store user interface arguably provide the highest-quality VR experience to date. However, at a total cost of ownership that may approach US \$1,500, it is currently out of reach for most consumers.

On the other end of the spectrum, Cardboard VR turns your existing smartphone into a VR device using just a few dollars' worth of parts: a cardboard box and two plastic lenses. While it may not rival the high-end experience of a Gear VR or the ultimate Oculus Rift retail release, Cardboard represents a low-cost alternative that could provide a first taste of VR for many.

Virtual reality is also pushing the envelope on user input. Because VR displays cut the user off completely from the outside world, we need to devise input systems that go beyond today's keyboard, mouse, and touch screens. Game controllers, familiar to many computer users, represent one approach; another is to look at the new generation of motion-controlled input devices from Leap Motion, Sixense, and others.

Well, that's it for our nickel tour of VR hardware. It's time to go build something.

APPENDIX A

Resources

This appendix contains lists of resources for learning more about virtual reality systems and programming tools, and VR in general.

Headsets, Input Devices, and Video Capture Systems

Headsets

High-end head-mounted displays for desktops and consoles

Several manufacturers are creating HMDs that work with desktop computers and game consoles. These include:

- The Oculus Rift (<https://www.oculus.com>)
- The HTC Vive (<http://www.htcvr.com>)
- OSVR HMDs based on the OSVR open specification (<http://sensics.com>, <http://bit.ly/osvr-hacker-dev>)
- Sony's PlayStation VR (<https://www.playstation.com/en-us/explore/playstation-vr/>)
- The FOVE eye-tracking HMD (<http://www.getfove.com>)

Mobile HMDs

Mobile virtual reality allows you to use the smartphone already in your pocket to experience VR, by simply dropping it into a stereoscopic viewer. Here are some of the options currently on the market:

- The Gear VR for Samsung smartphones (<http://www.samsung.com/us/explore/gear-vr/?cid=van-mb-cph-0716-10000089>)

- Google Cardboard (list of approved headset manufacturers: <http://bit.ly/get-cardboard>)
- The Durovis Dive (<http://www.durovis.com>)
- MergeVR's Merge Goggles (<http://mergevr.com>)
- The Wearality Sky (<http://wearality.com>)

Input Devices

Input poses new challenges to design and human factors. Since the user is visually cut off from the outside world, the mouse and keyboard don't work well as input devices. Many VR applications are being designed to work with existing popular game controllers, such as those for the Xbox 360, Xbox One, or PS4, but new types of input devices and systems are also being explored. Here are some of the more promising offerings:

- Leap Motion (<https://www.leapmotion.com>)
- The Sixense STEM System (<http://sixense.com>)
- The Oculus Touch (<https://www.oculus.com>)
- The Vive hand controller (<http://www.htcvt.com>)
- The MergeVR controller (<http://mergevr.com>)

360 Video

Not all VR is created using software engines. A whole class of virtual reality experience is being developed based on 360-degree video—panoramic movies in which you are immersed and can look around without restriction. The 360 videos can be based on computer graphics animation, or they can be captured from real life using special cameras. Several companies are developing combination hardware and software systems to capture and stitch mono and stereo 360 video. If you're interested in video applications, check out the following:

- Jaunt (<http://bit.ly/jaunt-vr>) creates camera systems, stitching software, and original film productions.
- Google is developing Jump (<http://bit.ly/cardboard-jump>), a capture setup for creating 360 videos for Cardboard. The Jump rig holds 16 GoPro HERO4 cameras arranged in a circular array.
- 360Heros (<http://www.360heros.com>) creates camera rigs, production software, and a 360 player, and offers production services to makers of 360 videos.
- On the low end, the RICOH THETA (<https://theta360.com/en>) is a small, handheld consumer-grade camera for capturing still and live motion 360 video.
- Kolor (<http://www.kolor.com>) develops 360 video stitching and playback software.

Applications and Content

Flagship Experiences

The following list represents just a handful of the amazing VR experiences and applications being developed:

- *EVE: Valkyrie* (<https://www.evevalkyrie.com>) is a VR multiplayer dogfighting shooter game set in the popular EVE universe: you command a heavily armed fighter in “the most realistic dogfighting game available on any platform.”
- *Elite: Dangerous* (<https://www.elitedangerous.com>) is a VR space adventure, trading, and combat simulator game; this is the fourth release in the highly successful *Elite* video game series.
- *TheBlu: Encounter* (<http://wevr.com>), a breathtaking underwater environment featuring a sunken ship and a giant, unbelievably realistic blue whale, was created by VR production and technology company WEVR as one of the first showcases for the HTC Vive. (Disclosure: WEVR is my current employer.)
- *Tilt Brush* (<http://www.tiltbrush.com/#intro>) for Vive is a new version of the innovative virtual reality painting and sculpting application created by San Francisco-based development team Skillman & Hackett, now working at Google. Originally designed for the Oculus Rift, the Vive version is even more amazing, making great use of the hand controllers to provide a transcendent content creation experience.
- *Henry* (<http://bit.ly/oculus-henry>) is a touching animated piece about a lonely but loveable hedgehog, created by Oculus Story Studio, the company’s in-house production team. The Story Studio is dedicated to moving VR forward as a new interactive storytelling medium.
- *The Fifth Sleep* (<http://bit.ly/the-fifth-sleep>) for Gear VR is a voyage through the human body and mind. Creators InnerspaceVR tell stories using animated 360-degree video developed using state-of-the-art production techniques and video game tools such as Unreal Engine and CRYENGINE.
- *Senza Peso* (<http://bit.ly/senza-peso>) is one of the original breakout cinematic VR experiences, developed for the Oculus Rift in 2014 by LA-based studio Kite and Lightning.

Noteworthy Companies

Here are a few other companies of note, not mentioned elsewhere in this appendix:

- *Within* (<http://with.in/>) is a production company telling stories in VR, founded by noted experimental filmmaker Chris Milk.

- Vantage.tv (<http://vantage.tv>) captures live events such as concerts and music festivals. You can experience the events online using their player and even share in the fun in real time with up to four friends.
- NextVR (<http://www.nextvr.com>) delivers live and on-demand virtual reality experiences in broadcast quality.

Download Sites

Many great VR apps and experiences are available for download, hosted by Oculus, Valve, and others. Here are a few sites to get you started:

- Oculus Share (<https://share.oculus.com>)
- SteamVR (<http://bit.ly/streamvr>)

WebVR Showcases and Applications

I am firm believer that the Web holds one of the keys to virtual reality's future. While the technology of WebVR is still on the bleeding edge, and trailing behind on performance (ever so slightly), development proceeds apace and there are already some really cool showcase experiences. For example:

- Mozilla's WebVR showcase site, MozVR (<http://mozvr.com>), offers demos, downloads, and a blog providing tips for developers and reporting on current projects.
- Chrome Experiments (<https://vr.chromeexperiments.com>) for VR is a site that features demos for Google Cardboard written completely with mobile browser technology.
- Vizor (<http://vizor.io>) is a browser-based content creation and cloud hosting platform for WebVR.
- WebVR 360 video players are being developed by EleVR (<http://elevr.com>) and LittleStar (<http://littlstar.com>).

SDKs, Development Tools, and Programming Languages

Device SDKs

The VR development stack runs the gamut of software SDKs, tools, programming languages, and frameworks. Depending on what device you're targeting, here are some of the SDKs you're likely to need:

- Oculus (<http://developer.oculus.com>) provides SDKs for developing for the Rift on the desktop and for the Gear VR for Samsung smartphones.

- OpenVR (<http://bit.ly/open-vr>) is a C++ API and runtime developed by Valve that allows access to VR hardware from multiple vendors, without requiring that applications have specific knowledge of the hardware they are targeting. It is currently being used to develop applications for the HTC Vive. It also comes with Unity and Unreal plugin packages.
- OSVR (<https://github.com/OSVR>), not to be confused with OpenVR, is an open source software platform for virtual and augmented reality applications. It allows discovery, configuration, and operation of a range of VR devices on a variety of platforms and supports many types of devices, from HMDs to trackers and more. OSVR is being developed by a close consortium of companies including VR developers Sensics and peripheral maker Razer.
- Google (<http://bit.ly/cardboard-dev>) provides both native Android (Java) and Unity SDKs for Cardboard.

Game Engine Middleware

Today, most virtual reality is developed using game engines and associated editing and pipeline tools, known collectively as *middleware*. Here are some of the most popular options:

- Unity3D (<http://unity3d.com>) is the indie game development tool of choice, featured heavily in this book.
- Unreal Engine (<http://bit.ly/unreal-engine-4>), created by powerhouse game developer Epic Games, is feature rich and growing in popularity for VR development.
- CRYENGINE (<http://cryengine.com>) is another powerful engine created by successful developer Crytek.

Programming Languages

The book covered several development languages. If you're unfamiliar with one or more of these languages, start with the Wikipedia entries for a history and then search online using your favorite search engine:

- Originally created by Microsoft for its .NET framework, C# (<http://bit.ly/wiki-csharp>) was chosen by Unity as its preferred language for developing scripts. An investment in learning C# will get you going with Unity, which targets the most platforms and VR systems of any game development tool today.
- An old horse in early web and enterprise server development, Java (<http://bit.ly/wiki-java>) got new life when it became the main programming language for the Android mobile operating system. The Google Cardboard native SDK and the GearVRf framework from Samsung (see the next section) are written in Java.
- JavaScript (<http://bit.ly/wiki-js>) is the scripting language for web browsers. It has grown from humble beginnings as a “toy” scripting language into a powerful lan-

guage used by millions of developers, from hobbyists to professional engineers. It is arguably the most widely used programming language on the planet. JavaScript is the language you want to use if you are looking at developing web-based VR applications, either for the newer WebVR API or for Cardboard-style rendering and head tracking.

Software Frameworks

In addition to the game engine middleware listed earlier, there are other software frameworks available for programming VR. These frameworks are code only, not packaged with an editor like Unity or Unreal. Some examples are:

- GearVRf (<http://gearvrf.org>) is a native Java framework for creating VR applications, developed by Samsung. GearVRf is fully open source with no license restrictions.
- WebVR is an open effort that includes specifications, software implementations (browsers), a mailing list for discussion, and JavaScript frameworks for coding. The WebVR specifications can be found at <http://bit.ly/webvr-spec>. The WebVR mailing list is at web-vr-discuss@mozilla.org.
- Firefox Nightly builds for desktop and Android are available at <https://nightly.mozilla.org>.
- Chromium development builds and mobile betas can be found at <http://bit.ly/chromium-webvr-builds> and <http://bit.ly/chromium-webview>.
- WebVR programming frameworks use WebGL for in-browser rendering and the WebVR APIs for head tracking and fullscreen stereo rendering. They include:
 - Three.js (<http://threejs.org>)
 - Babylon.js (<http://www.babylonjs.com>)
 - SceneJS (<http://scenejs.org>)
 - GLAM (<http://glamjs.org>)
 - A-Frame (<https://aframe.io/>)
- WebVR 360 video player engines include open source projects from eleVR (<http://bit.ly/elebr-web-player>) and Littlestar (<https://github.com/littlestar>).

Websites

There are a growing number of news sites, blogs, and podcasts devoted to virtual reality. Here is a selection of some of the best:

- Upload VR (<http://uploadvr.com>)
- Road to VR (<http://www.roadtovr.com>)

- Voices of VR (<http://voicesofvr.com>)
- Enter VR (<http://entervr.net>)

Meetup Groups

Meetup groups are a vital resource for learning about VR and networking with like-minded developers, artists, and thinkers. The San Francisco Bay Area alone plays host to more than a dozen VR-related meetup groups, including:

- Silicon Valley Virtual Reality (<http://bit.ly/sv-vr-meetup>), the original VR meetup group, always exploring the latest in virtual reality development.
- San Francisco Virtual Reality (<http://www.meetup.com/virtualreality>), a booming meetup group with over 1,000 members, featuring great demos and speakers.
- San Francisco WebVR (<http://www.meetup.com/Web-VR>), a group devoted to WebVR development (I'm the co-organizer of this group).
- Kaleidoscope SF (<http://www.meetup.com/kaleido-SF>), a group dedicated to virtual reality filmmaking and storytelling (formerly VR Cinema).