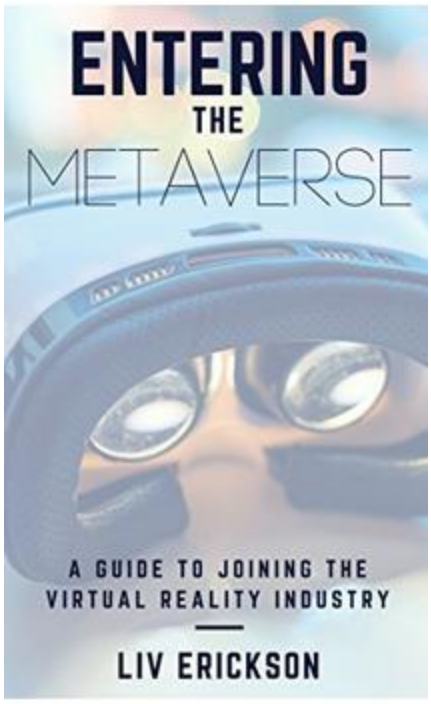

Entering the Metaverse

A GUIDE TO JOINING THE VIRTUAL REALITY INDUSTRY

Liv Erickson | HTTP://LIVIERICKSON.COM |Originally Published September 5, 2016

Contents

Overview

So, you've heard the hype about the virtual reality industry, and you've got this feeling you aren't quite sure about. It's the understanding that something big is happening, and you want to be a part of it. That 'lightbulb moment' has happened, where you're drawn to this new, exciting technology, and you aren't entirely sure what to do to get started, but it's there.

Congratulations on taking that first step.

Virtual reality as a technology has been the topic of quite a few science fiction stories throughout the past decades, but it's only within the past few years that this technology has been accessible to the average person to pick up and start using. In the 90's, virtual reality devices cost tens of thousands of dollars, and were generally only found in high-tech companies or universities with a lot of cash. Luckily, that is no longer true – devices like Google's Cardboard headset and the Oculus Rift have started to make their way into homes across the world, and people everywhere are getting more and more excited and involved with the potential of immersive technologies.

Since you're reading this, I'm assuming that you're one of those people.

Immersive technologies have the potential to fundamentally change everything about how we interact with computers and data. As human beings, we are spatial creatures who think and interact in three physical dimensions, but for the past several decades, all of those interactions have happened on variations of a 2D screen. Virtual and augmented reality technology has the ability to thoroughly change the relationship between humans and computers to make digital information more powerful and easy to understand than ever.

As we look to the opportunity presented by immersive technologies, we must also look at a problem facing the technical industry: a huge shortage of individuals working in STEAM (science, technology, engineering, art, and math) fields. There is still a huge gap in the understanding of who a developer is. She is the theatre professional building an augmented world around her with new art forms that bridge the physical world and a mobile phone. He is the educator working on building a science game to teach physics in an interactive way. Contrary to public portrayal of programming and technical roles being the domain of a 'geek' or a 'nerd', software creation is, put quite simply, being able to use computers and technology to solve a given problem.

The purpose of this short book is to help make your own path into the virtual reality seem more clear. Whether you are an artist, a writer, a systems administrator, a physician, software developer, singer, or student – you can create for virtual reality, and you can start today. Immersive technology is a whole world technology, and requires people from all backgrounds and experiences to build new applications that we have yet to imagine.

I'm here to demystify the Metaverse, and show you how.

Introduction

ABOUT THIS BOOK

This book is not meant to be an all-inclusive, zero-to-virtual-reality-expert guide. Truthfully, that's going to be a unique path for every individual, based on a lot of different factors that are hard to predict in a book. Instead, this guide is meant to be a broad introduction to today's virtual and augmented reality industry and help expose you to a number of options for creating immersive content. There is no one-size-fits-all solution to learning about virtual reality, and the opportunities for different specialties and roles across the industry are innumerable. My goal is that this book helps unmask some of those for you and helps you identify areas that you can dive deeper into, based on your own experiences and interests.

MY STORY

I had my own lightbulb moment two years ago. To give you the full story, though, it makes more sense to start at the very beginning.

I started coding back in 2006 as a sophomore in high school taking 'Computer Math', the prerequisite coding course for AP Computer Science. I had spent most of my childhood obsessed with computers, teaching myself Photoshop and web development (I ran a particularly embarrassing Star Wars fan blog and convinced several friends of mine to collaborate on a teen girl quiz site with me at the experienced age of 13), bargained with my parents for a laptop in exchange for good grades, and got my Nintendo 64 taken away from me on more than one occasion. By the time I got to high school, the idea of a programming class just made sense for me. I happened to participate in a pilot program that had me getting to school early to take English before the official start of the day, so I had room in my schedule for an extra class—and thus my complex love-hate relationship with programming began at the age of 15.

At the time, I didn't notice the patterns I see now—I loved the parts of the class where we would use Java's graphics libraries to draw scenes, and felt no greater sense of accomplishment (maybe even to this day) when I finished my first-ever game, a Star Wars themed 'brick break' application. I moved into AP Computer Science the following year, and while I did great on the exams, I met my first obstacle: a teacher who subscribed to the "CS isn't for girls" mentality. I finished the class knowing that I'd be a computer science major in college, but the seed of doubt was there.

In college, I jumped immediately into the CS courses and struggled through algorithms courses. I wasn't motivated to build another sorting program year after year, and although I look back on my formal language and systems-level classes fondly, during school, I found myself drawn to the design and management courses more than any of the coding ones. I declared myself a "CS major who hates programming" and filled my electives with theory over practical courses.

My senior year, I started to see the trends in the courses I loved. I had a wonderful professor who let me learn Windows 8 development as an independent study project, where I learned how to write tutorials and experiment with new platforms. I took an amazing class on Ethics in Gaming and Virtual Environments, which set the foundation for a lot of my interest in how VR and AR are both perceived and used by the general public, and how gender issues tie into the technology

industry. I started coding for fun again, but when I landed at Microsoft Silicon Valley as a program manager, I thought that I was happy to leave my developer days behind me.

Spoiler alert: that didn't happen. Maybe it's the nature of Silicon Valley, maybe it's just the fact that 7 years of development experience didn't want to go away, maybe it's that I finally saw The Matrix at the ripe old age of 22 — regardless of the reasons, I found myself envious of the passion that was surrounding me in the startup founders I met. The Thiel fellowship, the YC events — while I had a job that I really enjoyed, I was missing the fire that so many people I met seemed to have.

I started on my personal mission to find that fire, and dove into a number of new side projects: iOS and Android development, graphic design, web development — but none of them stuck. I was growing my skills technically, but still hadn't found anything that I felt inspired by. I couldn't see myself ever feeling the way it felt like everyone around me did.

One night, browsing through YouTube, I came across the PixelWhipt [i]YouTube channel and watched the first two episodes of VirtuAlly, a show about the VR industry and how it was growing rapidly. I watched in fascination as Ally talked about Oculus and interviewed people in the industry. I felt that burning sensation that I had been waiting for. The room seemed brighter — fueled by something bigger.

This was that "lightbulb moment" — the exact place in time where I realized that I had found what I wanted to spend my life working on.

I immediately went online and ordered a Google Cardboard, a device I had previously giggled off as something that was almost a joke. I found Silicon Valley Virtual Reality on Meetup.com[ii] and went to my first event. I tried the Oculus Developer Kit 2 for the first time and immediately ordered my own when I got home. I downloaded Unity[iii] and started teaching myself how to use it. At first, I made spinning cubes. As I learned the tools, I was also diving deeper into understanding more and more about the industry at large.

As I began teaching myself Unity, I realized that there was a lack of educational material around virtual and augmented reality development. I started writing down everything that I did with Unity and virtual reality development in a tutorial for a friend's developer resource site. They ended up closing down their site, but I wasn't discouraged, and published the walkthrough anyway. In all of my free time, I was pushing my laptop to its limit with VR demos and development environments. I wouldn't shut up about my love for VR, so I turned to the internet.

While at an SVVR[iv] meetup, I heard how Samsung's Developer Conference (SDC) had a virtual reality track, so I did what any normal person would do: took off 3 days of work to go to an event that I had no idea about. I attended all of the sessions on virtual reality and GearVR, and met other enthusiasts about VR. I loved everything about that week, and the momentum fueled me to keep writing about my experience.

Several weeks after SDC, the developer experience team at Microsoft offered me a position as the virtual and augmented reality developer evangelist in Northern California, and I officially joined the team in February of 2015. In the VR industry, I've found an amazing blend of life changing technology, wonderful, passionate people, and the power to work on something that I truly see impacting the world in an incredibly positive way. Today, I've taught countless classes on virtual

and augmented reality development, built a number of applications, advised quite a few virtual reality startups, and have been running a blog[v] on immersive technology for over two years.

When I first thought about joining the virtual and augmented reality industry, I was someone who felt incredibly lost. Even though it was overwhelming at times, and I felt like I was faking my way through my work, I didn't stop looking until I found something I knew would help fuel that drive I was looking for.

It isn't about where you've been or what your previous experience was – it's about how you can turn those into your next steps.

My background was not initially in gaming, and I haven't always loved programming, but when I realized that we were entering a new virtual reality renaissance, none of that mattered anymore.

My goal is to share with you what I've learned over the past several years and help demystify the virtual reality industry as it stands today. I don't pretend to be the end-all expert on the topic - those fall to many others who have laid the groundwork for 3D computing and immersive technology – but I hope that my knowledge and experience will benefit you, and help you begin to think about how you may look to find your own role in an exciting industry seeing a new resurgence in possibility.

Understanding the Virtual Reality Industry Today

VIRTUAL REALITY, A (BRIEF) HISTORY

Virtual reality is not a new concept. From science fiction stories to the Nintendo Virtual Boy, incarnations of immersive technologies have been present in our lives for decades. Even as far back as the 1960's, research labs were exploring ways to replace or augment our physical world with computer data.

In the 80's and 90's, the first taste of enterprise VR appeared in the form of computers costing upwards of $40,000, and consumer VR started to appear alongside Nintendo and Sega gaming consoles. Arcade pods for virtual reality cost up to $73,000, and a number of accessories sprung into the market to try and augment digital experiences.

Throughout the late 90's and early 2000's, virtual reality technology continued to develop out of the consumer eye, with the majority of research and advancements taking place in labs and experimental development companies. In 2011, Palmer Luckey created the Rift, a prototype VR device that launched a hugely successful Kickstarter project and resulted in his company, Oculus VR, being purchased by Facebook for $2 billion just three years later. Today's wave of consumer virtual reality devices is often credited, at least in part, to the success of Luckey and the Rift headset.

Over the past several years, a number of other electronics and technology companies have stepped forward with their own devices, software, and ecosystems to support a new generation of developers, creators, and consumers who are interested in wearable immersive technology. Google, Microsoft, Sony, HTC, Samsung, Intel, and Qualcomm have all shown off head-mounted hardware for virtual, augmented, and mixed reality applications, and while many of them are still in early development stages, the potential of 3D immersive computing is undeniably an upward trend in the technology industry.

VIRTUAL REALITY PLATFORMS

Today's VR devices come in a variety of forms, but the most common devices in today's ecosystem are head mounted displays (HMDs), devices that are worn on the head and contain a display that a user will wear in front of his or her eyes. The computational power behind these devices vary, ranging from a $15 Cardboard headset powered by your mobile phone to full room tracking with the $799 HTC Vive.

In this section, I detail some of the major platforms that are available for virtual reality content today, and the differences between choosing mobile, desktop, or other types of VR technologies.

While virtual reality has been around in a number of forms over the past several decades, the resurgence of the technology can be found powered by a new variety of different devices. Virtual reality devices today are generally *head mounted display* technologies, powered by software that displays applications rendered *stereoscopically*, or, in some cases, simply split side by side.

Mobile Virtual Reality

As the name suggests, mobile virtual reality is a platform for virtual reality content that is driven by a smartphone. These types of devices usually don't have their own display included, and are often powered entirely by the mobile phone when it is placed into the viewer. At a minimum, mobile VR devices contain lenses that distort the screen of the mobile phone, which runs applications rendered in a split-screen view side by side.

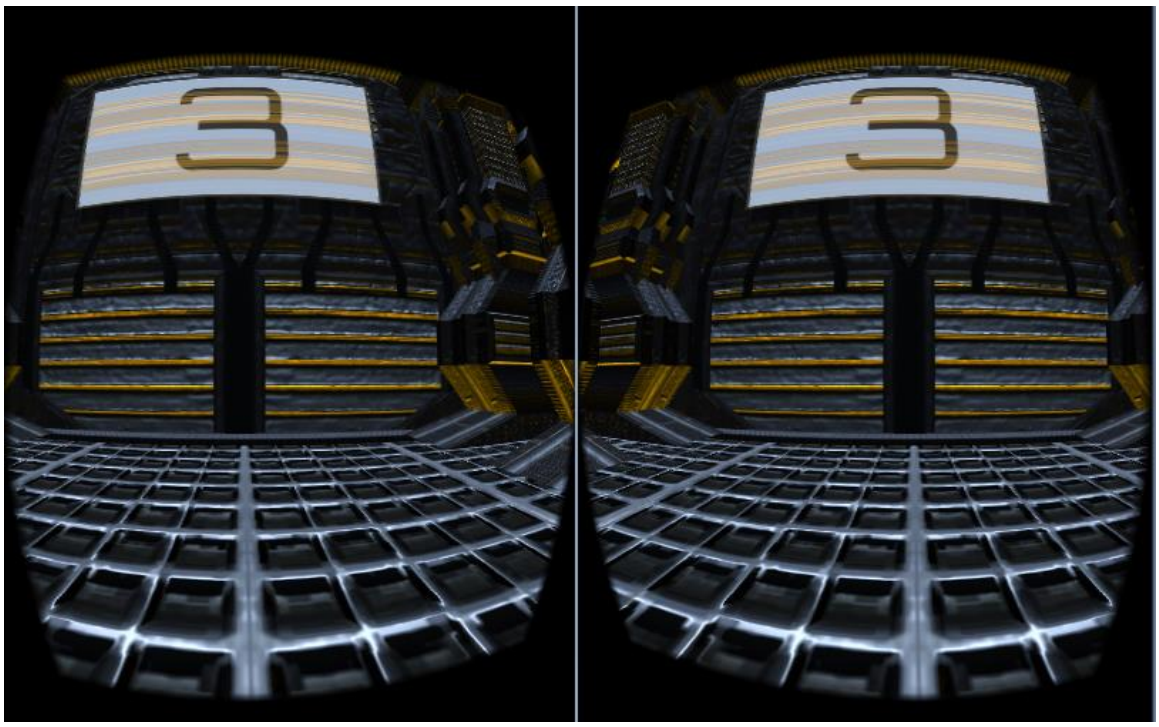

Figure 1: An image of a Cardboard VR mobile application. This application runs on a smartphone and is viewed through a set of lenses for comfortable viewing.

Cardboard VR

Cardboard VR was announced in 2014 at I/O, a developer conference hosted annualy by Google. What was initially regarded as a cheeky response to the growing interest in virtual reality quickly became one of the most prolific VR devices, and still remains the lowest cost option for VR viewers.

Cardboard viewers are exactly what you'd expect from the name – a cardboard housing with a slot for a smart phone and two basic lenses that distort the image on the screen to feel natural to look through. Applications are downloaded through the Play Store, and multiple apps support 360-degree video viewing in the device as well. A magnet on the side of the viewer serves as a trigger for sending an input to the app, and serves as a type of pointer/mouse click interaction.

There are a number of variations on the Google Cardboard hardware, including adjustable viewers to adapt more comfortably to individual users and devices made of more solid materials. Some of these devices are nothing more than the lenses and a clip to hold them to the front of the phone, designed for quick viewing, while others come with controller options, head straps, and in some cases, such as with the ViewMaster VR device, exclusive content.

Out of the different viewers based off of the Cardboard specs, several have stood out to me as being particularly interesting. The Homido VR viewer is one of the better variations on the Google Cardboard headset that I've tried, but it does have the downside of having an additional Bluetooth control instead of a button on the side to trigger different interactions. Wearality primarily works on lens technology, but they did have a pretty interesting proof of concept viewer showcasing wide field of view (FOV) – that seemed to be what kicked off the trend of ultra-small viewers made up of just the lenses themselves with a clip. One thing that I really love about Cardboard is how easy it is to customize – I also have two limited edition Cardboard viewers that were released when Star Wars: Episode VII came out.

Samsung Gear VR

At the high end of mobile virtual reality is the Samsung Gear VR headset – a specialized HMD that works exclusively with Samsung's flagship phones (Samsung S6 / S7, Note 5 families) and contains additional hardware to help offload some of the requirements from the phone onto the device itself. The Gear VR runs software that was developed in a partnership with Oculus, and provides an interface for navigating library content and purchasing new applications while in the virtual reality experience.

In addition to the lens components, the Gear VR has an adjustment wheel on the headset to help change the viewing distance and improve the overall quality of the experience. The Gear VR also has an inertial measurement unit (IMU) integrated into the headset to improve the rotational tracking when a user turns his or her head. The controls, which include a touch/gesture pad, a back button, and volume controls, are all added onto the headset itself to remove the need for an external controller.

Because Samsung is able to maintain more end-to-end control in the manufacturing process when compared to Cardboard viewers, applications are generally more polished and the device is often considered the highest performing on the mobile HMD market today.

In 2014 at Google I/O, Google announced their venture into the virtual reality industry with the Cardboard VR headset. While first seen as a bit of a toy, the number of low-cost mobile virtual reality headsets quickly exploded onto the market with a huge variety of custom headsets that range from $5 cardboard & lens kits to $120+ headsets that have adjustable lenses, custom controllers, and more – all powered by a mobile phone.

A few months after Google announced Cardboard, Samsung released the first version of their own VR headset, the Gear VR Innovator Edition, a headset designed to work with the Samsung Galaxy Note 4. Subsequent years showed the release of a headset for the S6 line of phones, and finally, in late 2015, Samsung released the consumer version of their Gear VR headset that was compatible with the full line of their flagship phones.

Now, mobile virtual reality has proven to be a key point in the growing virtual reality ecosystem, providing a cost-effect entry for both consumers and developers to begin creating virtual worlds. Developers can start designing for VR with applications that run on their smartphones, and consumers get a taste of 360 videos and immersive applications with something that can be made, quite literally, out of a pizza box.

Mobile VR headsets are growing in popularity, mainly due to the fact that the smartphone platform makes distribution of applications affordable and simple. Apps are sold in an app store the same as any non-VR application, and the cost to buy a viewer is generally quite low. There are some tradeoffs, both in interaction availability and in processing power, but mobile VR is proving to be a compelling introductory platform for users to immersive technologies.

Google Daydream

In 2016, Google shifted their VR efforts from the Cardboard moniker into the Google Daydream brand, a continuation of their earlier efforts on bringing virtual reality experiences to mobile devices. Daydream-ready devices are expected to ship later this year, and offer a number of improvements over the original Cardboard experiences.

Desktop Virtual Reality

Oculus Rift

Arguably The most well-known name on the VR market, the first version of the Oculus Rift kicked off the wave of modern-day virtual reality with over two million dollars pledged on the crowdfunding site Kickstarter to bring the first developer kit (DK 1) to life in 2012. The second device (DK 2) followed in 2014, and as of 2016, the Rift is a consumer-ready device shipping world wide – available for order, right now.

The Rift is a desktop virtual reality device produced by Oculus, and like the Gear VR contains Oculus Home software as a launching pad for applications. The Rift comes with built-in, removable headphones, and connects to a PC through HDMI and USB 3.0 cables.

In addition to the display itself, the Rift also has a standalone tracking component that uses an additional USB 3.0 port on the desktop. This handles positional tracking for the device. The Rift also ships with the Oculus Remote and an Xbox One wireless controller to interact with applications. You can currently order a Rift from Oculus for $599 USD.

HTC Vive

The HTC Vive is another choice for desktop virtual reality applications. Developed in partnership with Valve, the Vive comes with two hand tracked controllers and two ‘lighthouse’ base stations that provide tracking for room-scale VR. The Vive comes with a control box that houses an HDMI port and USB port, which is powered and connects through to the appropriate ports on the PC running the applications.

The Vive runs applications purchased through Steam VR and can run in full room scale mode or in standing-only mode, both of which require setting up the base stations. Unlike the Oculus, the Vive does not come with built-in headphones, but instead contains a cable opening at the base of the headstrap to plug in your own audio device.

Look Ma – Hands! My First Experience with the HTC Vive

September 11th, 2015

Being in the virtual reality industry makes work feel like play, 100% of the time. Yesterday was no exception – by way of my awesome friends over at Convrge, I finally got to go hands-on (literally!) with the HTC Vive, the VR headset under development from HTC and Valve.

Hands: The Missing Presence

I've been fortunate enough to try several different types of VR input devices over the past year, but the controllers with the Vive were an entirely new experience that helped improve presence immensely when paired with the Lighthouse tracking system. The caveat with the controllers was that they didn't always get picked up properly – it took almost an hour to get the system up and running before the tracking was working consistently – but when they did work, it was a mind-blowing experience.

With the controllers, it was hard to completely judge things like battery life, but I did notice that throughout the course of the evening, one of the controllers kept disappearing and kept needing to be restarted. A series of rechargeable batteries were on standby, and I think at least once (maybe twice?) they had to be replaced. Without being able to test a headset of my own (hey Valve… *wink wink*) it's hard to judge the overall use time on a single charge, but a quick search seems to indicate that this experience is fairly typical.

A Whole New (Series of) Worlds!

The first demo that I got a chance to play with was one where one controller was a bow and the other allowed you to notch and shoot an arrow at a series of targets. I had a blast lighting the virtual arrows on fire and attempting to shoot down balloons, and I was amazed how the feedback from the controllers made it feel like I was actually using a physical bow and arrow set. The feeling of accomplishment with completing tasks in VR with the Vive was equally as strong as it was for completing the equivalent task in the physical world – there's a lot of potential here for researching and evaluating our neurological reward system through VR environments.

The second demo was more passive, and involved floating underwater while a huge blue whale swam past amongst schools of smaller fish that could be batted away with a swipe of a controller-wielding hand. I don't think that anything up until that moment had truly captivated my understanding of scale in VR – it was immensely disconcerting in the best possible way to see a massive virtual creature like that just an arms length from where I was standing. When I took the headset off, I instinctively looked up again for the whale.

Demo #3 was one that I've been dying to try for ages: Tiltbrush! This application allows you to paint in 3D space around you – there's an excellent demo of the app used by a Disney Animator that showcases the technology. I immediately became enthralled with the different brushes and was hard-pressed to hand over the headset: I felt like I could stay in there forever. One of the most impressive things for me was how natural the user interface was – one controller served as the paint brush, while the other was the "palette" that contained different brush strokes, a color chooser, and utilities. The way the menu was designed for the Vive's controller was exceptionally executed, and I found myself able to switch between the various controls with ease after just a couple of minutes in the app. After drawing myself a beautiful night sky, I sat on the floor surrounded by color and took in the simplistic yet exceptional environment created by glowing light.

After Tiltbrush, we got a chance to play with a little scene that was inspired by DotA, inside a small cottage filled with little nooks and crannies to explore. This one was immensely impressive from an asset standpoint, and I found myself jumping quite a bit when facing a Liv-sized spider and hungry frog. My eyes tightly shut when faced with the arachnid, I didn't spend too long in this demo before switching it out for something a little friendlier.

I had heard folks talk about Job Simulator before, and to be perfectly honest, I had never quite grasped why the idea of virtual reality cooking was so appealing – but after giving it a try, I definitely got it – it was one of my favorite demos! The idea behind the Job Simulator demo is that you are in a kitchen and given a series of instructions for "cooking" various recipes from a robot overseeing your progress. Hilarity ensues as you grab different ingredients and make meals, throwing a bunch of them around the kitchen and dropping eggs. At one point, I was so engrossed in the experience, I almost dropped the controllers when I went to place them on the virtual counter top! The overall experience was the definition of 'delightful'. The only downside was that the level of interactivity provided made it more noticeable when elements in the scene couldn't be moved.

Last up was a Portal experience – and full disclaimer, I haven't played the actual game yet – where you were part of a Human Diversity Initiative and told to repair a robot. It was oddly disquieting, and I walked away engrossed in thoughts of AI. It was amazing how vividly the experiences stood out in my mind – a lot of the memories are as clear as if they were physical experiences.

I was impressed with how light the headset was, and it was a little more comfortable than my DK 2 is – I think I ended up wearing the headset on and off for about an hour and a half, and while it didn't quite fit as snuggly as my DK 2, it wasn't ever registering as being heavy on my face, which was nice. The controllers were a great combination of comfortable and sturdy, and the limited haptic feedback was enough to feel extraordinarily realistic in a lot of the different scenarios I played around in.

Final Thoughts

I get to try a lot of virtual reality technology, and the afternoon spent testing the Vive was hands-down one of the best times I've had since I started working on VR development. I desperately can't wait to add the Vive to my own collection of devices, and I'm looking forward to seeing what new content starts coming out for the Vive when it becomes more widely available. I'm keeping my fingers crossed that I might get selected for a developer kit (think of the kittens!) but either way, I'm now entirely sold that hands are a "must-have" for VR experiences that excel. The presence is unmatched, and I'm looking forward to seeing how the Oculus Touch controllers compare (hopefully at Oculus Connect)!

I cannot WAIT to see how this technology improves over the next several years: It's already pretty mind-blowing, and it's only going to get better.

OSVR HDK

The OSVR headset (short for Open Source Virtual Reality) is an open sourced headset that has a variety of options for developers and consumers to get started with desktop virtual reality at a lower price than is currently offered by Oculus or HTC. The OSVR headset can be purchased as a standalone device, but the schematics are also available for electronics-savvy individuals to build their own variations of the hardware. The modular approach supports customization at many levels, allowing for users to swap out parts as desired for their own custom builds.

FOVE

Fove is an under-development desktop HMD that includes pupil tracking technology to pinpoint exactly where a user is looking within the device. One of the major benefits of this type of tracking technology is the support of foveated rendering, where graphics are rendered specifically where the user is looking and are able to more closely mimic the way our eyes actually focus in the physical world.

PlayStation VR

Although not strictly a "desktop" VR headset, PlayStation VR (code named Project Morpheus) is an upcoming headset from Sony that is built on top of their PlayStation 4 hardware. The headset, according to Sony's website, will contain a 5.7" 1080p screen with up to 120 fps, and will be available for $399 beginning October 2016.

Other Types of VR & Immersive Technology

While mobile and desktop HMDs capture most of today's market, there are a few companies working on standalone devices that can be worn and used independently of a phone or desktop computer. These devices are generally on the more expensive end of the HMD market, and contain a variety of different hardware components to remove the need for an external computing device. Although not strictly a VR device, one such standalone device is the Microsoft HoloLens, which is a fully self-contained device running Windows 10.

In addition to the head mounted display technologies listed above, there are also other approaches to creating virtual environments, either through the use of several projectors showing images on walls in an enclosed room or with specialized systems such as a CAVE. These systems are usually quite a bit costlier than today's HMD devices, but have their own unique set of benefits over HMDs.

For brevity and as a result of platform (im)maturity, this book will focus on virtual reality, rather than getting into the specifics of mixed/augmented reality, though many of the techniques discussed in this book about developing for VR apply to those devices and platforms as well.

TYPES OF VIRTUAL REALITY APPLICATIONS

There are many different approaches to creating virtual reality content, and new types of capturing and making new content are being developed as this is being written. Generally speaking, immersive content will fall into one of several categories, or be a hybrid of a number of different approaches for creation.

360° Photography and Videography

360 degree applications, either in still form or video form, are passive virtual reality experiences where the viewer stands within and watches an environment that is surrounding them without taking an active part in the action.

These types of applications are shot on specialized cameras for filming immersive video, or on a series of regular cameras that capture the full range of visuals and are later stitched together to create a full effect. Smartphone applications can also be used to create still images by capturing a series of photos that are then automatically merged to form an equilrectangular projection.

Figure 2: A photograph with an equirectangular projection[1]

Companies such NextVR and JauntVR are two players working on live broadcasting events in 360, while other companies such as Samsung, Facebook, Nokia, and GoPro are all also working on their own custom camera hardware.

360 video content can be captured and viewed in both 2D and 3D formats, with popular networks like YouTube and Facebook supporting 360 photos and videos on their networks. Cameras such as the Ricoh Theta can be purchased for consumer creation at about $250 on Amazon, while professional-grade stereoscopic camera rigs that shoot in 3D can cost upwards of $50,000.

While 360 images and video can be great for capturing and streaming live events in an immersive format in real-time, it has the downside of being a static application – one where the user doesn't have the ability to move around and interact with.

Volumography & Photogrammetry

Another technique for capturing places that exist in the physical world and bringing them into virtual reality is a computer-generated mesh reconstruction of a location or an event. Two

[1] Photographed by Luca Blada, Licensed under Creative Commons 2.0 (https://creativecommons.org/licenses/by/2.0/)

approaches to this are volumography, which is the act of using light to calculate distances for an object or 3D space, and photogrammetry, which uses a series of photographs to recreate 3-dimensional objects or rooms.

Companies like Realities[vi] are using photogrammetry to enable virtual tourism, while the startup 8i is focusing on using volumography to capture human experiences in 3D. These experiences capture real world places and people, and convert them into experiences that can be saved and built upon using additional software development capabilities.

Real-time Computer-Generated Applications

To create fully interactive experiences for virtual and augmented reality, you will need to build some portion of your application using computer-generated graphics and software components. Applications that are computer generated allow you to build applications that are drawn by the computer to create your scenes. The film and video game industries have been using 3D CGI for many years, and many of the development techniques for special effects and building computer games carry over into building immersive experiences.

There are tradeoffs with building CGI experiences – the development life cycle may be longer than it would take to capture an experience in real-time, and rendering times for life-like graphics can have significant overhead to building applications. Passive experiences, such as CGI seen in films, can be rendered once and then displayed, but interactive experiences with multiple viewing angles will generally be processed and rendered in real-time, putting some requirements on the types of computers that are used with virtual and augmented reality experiences – particularly around graphics cards.

Real-time CGI also contains a number of benefits for creating virtual and augmented reality applications – you aren't limited to what you can capture in the physical world, which opens up settings and environments that currently would be possible to capture. You can also program different responses and interactions to elements within your application, so that the user has control over the environment and performing different actions and tasks within your program.

Creating VR Experiences

EARLY CONSIDERATIONS FOR DEVELOPING VIRTUAL REALITY APPLICATIONS

Now that you're familiar with the platforms are available with virtual reality and the types of applications that you can build for VR, you're at a point where you can start thinking through your own ideas and figure out a plan of action to bring those to life. I encourage you to read through the next section with an open mind, thinking about how the answers to these questions can help guide you through the process of building your first virtual reality experience.

What hardware is available to me?

Not every VR developer has every device out there. Some development studios are working on exclusives, and content creators who are close to launching popular titles often have contacts at the major device manufacturers who give them exclusive access to developer hardware. When you are just starting out on your journey as a VR developer, it may not be feasible to get hands-on with the more expensive devices right away, which isn't a problem, but is something to help you narrow down the scope of what your first projects should aim for. Desktop-caliber VR headsets generally require a beefier computer than what you may already have, but mobile VR has created a lower entry point for developers who are just starting out.

You probably don't want to try to build a very specific type of game without access to specialized hardware that you may need, so keep that in mind as you start thinking about your application and the hardware requirements it may have. Not having the exact consumer device won't be a showstopper, but you may not want to dive right in with developing for Oculus Touch if you want to be able to start testing right away and have no experience with building VR apps.

How much money am I willing to invest?

You can begin developing for VR without a huge upfront investment. Although there was some backlash when Oculus announced the $599 price tag for the first version of the Rift, the current VR industry doesn't require that much of an investment right out the gate, especially if you're a developer who wants to try out an experiment or two before making the decision to develop full time for VR. At the lower end of the spectrum, Cardboard is a great entry level headset and provides a free SDK for getting started—great for anyone with an iOS or Android device and a preliminary interest in virtual reality hacking. Many of the basics for virtual reality development are around 3D programming in general (of course, the more serious you get about building apps, the more specialized knowledge will be required) and learning how to build an app for Cardboard to start with is still going to be helpful if you decide to make the investment later on to buy more resource-intensive devices and hardware.

There are VR devices coming in at just about every price point: Cardboard viewers start at about $12 on Amazon. If you're looking for something a little higher end, you can find other mobile viewers from $35, or, if you already have a Samsung Galaxy S6, S6 Edge, Note 4, or Note 5, you can consider a GearVR for $99. The desktop-based Rift will run you $599, AND the HTC Vive will cost $799. Desktop devices will also require a fairly powerful PC to run on.

On top of the device costs that you might have, consider putting aside some money for events and meetups. Many VR meetups will have a small cover fee to help address the expenses of venues and

food, but are invaluable resources for networking with other developers and learning about the news in the industry. Conferences can be another great resource for learning and seeing what other developers are working on, but can be costly if not added to a budget in advance.

What is the time commitment I'm looking for?

Going down the rabbit hole of virtual reality can be a time-sucking endeavor, but it is also an incredibly rewarding one. First, you read about the VR industry and start to get the prerequisite background knowledge about the device ecosystems, design strategies, and figuring out what device to buy. Then, you get your first headset and are inundated with fun new apps to try and spend time in. After that, you get to dive deep into a particular platform, learning new APIs and toolsets, and you might start talking to developers on Twitter or ZapChain. This is something that is within your control, but it definitely helps to think about how much time you want to commit to building an application or learning more about the ecosystem.

If you're just starting out with the industry and want a quick primer, I made a 10 minute overview video on the basics of VR development that help provide a 30,000 foot view of today's developer ecosystem that might help you flesh out an idea of what approach to take for your own learning strategy. You can find a series of videos on introductory virtual reality development at http://justavrshow.com/.

How can I make my existing knowledge work for me?

When I first began experimenting with VR development, I chose Unity as the game engine I wanted to work with because I had 4 years of C# coding experience on the .NET platform building applications for Windows. If you are a developer, there are quite a few options for building virtual reality apps today, so one thing to think about when you're deciding where to start is whether you want to learn an entirely new skill set, or build off of your existing developer knowledge to ease into it. If you are new to developing, or don't want to write code, I'll cover these, as well as other non-programming options, in more depth later on in this chapter.

If you are a web developer:

- UnityScript is a JavaScript derivative that can be used for scripting for Unity games and applications
- WebVR is an experimental API that uses Three.JS and WebGL to create VR-enabled websites in Firefox and Chromium
- A-Frame is a new markup language from MozVR that allows you to write VR content using an HTML-style language for browser-based VR

If you are an Objective-C or Java developer:

- Unity supports building apps to both of these mobile platforms using C#, UnityScript, or Boo for scripting
- Cardboard and GearVR have native SDKs for using Java to build native Android applications through your mobile IDE of choice, or Objective-C in Xcode.

If you are a C# developer:

- Unity supports C# scripting with rich 3D building tools in their editor

- Unity cross-platform export functionality to multiple platforms, including Android, iOS, Windows
- Windows Universal Applications can adopt the Cardboard SDK, or, if you're interested in HoloLens development, implement the Windows Holographic APIs

If you are a C / C++ developer:

- Unreal uses C++ as their scripting language in the editor
- Write directly to OpenGL with the Oculus SDK (Link to Oculus documentation)
- OSVR provides a Core repository and an Unreal plugin
- OpenVR API interfaces in C++

This isn't, of course, a comprehensive list, but it may help you figure out what may be most relevant for you. Developers just looking to see what goes into a sample application or poking around at an example codebase may find the resources at the end of this book particularly useful.

What do I want to build?

You don't need to know the exact answer to this question to start playing with VR development, but if you have a long-term idea or two in mind, you can use that to sculpt out how you want to structure your learning path and demo projects. If you know in the long run you want to write a VR-enabled website that makes heavy use of existing APIs and JavaScript libraries, working in WebVR from the start is likely going to be more helpful than sitting down with Unreal and learning how to use Blueprint. Also consider the different applications of virtual vs. augmented reality devices, and whether you have a specific device family in mind to build for.

In addition to helping you decide what tools you'd like to start working with, having a basic idea of what you think you may want to build in the future will help break down your projects into actionable, applicable learning steps. If your dream application involves beautiful environments and rich textures, learning how to use a terrain editor is a good first step. Curious about writing your own stereoscopic renderer to create innovating tooling solutions or upgrading an existing game that you've written from scratch? Playing around with a native SDK and the graphics pipeline will help you see how other implementations are being done.

Do I have a niche area of interest already?

If you're incredibly passionate about rich 3D audio, it might not make sense to get started from scratch learning a particular framework for, say, lighting and environment design. Consider what your interests are and how they may be relevant to the growing VR industry. See if you can find a few developers already working in the realm you might be interested in and ask if you can send them a few questions over email to see if it might be an area worth digging into more. If you've got an interesting idea, try looking at GitHub to see if there are any projects that you can read through and contribute to. Be creative!

BUILDING YOUR FIRST VIRTUAL REALITY APPLICATION

The individual requirements for building your first application will vary from person to person, but generally, any project that you build will likely go through some variation of the following stages. These steps can help you scope out an idea and turn it into an actionable project, or give you a structure to think about when starting out on your first VR project. You might also feel more

comfortable just diving in to a new project and tweaking it to create something new, letting the act of building the experience guide you. How you approach this is entirely up to you – this section is simply meant to act as a guiding framework for virtual reality applications.

Planning a Project

Some projects come into existence from messing around with different code samples and experimenting with different components of the development pipeline, but others may prefer to sketch out a project's structure and do some planning before diving in directly to creating something themselves. It took me years to learn how to start building side projects and coming up with my own requirements for apps that I wanted to build, so I wanted to share with you my approach to planning new applications and projects.

When planning a new project, I'll generally separate out the planning stage into several discrete steps:

Ideation:

You all likely have vague ideas of things that you'd like to create, but turning those ideas into reality can sometimes seem impossible, especially if you've just started learning a new technology. The ideation stage of planning an application or experience can help you scope what you want to build to a specific set of goals. At the end of the ideation stage, you'll be able to answer the question: "*What do I want to build?*"

A good takeaway from the ideation stage is a one or two sentence statement that defines what your application idea is.

Example: I will build a Google Cardboard mobile virtual reality application that teaches kids about the solar system by letting them explore different planets.

Project (Requirements) Planning:

This is the phase where your application begins to take shape into a series of actionable items. You'll start to get into the details about scenarios that you want to approach and the stories that you want to tell. You'll start to figure out what steps need to happen to create an experience, and identify where you'll need to bring different roles together. This is where you'll start to create a specification that will define what your application will look like, and how you'll approach the development process. At the end of the project planning phase, you'll be able to answer the question: *"What will my project do, and how will I know when we have successfully completed the experience?"*

A good takeaway from the project requirements planning stage is a list of tasks that you'd like to accomplish with building your application, and what needs to be done to complete those tasks.

Example: To build my space education application, I need to do the following:

Task	Action
Find the models for the planets and an environment for the space background	Buy space assets / find free assets on the Asset store, or make them myself with Blender
Build one scene for each planet, and a basic scene that lets users pick the planet to explore	Create 10 scenes in Unity and write in application behavior to navigate between them using the Cardboard SDK
For each planet, add in a user interface and explore interactions that show information about the planet	Learn about the user interface system to display text and build out simple UIs to show and hide information
Display information about a planet's size, length of year/day, distance from sun, temperature, and other important information	Gather information about each planet in the solar system and save this information to use in the application
Make the application available for download on the app store	Collect information about publishing an app on the Play Store and what steps need to be taken, then follow through on them

Additional Exercises

These exercises are designed to help you think about previous virtual reality experiences that you may have tried, and use that information to think about how you would go about building your own application.

Think about the different types of VR applications that you've tried and your opinions about them.

- What were some of the things that worked really well?
- Were they mobile or desktop – how did the platform choices change the overall experience?
- What were some of the things that you struggled with in using the app?
- How could you change those?

Take some time to make a list of general ideas that you'd like to see in virtual reality.

These can be specific or broad, just list out as many as you can think of. Maybe they're completely new ideas, or new approaches to some existing applications. The goal of this exercise is to practice thinking about forming ideas that you can build as part of your learning experience.

Choose the idea you're most excited about from your list.

It doesn't matter if the idea is a single experience or if you want to build the Oasis from Ready Player One – just pick the one you'd be the most excited to build. Think about each of your ideas and hone in on them. Think about the following:

- What makes your idea unique to virtual reality as a platform?
- Would mobile or desktop VR make the most sense for your audience?

- Who would you build this app for?
- What are the limitations with existing VR hardware that would need to consider?
- What features would you add?
- What would be your first step to building it?

Designing a VR Project

The field of design in the technology industry is increasingly important as software enters our lives through more and more avenues. Virtual and augmented reality in particular, as spatial platforms for computing, necessitate talented creative agents in the fields of human-computer interaction, psychology, visual design, and 3D art.

Among many other definitions, **design is the context of:**

- Imagining what an application looks & feels like
- Planning and creating prototypes / mockups of an application
- Establishing the aesthetic and interactions of an application
- Identifying how usage patterns fit in an application
- Defining what gets built

*The first rule of virtual reality design: do **not** make the user sick!*

Virtual reality devices are far more likely than a 2D display to contribute to a user getting simulator sickness – a particularly rough breed of motion sickness caused when an experience sends conflicting inputs to a user's *proprioceptive system*, the part of the human body responsible for our sense of self in the world around us. This is generally due to the visual input from a VR headset not aligning with the user's physical movement, which can be triggered or exacerbated by a number of factors within a virtual reality experience.

Areas of Design

Within the field of immersive technology, there are a number of different design areas that contribute to the creation of a VR or AR experience, and virtual and augmented reality technologies each have their own unique considerations on a per-platform and per-device basis. While this is not meant to be an exhaustive list of the specialties within VR design, the following list will provide you with a primer on several of the main areas of consideration within the engineering lifecycle of a virtual reality application.

Experience Design

Perhaps one of the biggest questions to address once you've narrowed down the idea you have into discrete action items is determining what the final experience is going to be like. Experience design specifies what the overarching theme of the application is that you're building. What is the flow that a user will go through to accomplish the goals you've set for them? Which stages are presented to the user, and in what order? How do the states of an application change dependent on a user's action? Is there a beginning, middle, and end? How are each of those defined?

Interface Design

The interface of an application in the field of human-computer interaction (HCI) as it relates to virtual and augmented reality is the bridge between the user and the experience. This may be the design and flow of a menu or series of menus that open when an application first begins, a control panel within an experience, gestures, or buttons to define different actions - among many, many other applications.

Interface design focuses on creating the patterns and objects that a user interacts with during the course of an experience to navigate within the experience and control different elements. This type of design may also include elements of interaction design, but can generally be thought of as the software component of HCI, whereas interaction design leans a little more into the human component side.

Some key considerations for virtual reality interfaces:

- Capturing and guiding someone's visual attention is a form of your interface
- Visual elements can be in 3D or be a 2D form factor attached to an object in 3D space
- Avoid eye strain by having multiple different user interface (UI) elements at varying distances or focal points from the user
- Less text = more enjoyment
- 2D screens should be curved
- Heads up displays ("HUDS") are generally not recommended
- Consider spatial (in-world) UI and contextual diegetic UI (in-world, attached to objects)

Interaction Design

Interaction design ties into interface design in how it defines and enables someone to interact with your application. Identifying the most comfortable use patterns based on controller or gesture input, and figuring out what is available on a platform by platform basis are two examples of interaction design in virtual and augmented reality. You may be working with a specific set of controls, or you may need to design different workflows across multiple platforms. You'll also want to consider the various states that your application can be in, and how your application adopts specific standards that are in place to remain consistent with what is expected from the system.

Intuitive interaction design in virtual and augmented reality is particularly important as the platform, being new, isn't always something that a new user will feel comfortable poking around in. Today, with something like a website, we have a sense of familiarity and can generally brute-force our way through a poorly designed web experience, but we cannot make the same assumptions for virtual and augmented reality tech at such an early stage.

Different VR devices require different strategies for handling interactions. Some inputs that you may run into or want to support within your own experiences:

- Gaze tracking
- Eye (Pupil) tracking
- Scroll pads (e.g. Gear VR)
- Single click (e.g. Cardboard)
- Gestures

- Controllers (standard or specialized, such as Vive or Touch controllers)
- Haptics

Ergonomic Design

A particularly important area of concern for virtual and augmented reality is the ergonomic factor of the experience. You want to be designing an application that is comfortable and safe for users to be in. Avoid placing objects in places that are challenging to interact with (e.g. above someone's head) or require detailed, repetitive movements within your gameplay.

With very few exceptions, your application's camera should generally align with a player's head movement. Some slow motions that pan throughout the scene can work, but you don't want to be forcing the primary direction to move counter to what the user is doing. Movement that doesn't align with the player's actions increases the likelihood of simulator sickness.

When building your application's environment, objects within your app should be scaled to that extensive reaching isn't required - especially important for room scale VR. Keep in mind that the user can't see what is around them in the physical world, and it's very easy to get turned around in virtual reality!

Environment Design

When building a virtual environment, it's important that you pay attention to several factors: the level of immersion that your environment adds to, the degree of exploration that you want available to your users, and how you capture the attention of your users within the environment to draw them to a specific part of the world.

Game engines such as Unity and Unreal have a built-in environment editor for building outdoor terrains, and you'll generally build out your environment within each of your scene. The components that comprise your environment (and anything within it) that represent physical objects are called models. Your environment also contains audio elements, lighting elements, user interface elements, and the camera representing the player themselves.

A common misconception with virtual reality is that an environment needs to be photorealistic in order to be immersive – but the human brain is surprisingly content in low-fidelity, cartoon-like environments when the experience is performant and well-designed. Your environment will be one of the places you have an extraordinary amount of creative control and enjoyable environments will encourage users to spend more time within your app.

Character (Model) Design

Character design is the act of designing and creating the characters within a world, complete with animations and behaviors. This type of design can cover:

- Creating the physical model
- Adding textures and visual elements to a character or model
- Programming animations and behaviors based on the player's actions (writing the AI)
- Specifying the personalities and traits of different things within your experience

Visual Design

Generally speaking, in the context of virtual and augmented reality, visual design is the process of creating a cohesive visual style for your application. This includes choosing and creating fonts, color palettes, stylistic designs for models and environments, and more to ensure that your application has the visual feel that you want to achieve.

Design Considerations for Different Devices

What platform are you building for?

Imagine an HTC Vive application that only had a single button input and the camera stayed in one place. What would be the purpose of using the Vive for something like that? One of the biggest draws for the Vive headset is the support for full room scale experiences and the two controllers providing interesting interactions that feel more natural to hold.

What input devices are available to your users?

A fairly common scenario that I see with mobile VR and AR applications is that the developer assumes that anyone who downloads the app will have a mobile bluetooth controller to navigate with. A user with a generic Cardboard viewer may find herself downloading applications not knowing they require a controller, and be left with apps that she can't actually use! There are a lot of different options for input, and Unity does a great job of abstracting out some of the specific details to support multiple devices, but when designing your experience, it's important to take into consideration what users will have and their experience as a result.

Additional Exercises

Taking the idea for your sample project, brainstorm and consider how you would design this application.

This is intended to be an open-ended exercise! Think about the platform that you think is most appropriate for creating your experience, and why. Brainstorm several potential approaches to the type of application that you could build, and choose your favorite. Decide on 2-3 features that your app would include and jot them down.

Using your idea from the first exercise and the features that you've chosen, or a project idea of your own, choose a potential "scene" that you can create.

Physical prototyping can be extraordinarily helpful in determining layouts and elements for a 3D environment, especially to help establish a sense of positioning. Grab some paper, a pen, and start prototyping your scene. Consider the following:

- Where would your user interface go?
- How can you represent different types of interactions and behaviors?
- How can you create sketches to help solidify different aspects of design for your app?

Using your prototype as a guideline, think about the scenes you have in mind to create and answer the following questions:

- How well do you think your experience would translate into an actual virtual reality device?

- Which platform did you envision yourself using while you were designing the experience?
- Was it the same as the one that you had originally thought would make sense?
- How would you need to change your experience to adapt it to another platform?
- If you chose something like Cardboard, how could you change your feature set if you were to design it for an Oculus or HTC Headset? If you chose a desktop system, how would you need to redesign the functionality to work for Gear VR or Cardboard?

Understanding your Development Options

The approach that you will take to developing your first virtual reality application will depend on your previous experience with programming, what you'd like to learn, and the platform that you'd like to target. Because of the variation in workflow with each development environment, I won't go into a step-by-step tutorial for each possible option (by the time it was written, something would have changed!) but this section will provide an overview of the options and when they are good options to choose for different types of projects. Use this section to think about what you would be most interesting in pursuing further.

The next section of this book will provide an in-depth set of resources for the different virtual reality development tools and tutorials for learning more about different areas of VR and AR development.

Game Engines

Unity

The Unity Editor is a popular choice for virtual reality applications due to its extensive support of virtual and augmented reality devices and platforms. Unity, which can be used to build for Oculus, the HTC Vive, HoloLens, iOS, Android, and the web, among other platforms, contains a feature-rich editor that can be used to create environments for cross-platform 3D applications.

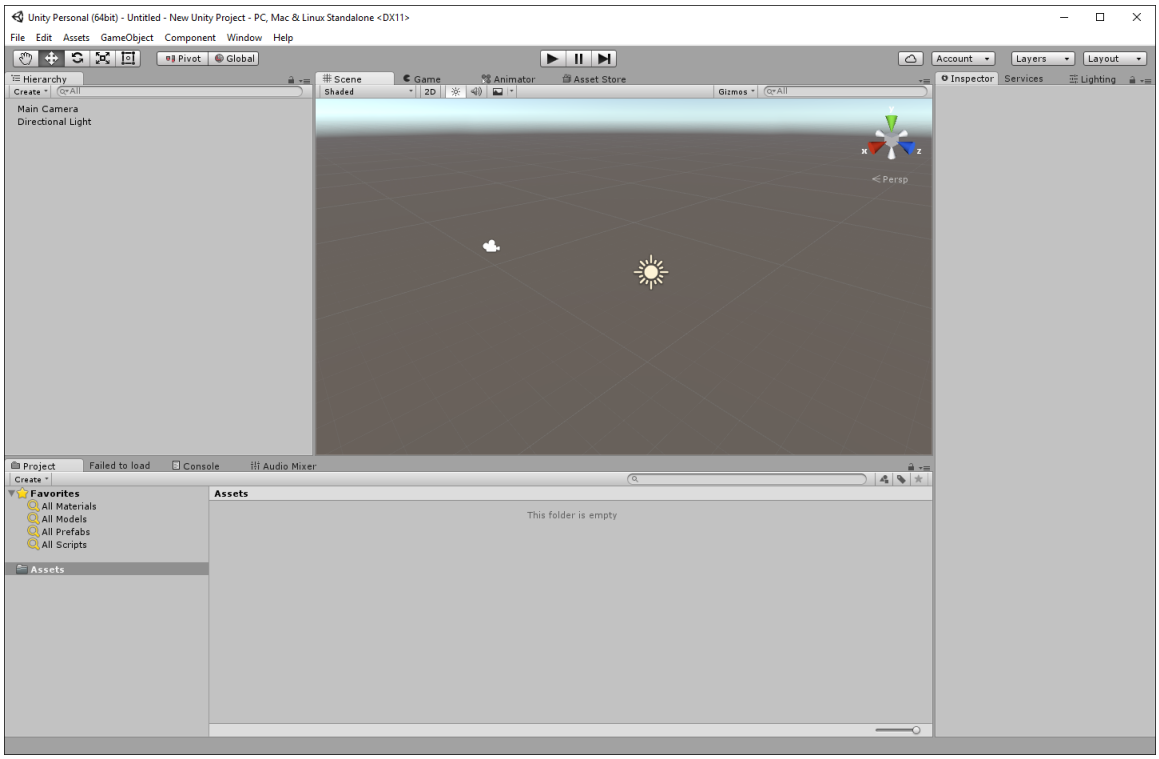

Figure 3: A screenshot of the default Unity Editor, version 5.3.4

Many virtual and augmented reality platforms include plugins that provide support in Unity, allowing developers to quickly and easily implement functionality for different devices and inputs. Unity has an extensive set of resources for new developers to learn from, including a number of full projects and available assets to use in projects.

Unity allows developers to write scripts to control elements of their application in C#, UnityScript (a derivative of JavaScript), and Boo, while also providing a lot of built-in functionality to support common application elements like physics, lighting, animation, and spatial audio.

Unreal

Unreal Engine is another popular game engine for virtual reality developers. Unreal Engine 4, a product of Epic Games, is a fully featured editor for building highly realistic environments and scenes for 3D applications. Like Unity, it provides functionality to support common application requirements, works across a number of different platforms, and has support for plugins to enhance functionality of the default editor.

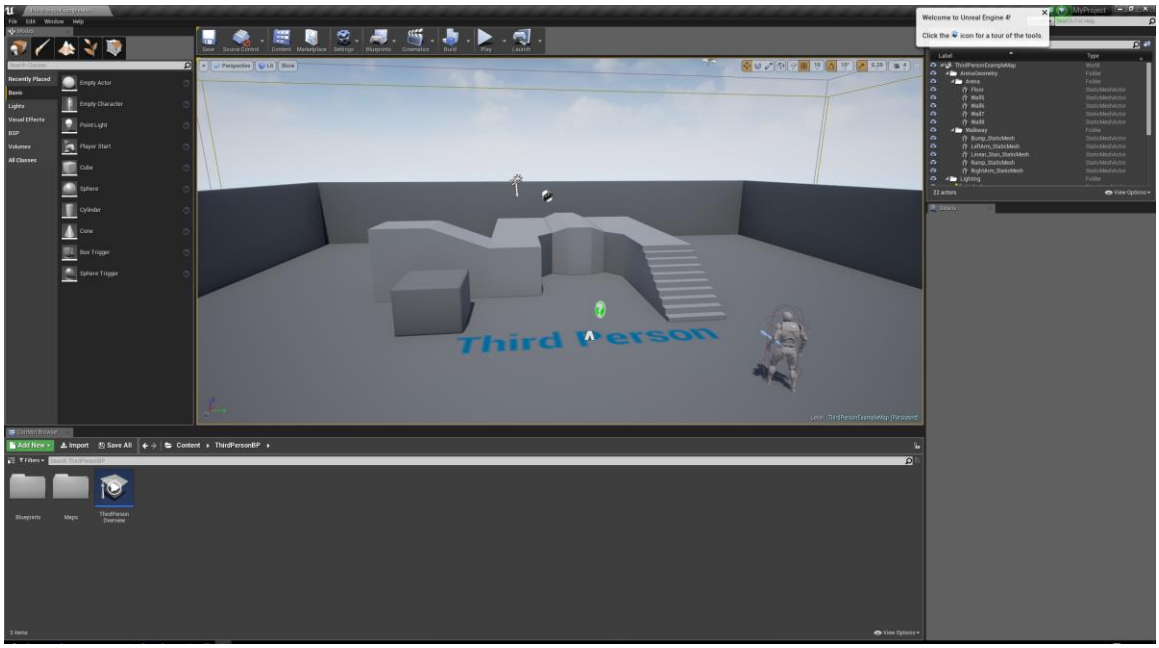

Figure 4: Unreal Engine 4.1 with the Third Person Template Loaded

Many popular virtual and augmented reality platforms have a development option for Unreal Engine, but they may not be as fully featured as the corresponding plugin for Unity. Unreal provides two options for scripting behaviors in your application: you can write C++ code, or use a visual programming language called 'Blueprint', which is a node-based, graph-like way to develop interactions for the actors within your scenes.

CryEngine

Crytek's CryEngine development environment is another option for developers looking to work with a game engine for virtual reality development. The CryEngine editor supports most of the major virtual reality devices through its own editor, and has a pay-as-you-can model for users.

The VR Web

WebVR Framework

The WebVR API[vii] is an under-development API that lets a web browser communicate with virtual reality devices that are attached to the computer. It also allows a mobile phone to act as a headset display, allowing websites to become VR apps that run on both desktop and mobile devices. Applications that utilize the WebVR API can currently run in the Firefox Nightly browser, and specific builds of Chromium, as well as on mobile phones.

At the time of this writing, the WebVR API has reached 1.0, and is expected to land in Chrome sometime around the end of 2016. It has also been proposed to the W3C as an official web standard, and many 3D JavaScript graphics libraries, such as Three.JS and Babylon.JS, have support for WebVR included.

Why You Should Care (At Least a Little) About The VR Web

October 19th, 2015

I'm a huge fan of WebVR. I've written some getting started posts about it and it's one of my favorite things to talk about at conferences. Even within the growing VR community, though, WebVR is a hot topic of discussion. John Carmack at Oeculus Connect 2 this year seemed skeptical about the future of WebVR, saying that he wasn't writing it off but appearing to hesitate on its potential. I've had huge numbers of people reactive positively to showing off a quick WebVR demo on my phone when I talk about what I do, but I've also had several people berate WebVR for not hitting 90FPS constantly and complaining that there was no valid use case scenarios as of yet.

I'm inclined to disagree with that last part – I often get people asking me why I'm so interested in the web as a potential virtual reality platform, so I figured I'd jot this down and open myself up further to the world of debate around the topic.

The internet isn't going anywhere

The web has exploded over the last decade into a host of services that dictate our actions. Amazon rules retail and it's never been easier to buy whatever you want without leaving the apartment. As VR opens up a new way to showcase goods and services, WebVR is a perfect way to mix VR content with non-VR content on the existing internet. Got your headset plugged in, and see a lamp you're interested in? Now you can see it in VR to get a sense of scale and what it might look like in a room with your wall colors. Okay, so Amazon might not have "WebVR" support on their radar any time soon, but I'd like to think it's only a matter of time before you can view the VR version of a furniture store from the comfort of your own Oculus.

Simpler integration with web APIs

It is way easier to integrate existing web content into a WebVR application than into Unity, at least in my experience. This content, while available to native VR apps, allows you to use the huge trove of existing JavaScript libraries to build out complex applications without needing to customize how they're integrated into different game engines. This comes in handy when you don't want to write something from scratch to integrate with the latest version of that framework you love so much.

Minimal getting started overhead

I've spent a lot of time mentoring at hackathons, teaching VR workshops, and showing random people that I meet my WebVR sites (sorry I'm not so sorry!) and it's far and away the easiest way to get people understanding what I do with VR development. I can give them a link to go to and they immediately see how their own phones are now VR devices, too. They don't need an Oculus to get started or a complex IDE – anyone who understands the basics of web development can try out the sample code immediately. JavaScript is growing in popularity – and that pool of developers getting excited about VR helps the industry advance.

No installations, no walled gardens – just content

This is pretty self-explanatory: VR on the web, just like traditional web content, is more open than curated app stores. I love not having to tell people that something I built doesn't work on their phone. I like that there's more freedom in the content that can be posted, at least at this stage of how the web is policed (this is another topic entirely, but at least at this point in time it's not individual OEMs deciding the content).

A-Frame

A-Frame is a project from the MozVR team that simplifies the creation process for VR-enabled websites using a markup-style language to display scenes using WebGL with VR support in the browser. It's built within the WebVR API specifications and utilizes the WebVR boilerplate template, extending the capabilities of webpages via HTML-styled tags for common 3D components.

As of this writing, A-Frame supports:

- WebVR-enabled Chromium Builds (Desktop VR)
- FireFox Nightly (Desktop VR)
- iPhone (Mobile VR)
- Android (Mobile VR)

Because A-Frame provides a layer over WebVR, the underlying JavaScript library supports a wide range of VR-friendly elements. They also display nicely in non-VR browsers, so your experience won't be broken for users on a standard desktop. In addition to a wide range of 3D elements to create immersive and interactive scenes, A-Frame elements support:

- Panoramic images
- 2D Content integration
- 360^{O} video

There are several ways to start working with A-Frame (which are all documented over on the A-Frame site) but the simplest method is to grab the minified JavaScript file and simply create an index.html page. Each of the A-Frame elements are prepended with an 'a-' within their tags, and are surrounded by <a-scene> tags to indicate where elements need to be rendered.

Sample A-Frame code:

```
<a-scene>
    <a-camera position = "0 1 0"></a-camera>
    <a-sky color = "#A1D4FF"></a-sky>
    <a-cube position="0 0 -5" rotation = "0 45 45" color="#6100ab"></a-cube>
    <a-sphere position ="-3 0 -5" color= "#AB00ab" height="2"></a-sphere>
    <a-cylinder position= "3 0 -5" rotation = "0 0 60" color = "#812FAE">
    </a-cylinder>
    <a-plane position = "0 -1 -5" rotation = "90 0 0" width = "10"
             depth = "10" color = "#999999"></a-plane>
</a-scene>
```

Native Development

More experienced developers may find that they want to have a more specialized degree of control over their applications – game engines do an excellent job of reducing the complexity for getting started with development, but much of the final project is abstracted away. In this case, developers may choose to work natively on their chosen device, and implement the experience from the ground-up without relying on a game engine to handle the common core mechanics.

Oculus SDK

The Oculus SDK comes in two flavors: the PC SDK, for developing Rift applications, and the mobile SDK, for creating apps for the Gear VR headset. The PC SDK is a C++ library for developing Windows VR applications using Visual Studio, while the mobile SDK is Java for developing an Android application.

SteamVR / OpenVR

Valve, the company behind SteamVR, has made their underlying OpenVR libraries available for developers to support a number of different headsets out of the box for their applications. While the full source code for these aren't available, developers can grab the OpenVR library from GitHub and get support for room tracking and hand controllers in their native applications.

OSVR

In addition to the HDK, OSVR also supports cross-device development with their open source library. The SDK supports their own headset, as well as enabling other desktop devices to work in a single application build.

Daydream / GVR

The latest Google VR (GVR) SDKs for Daydream mobile VR support native Android development for applications. The Google VR SDK provides a C/C++ reference for developers, and allows them to use OpenGL to create 3D experiences while allowing the GVR SDK to handle many of the VR-specific features.

Non-Programming Options

Vizor

Vizor Create is an online web application for node-based visual programming of virtual reality websites. Instead of writing text-based code, you can drag and drop notes that are connected to create a graph of the VR scene.

High Fidelity

High Fidelity is a world-building platform for virtual reality that focuses on user created content distributed across the computers of users who are accessing the worlds. They have their own world-building software that doesn't require a background in programming, but support a number of different headsets and peripheral devices.

Building Your Expertise

DOCUMENTING AND SHARING PROJECTS

Everyone has a story to tell, and figuring out how to tell yours is a rewarding experience. Not only does it give you the opportunity to share your knowledge and learnings with the world – it helps establish a personal brand and style, showcase your projects, and create a reference to go back to in future projects to build upon.

I am a firm believer that everyone should have a blog, but there are a variety of ways to document and share projects. For developers, sharing projects on GitHub can be a great way to build out a portfolio of work and technical skills. Designers have a variety of options for online portfolios that demonstrate their process. Anyone can find and submit conference talks on any number of subjects related to the immersive technology industry, from storytelling to coding, art, design – to name just a few.

Three Myths Holding You Back

The biggest misconception about finding and sharing your experiences is that it requires a significant amount of prior skill in writing, developing, and public speaking. "I'm a beginner, I can't teach! I'm a bad writer, I can't blog! No one wants to hear me speak, I don't know that I have anything to say!"

Myth #1: I need to be a good writer to blog

Having a blog is an incredibly useful tool for anyone looking to learn and connect with a new community, but for many, it feels almost unimaginable to have one for their own work. When I first began blogging, it was for myself – a simple way to document what I had done throughout different projects or comment on things that I found interesting and useful. As with most skills, practicing helps tremendously when it comes to confidence. My earliest blog posts were rough, and I still write somewhat rambling posts, but they've improved significantly over the two years that I've been focused on honing my skills.

You don't need to start off blogging publically – blogging sites like Medium and Wordpress will allow you to explicitly say who is able to read your posts. You also don't need to start off by setting goals on a certain posting frequency.

Some ideas:

- If you enjoy a virtual reality application you've just tried, jot down what you enjoyed about it. This can help you establish patterns in what works nicely in apps verses what doesn't, and help you start thinking critically about features that you might want to learn how to build

- When you work through a tutorial project or sample code, track what went well and what didn't. Even if you don't complete the project, this can help you remember the skills you've been learning and look at different ways that you can approach problems in the future

- Share your experiences when you attend any virtual or augmented reality events, or if you find a particularly interesting article about something going on in the industry

Posts don't have to be long to be helpful – one of my most-read blog posts of all time is a short little blurb that I wrote about fixing an obscure error in Visual Studio. I almost didn't even publish it, because it seemed like it would be a silly thing to share, so I was surprised to find it being read as often as it was! You never know what different people might find helpful and interesting, and even on the same topic, people all have different opinions and ways to approach different solutions when designing or programming.

Myth #2: My side projects aren't worth sharing

Throughout my college career, I worked on a fair number of small little projects that I didn't ever bother to save backups of or share with anyone – I had an assumption that they were too insignificant to bother talking about or documenting. As a result, despite the fact that I've been writing code for almost ten years, almost all of the code I still have access to is from the last three, when I decided to start documenting even the smallest sample projects and post them on GitHub.

In well-established developer communities, there are a thousand different ways to find 'Hello World', but today's immersive technology industry is working with new tools, platforms, and libraries. Imagine the video game industry in 1993, or mobile phone development in 2006 – virtual and augmented reality as a scalable consumer technology is just beginning. More voices are needed to help continue to help create resources for developing and creating on these platforms.

Myth #3: I'm starting completely from scratch

One of the most compelling aspects of immersive technologies is the degree to which experiences and applications mirror our physical world. Applications in virtual reality and augmented reality are less like software and more like new objects that interact with or mimic the reality that we've grown up and spent our lives in. A common misconception in the computing industry is that only pure programming roles are "technical" jobs, but the truth is that great software is built when the creators are coming from a variety of backgrounds and skill sets.

No matter what your previous experience is, the odds are pretty good that you've been developing skills that will help you with creating for virtual reality. With today's opportunities for online learning, the ability to take courses as a non-degree student at local universities, and a proliferation of free access to research and developer resourcing, it's never been easier to find a way to build on your existing knowledge to start creating immersive content.

FINDING AND BUILDING COMMUNITY

As I mentioned at the start of this book, my first conference was Samsung's Developer Conference. I had never been to a professional developer conference, and had only been studying VR development for a couple of months, but I impulsively took advantage of the $99 offer to the meetup attendees, took three days off work, and took the train into San Francisco.

Showing up to a conference alone for the first time can be an intimidating experience. I wandered through the massive venue two hours before the keynote began, finally taking a spot over by a developer lounge and opening up Twitter.

As luck or fate would have it, I was quickly joined by a couple of guys who had just come up from southern California – to my delight, they were in the same situation I was, just starting out learning VR development and figuring out the ecosystem. We went to the keynote sessions together, and they introduced me to more people to meet. I started recognizing people who I had been talking to on Twitter, and seamlessly found myself welcomed into the VR development community with open arms.

Having people who share your passions and encourage your growth is a monumental help when starting the transition into a new industry. The virtual reality community as a whole is immensely open and collaborative, something I found important to contribute to throughout my own journey. Today, cities and regions around the world are building their own communities in immersive technologies, and specialized areas of focus in different industries and professional roles are forming to bring together people with a variety of interests and skills relating to VR and AR.

The internet has made it easier than ever to find and network with like-minded individuals interested in learning and building virtual and augmented reality experiences. I strongly recommend that you think through this section and consider how you will start finding and connecting with other VR and AR professionals and enthusiasts.

Meetup Groups

Meetup.com is an excellent resource for finding communities to join, and many of them are free or low-cost to attend. With virtual reality meetup groups, not only will you have the opportunity to meet other women and men passionate about the technology – you'll also probably get to try demos of new applications and hardware devices!

If you live near a major metropolitan area, you'll likely have a few options of meetup groups in your area that focus on immersive technology - some groups will likely the entire spectrum of immersive technologies, whereas others may have more specific focuses, such as a particular type of technology or application of VR and AR tech.

Making the most out of a VR meetup:

- Arrive early – the busiest times for demos at meetups tend to be after any talks are completed and the networking portion begins. If you're able to arrive at the event right when doors open, it likely will be less crowded, which gives you more opportunities to talk to developers and people showcasing demo apps and hardware.

- Talk to as many people as you can. Showing up to an event where you don't know anyone can be a little intimidating, but virtual and augmented reality is still a young enough technology that most people in the industry are extremely friendly and welcoming to newcomers. Ask people there what they work on or if they're demoing anything – let them know you what you're up to, and ask if they have any advice!

- Make it a point to meet and thank the organizers if you can. There's a lot that goes into organizing and running successful meetups, and getting to know the people running the events can help a lot in finding people to help you navigate networks and answer questions about the industry.

- Keep an ear out for other upcoming events that are good to attend. With more and more popping up each month, it can be easy to get overwhelmed with potential meetups, events, and conferences, so chat with different people you meet and see if they have any recommendations for where you should head to if you're looking to learn more.

- Try demos! Many virtual reality meetup groups will have an open demo policy, and while some people may have tables set up to demo, you never know who will have a mobile VR app they're working on that they're waiting to pull out.

- Share what you're working on. Once you're building prototypes and have some of your first applications that you're building, talk to other creators about their process and for feedback. When you first start working on applications in your free time or as side projects, it can be hard to get exposure to other developers, so take advantage of being surrounded by other people working on their own virtual and augmented reality applications and talk about processes. Compare tools and ask questions about their workflow.

- If this is your first time at a meetup, tell people that you meet! They'll usually be more than happy to talk to you about what it is that they're working on and offering their own stories about how they got involved.

A few of my favorite questions to ask at meetups:

"How did you get into VR?"

"Are you demoing anything today?"

"What's your favorite thing that you've seen so far?"

"When did you start working on this project / at this company?"

Starting & Running a Meetup:

If you're having a hard time finding an in-person organization that you feel connected to, you might be interested in starting your own VR or AR meetup group. You don't need to commit to a location or date before creating a community on Meetup, so start your group and see who starts joining. Small meetups can meet casually, like at a bar, library, or restaurant, whereas larger groups may have better luck finding a sponsor company who will let groups come in after working hours to use their office space. Find one or two other people that have a similar interest in learning something, and build the type of event that you'd like to attend!

My First Virtual Reality Meetup

November 9th, 2014

There is a way to make dreams come true, and that is virtual reality. Several weeks ago, I stumbled across the Silicon Valley Virtual Reality meetup group and immediately RSVP-ed to their 15th meetup, conveniently located just two miles away from my apartment and scheduled for this past week. I set out with determination, headed to Coder Dojo, and arrived just as the doors were opening for the line of people who had begun forming outside of the back room. I was immediately awestruck by the number of demos people had managed to fill the room with and my attention was grabbed by a guy wearing an Oculus Rift running a demo based on my favorite movie series of all-time.

I must have looked as amazed as I felt, because as soon as he was done with the demo, he turned and asked if I wanted to try it. I have to say – my first experience with the Oculus Rift was definitely an awesome one. The lightsaber controls used the two controllers that I was holding (see image above) and it genuinely felt like I was training in force combat. I didn't notice any motion sickness, but I did get a pretty solid adrenaline rush every time I failed to defend myself from the lasers and wound up a little dizzy afterwards due to the all spinning I was doing (probably optional – I just got really into it). I had the most ridiculous grin on my face the entire time: I was actually getting to feel what Luke felt in the first Star Wars movie!

On a scale of 1 – the most awesome thing I've ever seen in a video game, this comes pretty close to being the best.

What I loved about SVVR is that everyone there was really fucking excited about virtual reality. At meetups that I've been to before, people were interested but also half paying attention to their laptop, only there for free swag – here, everyone seemed so excited about VR that the entire room had an energy about it that really seemed to embody the passion of the people there. I was promptly sucked into a discussion about an augmented reality headset that Seebright is working on, and got to see a prototype of it that seemed really promising for a hybrid virtual/physical world. I was two demos in and we hadn't even really gotten started! At seven the talks kicked off and I found myself hanging on to all of it – there is some really cool stuff in the pipeline!

Robert Wang, the CEO of Nimble VR, gave some insight into the product and showed a demo of how it worked. He was followed by Thibaut Weise and Doug Griffin of FaceShift, a company that does 3D modeling for facial tracking software, who talked about their latest tools for recognizing expressions and showed off some really cool demonstrations of how their software used the depth camera mounted to the top of a laptop to do real-time mapping to an animated character. We also got a chance to hear from Nicholas DiCarlo, Samsung's Vice-President of Immersive Technologies & Virtual Reality, about the Samsung Developer Conference next week (which I cannot WAIT for!) and got a few other updates from various other companies from around the room.

I am already counting down the days until the next meetup. It was just honestly that good.

Virtual Reality Hackathons

The term 'Hackathon' can seem intimidating when you consider the traditional connotation of hacking, but it's not about breaking into other people's machines and stealing data. Hackathons are awesome opportunities to dedicate a weekend to learn about something new. Often times, you'll walk away with a new project that you can point to and say "I did part of that!"

Every hackathon is different. Some are very specialized events for a specific technology hosted by a single company, while others are community-driven and bring in a number of different headsets and technologies to choose from. The attendees of a hackathon may be there for networking purposes, to learn new skills, or to compete – usually some combination of all of those things. The important takeaway from a VR Hackathon is that it's perfectly fine to go into an event without existing knowledge – you'll be surprised what you end up with after three days of working on a project with a new team!

If you're attending a hackathon for virtual or augmented reality, consider some of these questions to get your team off on the right foot:

- **Which platform are we building for?** In my experience, a polished, focused project written for one specific platform will win over a project that was less finished because it tried to do everything. This is often the case when projects are not scoped to an individual platform, especially if the team members don't already have expertise in the specific nuances of building to an individual platform. If you're building for Google Cardboard, pick one phone and stick with that. Don't waste time trying different builds for iOS vs Android to just figure out through trial and error which one works – the result is a lack of ownership and a defined set of requirements to work from throughout the event.

- **What devices do we have access to?** Some hackathons will provide hardware for attendees to work on. A hackathon may be device specific ("The HTC Vive Hackthon") and others may just be technology-focused, but when teams start planning out projects, they should keep in mind what they want to work on while being realistic about testing and building their apps. If your team doesn't have a dedicated device that you are hacking for, make sure you understand what the availability looks like. I've been to events where sponsors have brought some devices, underestimated how popular they'd be, and given them out on a "first-come, first-serve" basis. Some teams will still hack on their APIs and SDKs in hopes of being able to get a device at testing time, but this is a big risk and should be something the team is on board with. Unexpected issues pop up all the time in hackathons, and this is one I see frequently.

- **How well do we know our tools, and who can help us if we get stuck?** I love to use hackathons as ways to learn new toolsets, but it's important that your team communicates and is honest about skills on different tools and is in agreement on what stack is being used. Generally, if you are coming to a hackathon with the goal to use a new tool, I like to suggest that you only come in trying to learn one new tool. If it is your absolute first time building for the Oculus Rift, stick to the game engine you're familiar with. If you're sitting down with a new tool, make sure that you're scoping out your tasks to account for the learning curve. I've been at events where I'll see someone who is trying to learn a new SDK, for a new device, in a new language, and it's a struggle. I won't tell you anything is

impossible, but being overly ambitious is a very, very common cause of hackathon projects not being completed. It's also important to recognize who in the room can help you if you get stuck. This varies from event to event, but you should pay attention when sponsors introduce their mentoring teams to get a feel for who might be able to figure out a bug you get stuck on – I've had times where I've gotten stuck on sometime minute for hours, only to find out when I ask for help that it was a missing check box in a component. Such is the life of programming – but at a hackathon, it's important to have as few of those stoppers as possible.

- **Why are we here?** People generally come to hackathons for one of three things: to learn, to compete, or for fun. That's certainly not to say that there isn't a significant amount of overlap between why people are at a given event, but it's incredibly important to clarify these things with your team members at the start of the event. If you have one or two really competitive team members who want to win, it's going to be hard for the team to work cohesively if the rest of the members are just experimenting and don't wind up with every deliverable met. Make sure that your team members are all on the same page about what the goal is, and be honest about it!

Final Thoughts

When I first started writing this book back in May of 2016, I wasn't entirely sure what it would turn into. Over the past several months that it's taken to turn this from a jumbled bunch of words into a coherent story, so many things have changed in the industry. It's a completely exhilarating time to explore emerging technology as an area of study, regardless of background, and I continue to be amazed by things that I see each and every day.

One of the things that I struggled with during the process of writing this book was figuring out how to distribute and sell it. There is so much potential in the industry, and so much value in being able to share that. I also recognize that, as much as I'd like my words to hold permanence, the nature of emerging technology is that it is just that – emerging. It's fluid, and changing rapidly. What I try to capture in this book is more of the mindset, advice, and things that have helped me along the way, as well as *pointers to learning more and helping you tailor something that fits you and the path that you want to take.

It is a path that many are still struggling to take. If you've found value in this e-book, and presently or at some point are able to pay it forward, I ask that you do what you can to help others beginning their own journey to a career in the tech industry by donating or volunteering your time to one of many wonderful organizations that support underrepresented communities in STEM fields, a few that I've listed here:

- Girls Who Code
- Women Who Code
- Black Girls Code
- Code2040

I also want to say thank you to you, the reader – this journey, though young, has been absolutely incredible. I am so thankful for the opportunities that I have had to share my voice, and hope that you will find yours within the VR industry, too.

Resources

These links will help you go into more depth to learn different aspects of virtual reality design and development depending on where you find yourself most interested. Some of these are general virtual reality overview tutorials, or teach about underlying graphics principles, while others are links to the developer resources of the major virtual reality devices and platforms. This is far from a fully comprehensive list: I keep a running list at http://github.com/misslivirose/learnvr of resources that I find helpful, and encourage you to check there periodically for additional information about the changing industry.

GENERAL RESOURCES

Beginner Unity Maze Tutorial – this was the first full project that I built in Unity in 2015. While some of it may be outdated, this covers a lot of the basics of putting together an application that can be adapted to virtual reality. http://livierickson.com/blog/unity-tutorial/

Just A/VR Show – this is my web series of introductory videos that cover a variety of topics related to virtual reality development. Videos range from overviews of the industry to specific tutorials on WebVR, Vive, Unity, Cardboard, A-Frame, and VR design. http://justavrshow.com/

Learning Virtual Reality – Learning Virtual Reality: Developing Immersive Experiences and Applications for Desktop, Web, and Mobile by Tony Parisi is a great overview book that goes more in-depth for building applications on the Oculus, GearVR, Cardboard, and WebVR. Complete with code samples and available on Amazon.

3D DESIGN & MODELING, VISUAL DESIGN

Blender – Blender is a free tool for creating and rigging 3D models and simulations. You can download the program and find tutorials for getting started at: www.blender.org

Maya – Maya is an Autodesk program that allows you to build, animate, and render models and is an industry standard for 3D artists. You can learn more about Maya and download a free trial at: http://www.autodesk.com/products/maya/overview

ZBrush – Pixologic's 2D/3D sculpting and digital painting tool is another option for creating and texturing models for a VR experience. Find more on their website: http://pixologic.com/

DEVICE-SPECIFIC RESOURCES & DEVELOPER TOOLS

Unity – Unity (http://unity3d.com/) has a number of resources available for learning how to use their engine. Tutorials, videos, live training, and documentation can be found on their site at http://unity3d.com/learn

Unreal – Epic's Unreal Engine (http://unrealengine.com) also contains a developer portal where they host video content for getting ramped up on using their engine. You can learn more at: https://docs.unrealengine.com/latest/INT/Videos/

CryEngine – Crytek's game engine (http://cryengine.com/) has a creator series for learning about the engine, which can be found at: https://www.cryengine.com/tutorials

Oculus Developer Resources – these are the full documents for building on the Rift, including the required downloads, runtimes, and sample projects. http://developer.oculus.com/

Cardboard Sample Projects – the documentation for building on Google Cardboard (adaptable to other mobile devices as well) including a Unity plugin, native SDKs for building applications directly on iOS or Android. http://developers.google.com/cardboard/

SteamVR Developer Portal – the documentation for OpenVR, generally for developing with the HTC Vive. Other devices may also have support for OpenVR.
https://developer.valvesoftware.com/wiki/SteamVR

WebVR: As mentioned above, the VR web is just beginning to take shape and make it into new browsers. You can find out more information about the WebVR API at http://webvr.info and follow the W3C community group for WebVR at: https://www.w3.org/community/webvr/

A-Frame: At the time of this writing, A-Frame is at version 0.3.0. Information about A-Frame can be found at http://aframe.io. More general information about Mozilla's work on A-Frame can be found at http://mozvr.com

About The Author

Liv Erickson is a virtual and augmented reality developer, evangelist, blogger, author, and speaker currently living in San Francisco, California. She is a Hokie transplant to the Bay Area, and enjoys infrequent rock climbing, playing AudioShield, traveling, and playing with her cat Mosby. Liv is best known for her development on KittenVR, her blog, The Matrix is My Office, and giving talks around the world in an attempt to trick and/or convince people that they should, like she did, drop everything and immediately begin working on creating the Metaverse. You can (and should) follow her on Twitter @missLiviRose.

---

[i] PixelWhipt on YouTube: http://youtube.com/pixelwhipt
[ii] Meetup – 'Find Your People': http://www.meetup.com/
[iii] Unity3D – a game engine for 3D development with extensive virtual reality development support: http://unity3d.com/
[iv] Silicon Valley Virtual Reality, a community group for virtual reality enthusiasts in the San Francisco Bay Area: http://svvr.com/, http://www.meetup.com/SiliconValleyVirtualReality/
[v] My virtual and augmented reality development blog, The Matrix is My Office: http://livierickson.com/blog/
[vi] Realities is a virtual reality company that brings real-world locations into VR through photogrammetry to create virtual tourist experiences. Their application is currently available for the HTC Vive: http://realities.io/
[vii] WebVR API documentation: http://webvr.info/