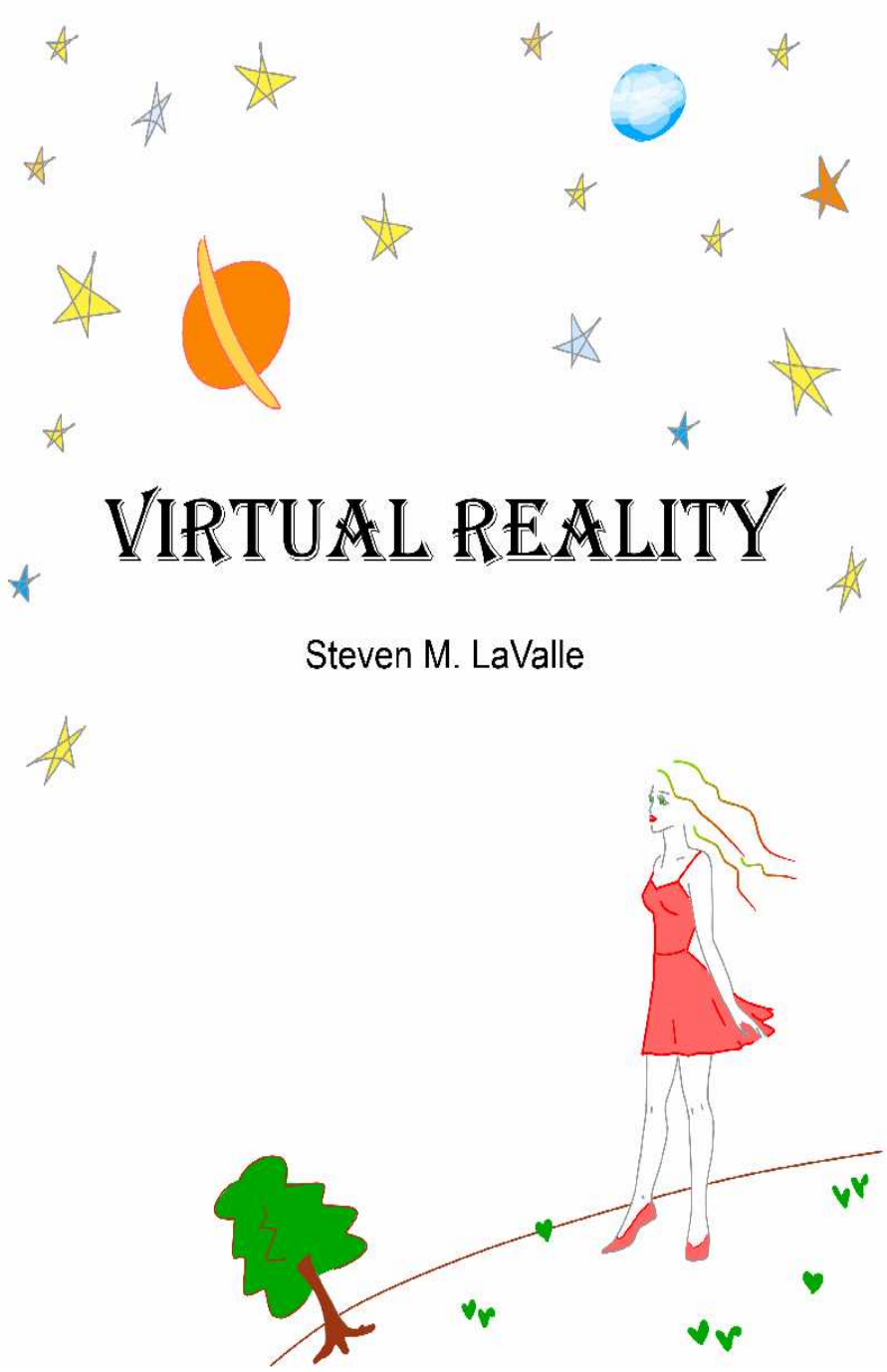


VIRTUAL REALITY

Steven M. LaValle

University of Illinois

Available for downloading at **http://vr.cs.uiuc.edu/**

DRAFT
June 29, 2016

Contents

Preface

The Rebirth of Virtual Reality

Virtual reality (VR) is a powerful technology that promises to change our lives unlike any other. By artificially stimulating our senses, our bodies become tricked into accepting another version of reality. VR is like a waking dream that could take place in a magical cartoon-like world, or could transport us to another part of the Earth or universe. It is the next step along a path that includes many familiar media, from paintings to movies to video games. We can even socialize with people inside of new worlds, either of which could be real or artificial.

At the same time, VR bears the stigma of unkept promises. The hype and excitement has often far exceeded the delivery of VR experiences to match it, especially for people without access to expensive laboratory equipment. This was particularly painful in the early 1990s when VR seemed poised to enter mainstream use but failed to catch on (outside of some niche markets). Decades later, we are witnessing an exciting rebirth. The latest technological components, mainly arising from the smartphone industry, have enabled high-resolution, low-cost, portable VR headsets to provide compelling VR experiences. This has mobilized leading technology companies to invest billions of US dollars into growing a VR ecosystem that includes art, entertainment, enhanced productivity, and social networks. At the same time, a new generation of technologists is entering the field with fresh ideas. Online communities of hackers and makers, along with college students around the world, are excitedly following the rapid advances in VR and are starting to shape it by starting new companies, working to improve the technology, and making new kinds of experiences.

The Intended Audience

The book is growing out of material for an overwhelmingly popular undergraduate course on VR that I introduced at the University of Illinois in 2015 (with hardware support from Oculus/Facebook). I have never in my life seen students so excited to take a course. We cannot offer enough slots to come even close to meeting the demand. Therefore, the primary target of this book is undergraduate students around the world. This book would be an ideal source for starting similar VR courses at other universities. Although most of the interested students have

been computer scientists, the course at Illinois has attracted students from many disciplines, such as psychology, music, kinesiology, engineering, medicine, and economics. Students in these other fields come with the most exciting project ideas because they can see how VR has the potential to radically alter *their* discipline. To make the course accessible to students with such diverse backgrounds, I have made the material as self-contained as possible. There is no assumed background in software development or advanced mathematics. If prospective readers have at least written some scripts before and can remember how to multiply matrices together, they should be ready to go.

In addition to use by students who are studying VR in university courses, it also targeted at developers in industry, hobbyists on the forums, and researchers in academia. The book appears online so that it may serve as a convenient references for all of these groups. To provide further assistance, there are also accompanying materials online, including lecture slides (prepared by Anna Yershova) and recorded lectures (provided online for free by NPTEL of India).

Why Am I Writing This Book?

I enjoy teaching and research, especially when I can tie the two together. I have been a professor and have taught university courses for two decades. Robotics has been my main field of expertise; however, in 2012, I started working at Oculus VR a few days after its Kickstarter campaign. I left the university and became their head scientist, working on head tracking methods, perceptual psychology, health and safety, and numerous other problems. I was struck at how many new challenges arose during that time because engineers and computer scientists (myself included) did not recognize human perception problems that were disrupting our progress. I became convinced that for VR to succeed, perceptual psychology must permeate the design of VR systems. As we tackled some of these challenges, the company rapidly grew in visibility and influence, eventually being acquired by Facebook for $2 billion in 2014. Oculus VR is largely credited with stimulating the rebirth of VR in the consumer marketplace.

I quickly returned to the University of Illinois with a new educational mission: Teach a new generation of students, developers, and researchers the fundamentals of VR in a way that fuses perceptual psychology with engineering. Furthermore, this book focuses on principles do not depend heavily on the particular technology of today. The goal is to improve the reader's *understanding* of how VR systems work, what limitations they have, and what can be done to improve them. One important component is that even though technology rapidly evolves, humans who use it do not. It is therefore crucial to understand how our sensors systems function, especially with matched with artificial stimulation. This intent is to provide a useful foundation as the technology evolves. In many cases, open challenges remain. The book does not provide the solutions to them, but instead provides the background to begin researching them.

Online Materials

The entire book is posted online at:

`http://vr.cs.uiuc.edu/`

along with pointers to additional materials, such as lecture videos and slides.

Suggested Use

This text may be used for a one-semester course by spending roughly one week per chapter, with the exception of Chapter 3, which may require two weeks. The book can also be used to augment other courses such as computer graphics, interfaces, and game development. Selected topics may also be drawn for a short course.

Depending on the technical level of the students, the mathematical concepts in Chapter 3 might seem too oppressive. If that is the case, students may be advised to skim over it and jump to subsequent chapters. They can understand most of the later concepts without the full mathematical details of Chapter 3. Nevertheless, understanding these concepts will enhance their comprehension throughout the book and will also make them more comfortable with programming exercises.

Lab Component

We currently use Oculus Rift DK2s on gaming PCs with expensive graphics cards (nVidia Titan Black with 6GB RAM). Development on many more platforms will soon become feasible for this course, including Samsung Gear VR, HTC Vive, and even Google Cardboard, but the quality is generally unacceptable. For software, almost all students develop VR projects using Unity 3D. Alternatives may be Unreal Engine and CryENGINE, depending on their level of technical coding skills. Unity 3D is the easiest because knowledge of C++ and associated low-level concepts is unnecessary.

Acknowledgments

I am very grateful to many students and colleagues who have given me extensive feedback and advice in developing this text. It evolved over many years through the development and teaching at the University of Illinois. The biggest thanks go to Anna Yershova, who has also taught the virtual reality course at the University of Illinois and collaborated on the course development. We also worked side-by-side at Oculus VR since the earliest days.

Among many helpful colleagues, I especially thank Ian Bailey, Paul MacNeilage, Betty Mohler, Aaron Nichols, Yury Petrov, Dan Simons, and Richard Yao for their helpful insights, explanations, suggestions, feedback, and pointers to materials.

I sincerely thank the many more people who have given me corrections and comments on early drafts of this book. This includes Blake J. Harris, Peter Newell, Yingying Ren, Matthew Romano, Killivalavan Solai, Karthik Srikanth, Jiang Tin, David Tranah, Ilija Vukotic, and Kan Zi Yang.

Steve LaValle
Urbana, Illinois, U.S.A.

Chapter 1

Introduction

Chapter Status	Taken from *Virtual Reality*, S. M. LaValle
	This online chapter is not the final version! Check **http://vr.cs.uiuc.edu/** for information on the latest draft version. This draft was compiled on June 29, 2016.

1.1 What Is Virtual Reality?

Virtual reality (VR) technology is evolving rapidly, making it precarious to define VR in terms of specific devices that may fall out of favor in a year or two. In this book, we are concerned with fundamental principles that are less sensitive to particular technologies and therefore survive the test of time. Our first challenge is to consider what VR actually means, in a way that captures the most crucial aspects in spite of rapidly changing technology. The concept must also be general enough to encompass what VR is considered today and what we envision for its future.

We start with two representative examples that employ current technologies: 1) A human having an experience of flying over virtual San Francisco by flapping his own wings (Figure 1.1); 2) a mouse running on a freely rotating ball while exploring a virtual maze that appears on a projection screen around the mouse (Figure 1.2). We want our definition of VR to be broad enough to include these examples and many more, which are coming in Section 1.2. This motivates the following.

Definition of VR: Inducing targeted behavior in an organism by using artificial sensory stimulation, while the organism has little or no awareness of the interference.

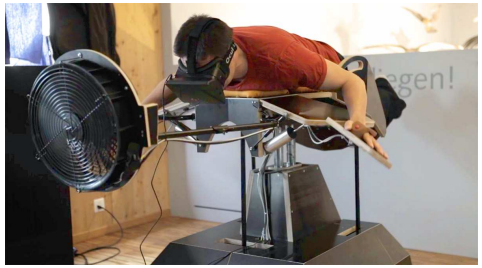
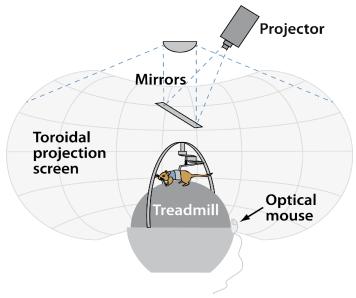
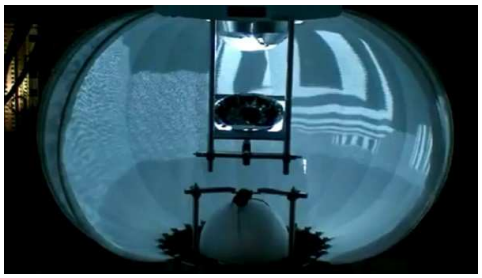


Figure 1.1: In the Birdly experience from the Zurich University of the Arts, the user, wearing a VR headset, flaps his wings while flying over virtual San Francisco, while a motion platform and fan provide additional sensory stimulation. The figure on the right shows the stimulus presented to each eye.



(a)



(b)

Figure 1.2: (a) An experimental setup used by neurobiologists at LMU Munich to present visual stimuli to rodents while they run on a spherical ball that acts as a treadmill (Figure by Kay Thurley). (b) A picture of a similar experiment, performed at Princeton University.

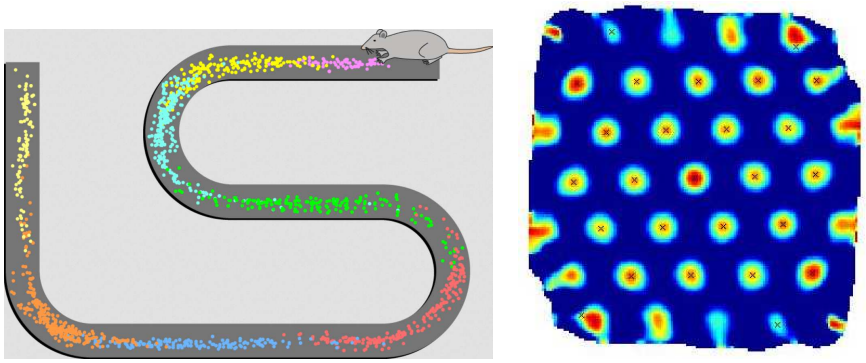


Figure 1.3: (a) We animals assign neurons as *place cells*, which fire when we return to specific locations. This figure depicts the spatial firing patterns of eight place cells in a rat brain as it runs back and forth along a winding track (figure by Stuart Layton). (b) We even have *grid cells*, which fire in uniformly, spatially distributed patterns, apparently encoding location coordinates (figure by Torkel Hafting).

Four key components appear in the definition:

- *Targeted behavior:* The organism is having an "experience" that was designed by the creator. Examples include flying, walking, exploring, watching a movie, and socializing with other organisms.
- *Organism:* This could be you, someone else, or even another life form such as a fruit fly, cockroach, fish, rodent, or monkey (scientists have used VR on all of these!).
- *Artificial sensory stimulation:* Through the power of engineering, one or more senses of the organism become hijacked, and their ordinary inputs are replaced by artificial stimulation.
- *Awareness:* While having the experience, the organism seems unaware of the interference, thereby being "fooled" into feeling present in a virtual world. This unawareness leads to a sense of *presence* in another world, or acceptance of it being natural.

Who is the fool? The idea of "fooling" an organism might seem fluffy or meaningless; however, this can be made surprisingly concrete using research from neurobiology. When animals explore their environment, neural structures composed of *place cells* are formed that encode spatial information about their surroundings [109]; see Figure 1.3(a). Each place cell is activated precisely when the organism returns to a particular location that is covered by it. Although less understood,

grid cells even encode locations in a manner similar to Cartesian coordinates [105] (Figure 1.3(b)). It has been shown that these neural structures may form in an organism, even when having a VR experience [26, 57]. In other words, our brains may form place cells for places that are not real! This is a clear indication that VR is fooling our brains, at least partially.

Terminology regarding various "worlds" Several terms related to VR are in common use at present. The term *virtual environments* predates widespread usage of VR, and is commonly considered to be synonymous; however, we emphasize in this book that VR can be an interface to a captured "real" world just as well as experiencing a completely artificial world. *Augmented reality* (AR) refers to systems in which most of the visual stimuli are propagated directly through glass or cameras to the eyes, and some additional structures appear to be superimposed onto the user's world. The term *mixed reality* is sometimes used to refer to an entire spectrum that encompasses VR, AR, and normal reality. *Telepresence* refers to systems that enable users to feel like they are somewhere else in the real world; if they are able to control anything, such as a flying drone, then *teleoperation* is an appropriate term. For our purposes, virtual environments, AR, mixed reality, telepresence, and teleoperation will all be considered as perfect examples of VR. The most important idea of VR is that the user's perception of reality has been altered through engineering, rather than whether the environment they believe they are in seems more "real" or "virtual". We will instead use these terms to distinguish whether VR is employed: The *real world* refers to the physical world that contains the user, and the *virtual world* refers to the perceived world as part of the targeted VR experience.

Interactivity Most VR experiences involve another crucial component: *interaction*. Does the sensory stimulation depend on actions taken by the organism? If the answer is "no", then the VR system is called *open-loop*; otherwise, it is *closed-loop*. In the case of closed-loop VR, the organism has partial control over the stimulation, which could vary as a result of body motions, including eyes, head, hands, or legs. Other possibilities include voice commands, heart rate, body temperature, and skin conductance (are you sweating?).

First- vs. Third-person If you are reading this book, then you most likely want to develop VR systems or experiences. Pay close attention to this next point! When a scientist designs an experiment for an organism, as shown in Figure 1.2, then the separation is clear: The laboratory subject (organism) has a *first-person* experience, while the scientist is a *third-person* observer. The scientist carefully designs the VR system as part of an experiment that will help to resolve a scientific hypothesis. For example, how does turning off a few neurons in a rat's brain affect its navigation ability? On the other hand, when engineers or developers construct a VR system or experience, they are usually targeting themselves and people like them. They feel perfectly comfortable moving back and forth between being

the "scientist" and the "lab subject" while evaluating and refining their work. As you will learn throughout this book, this is a bad idea! The creators of the experience are heavily biased by their desire for it to succeed without having to redo their work. They also know what the experience is supposed to mean or accomplish, which provides a strong bias in comparison to a fresh subject. To complicate matters further, the creator's body will physically and mentally adapt to whatever flaws are present so that they may soon become invisible. We have seen these kinds of things before. For example, it is hard to predict how others will react to your own writing. Also, it is usually harder to proofread your own writing in comparison to that of others. In the case of VR, these effects are much stronger and yet elusive to the point that you must force yourself to pay attention to them. Take great care when hijacking the senses that you have trusted all of your life. This will most likely be uncharted territory for you.

More real than reality? How "real" should the VR experience be? It is tempting to try to make it match our physical world as closely as possible. Our brains are most familiar with this setting, thereby making it seem most appropriate. This philosophy has dominated the video game industry at times, for example, in the development of highly realistic first-person shooter (FPS) games that are beautifully rendered on increasingly advanced graphics cards. In spite of this, understand that extremely simple, cartoon-like environments can also be effective and even preferable. Examples appear throughout history, as discussed in Section 1.3.

As a VR experience creator, think carefully about the task, goals, or desired effect you want to have on the user. You have the opportunity to make the experience *better than reality*. What will they be doing? Taking a math course? Experiencing a live theatrical performance? Writing software? Designing a house? Maintaining a long-distance relationship? Playing a game? Having a meditation and relaxation session? Traveling to another place on Earth, or in the universe? For each of these, think about how the realism requirements might vary. For example, consider writing software in VR. We currently write software by typing into windows that appear on a large screen. Note that even though this is a familiar experience for many people, it was not even possible in the physical world of the 1950s. In VR, we could simulate the modern software development environment by convincing the programmer that she is sitting in front of a screen; however, this misses the point that we can create almost *anything* in VR. Perhaps a completely new interface will emerge that does not appear to be a screen sitting on a desk in an office. For example, the windows could be floating above a secluded beach or forest. Furthermore, imagine how a debugger could show the program execution trace.

Synthetic vs. captured Two extremes exist when constructing a virtual world as part of a VR experience. At one end, we may program a *synthetic* world, which is completely invented from geometric primitives and simulated physics. This is common in video games and such virtual environments were assumed to be

the main way to experience VR in earlier decades. At the other end, the world may be *captured* using modern imaging techniques. For viewing on a screen, the video camera has served this purpose for over a century. Capturing panoramic images and videos and then seeing them from any viewpoint in a VR system is a natural extension. In many settings, however, too much information is lost when projecting the real world onto the camera sensor. What happens when the user changes her head position and viewpoint? More information should be captured in this case. Using depth sensors and SLAM (Simultaneous Localization And Mapping) techniques, a 3D representation of the surrounding world can be captured and maintained over time as it changes. It is extremely difficult, however, to construct an accurate and reliable representation, unless the environment is explicitly engineered for such capture (for example, a motion capture studio).

As humans interact, it becomes important to track their motions, which is an important form of capture. What are their facial expressions while wearing a VR headset? Do we need to know their hand gestures? What can we infer about their emotional state? Are their eyes focused on me? Synthetic representations of ourselves called *avatars* enable us to interact and provide a level of anonymity, if desired in some contexts. The attentiveness or emotional state can be generated synthetically. We can also enhance our avatars by tracking the motions and other attributes of our actual bodies. A well-known problem is the *uncanny valley*, in which a high degree of realism has been achieved in a avatar, but its appearance makes people feel uneasy. It seems almost right, but the small differences are disturbing. There is currently no easy way to make ourselves appear to others in a VR experience exactly as we do in the real world, and in most cases, we might not want to.

Health and safety Although the degree of required realism may vary based on the tasks, one requirement remains invariant: The health and safety of the users. Unlike simpler media such as radio or television, VR has the power to overwhelm the senses and the brain, leading to fatigue or sickness. This phenomenon has been studied under the heading *simulator sickness* for decades; in this book we will refer to adverse symptoms from VR usage as *VR sickness*. Sometimes the discomfort is due to problems in the VR hardware and low-level software; however, in most cases, it is caused by a careless VR developer who misunderstands or disregards the side effects of the experience on the user. This is one reason why human physiology and perceptual psychology are large components of this book. To develop comfortable VR experiences, you must understand how these factor in. In many cases, fatigue arises because the brain appears to work harder to integrate the unusual stimuli being presented to the senses. In some cases, inconsistencies with prior expectations, and outputs from other senses, even lead to dizziness and nausea.

Another factor that leads to fatigue is an interface that requires large amounts of muscular effort. For example, it might be tempting move objects around in a sandbox game by moving your arms around in space. This quickly leads to

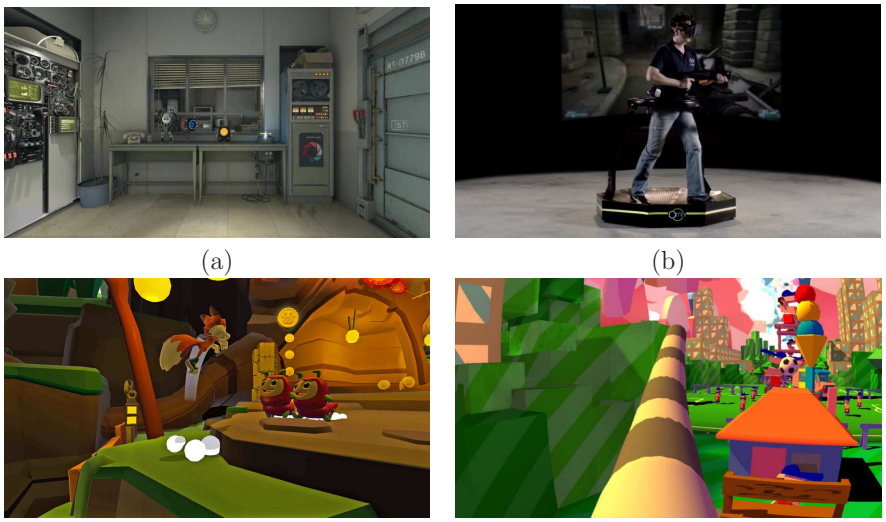
fatigue and an avoidable phenomenon called *gorilla arms*, in which people feel that the weight of their extended arms is unbearable. For example, by following the principle of the computer mouse, it may be possible to execute large, effective motions in the virtual space by small, comfortable motions of a controller. Over long periods of time, the brain will associate the motions well enough for it to seem realistic while also greatly reducing fatigue.

1.2 Modern VR Experiences

This section gives you a quick overview of what people are doing with VR today, and provides a starting point for searching for similar experiences on the Internet. Here, we can only describe the experiences in words and pictures, which is a long way from the appreciation gained by experiencing them yourself. This printed medium (a book) is woefully inadequate for fully conveying the medium of VR. Perhaps this is how it was in the 1890s to explain in a newspaper what a movie theater was like! If possible, it is strongly recommended that you try many VR experiences yourself to form first-hand opinions and spark your imagination to do something better.

Video games People have dreamed of entering their video game worlds for decades. By 1982, this concept was already popularized by the Disney movie Tron. Figure 1.4 shows several video game experiences in VR. Most gamers currently want to explore large, realistic worlds through an avatar. Figure 1.4(a) shows Valve's Portal 2, which is a puzzle-solving adventure game developed for the HTC Vive VR headset. Figure 1.4(b) shows an omnidirectional treadmill peripheral that gives users the sense of walking while they slide their feet in a dish on the floor. These two examples give the user a *first-person* perspective of their character. By contrast, Figure 1.4(c) shows Lucky's Tale, which instead yields a comfortable *third-person* perspective as the user seems to float above the character that she controls. Figure 1.4(d) shows a game that contrasts all the others in that it was designed to specifically exploit the power of VR.

Immersive cinema Hollywood movies continue to offer increasing degrees of realism. Why not make the viewers feel like they are part of the scene? Figure 1.5 shows an immersive short story. Movie directors are entering a fascinating new era of film. The tricks of the trade that were learned across the 20th century need to be reinvestigated because they are based on the assumption that the cinematographer controls the camera viewpoint. In VR, viewers can look in any direction, and perhaps even walk through the scene. What should they be allowed to do? How do you make sure they do not miss part of the story? Should the story be linear, or should it adapt to the viewer's actions? Should the viewer be a first-person character in the film, or a third-person observer who in invisible to



(a) (b) (c) (d)

Figure 1.4: (a) Valve's Portal 2 demo for the HTC Vive headset is a puzzle-solving experience in a virtual world. (b) The Virtuix Omni treadmill for walking through first-person shooter games. (c) Lucky's Tale for the Oculus Rift maintains a third-person perspective as the player floats above his character. (d) In the Dumpy game from DePaul University, the player appears to have a large elephant trunk. The purpose of the game is to enjoy this phenomenon while knocking things down.

the other characters? How can a group of friends experience a VR film together? When are animations more appropriate versus the capture of real scenes?

It will take many years to resolve these questions and countless more that will arise. In the meantime, VR can also be used as a kind of "wrapper" around existing movies. Figure 1.6 shows the VR Cinema application, which allows the user to choose any seat in a virtual movie theater. Whatever standard movies or videos that are on the user's hard drive can be streamed to the screen in the theater. These could be 2D or 3D movies. A projector in the back emits flickering lights and the audio is adjusted to mimic the acoustics of a real theater. This provides an immediate way to leverage all content that was developed for viewing on a screen, and bring it into VR. Many simple extensions can be made without modifying the films. For example, in a movie about zombies, a few virtual zombies could enter the theater and start to chase you. In a movie about tornadoes, perhaps the theater rips apart. You can also have a social experience. Imagine having "movie night" with your friends from around the world, while you sit together in the virtual movie theater. You can even have the thrill of misbehaving in the theater without getting thrown out.

Figure 1.5: Oculus Story Studio produced *Henry*, an immersive short story about an unloved hedgehog who hopes to make a new friend, the viewer.

Figure 1.6: VR Cinema, developed by Joo-Hyung Ahn for the Oculus Rift. You can choose your seat and watch any movie you like.

(a) (b)

Figure 1.7: An important component for achieving telepresence is to capture a panoramic view: (a) A car with cameras and depth sensors on top, used by Google to make Street View. (b) Bublcam is a cheap, portable way to capture and stream omnidirectional videos.

Telepresence The first step toward feeling like we are somewhere else is capturing a panoramic view of the remote environment (Figure 1.7). Google's Street View and Earth apps already rely on the captured panoramic images from millions of locations around the world. Simple VR apps that query the Street View server directly enable to user to feel like he is standing in each of these locations, while easily being able to transition between nearby locations (Figure 1.8). Panoramic video capture is even more compelling. Figure 1.9 shows a frame from an immersive rock concert experience. Even better is to provide *live* panoramic video interfaces, through which people can attend sporting events and concerts. Through a live interface, interaction is possible. People can take video conferencing to the next level by feeling present at the remote location. By connecting panoramic cameras to robots, the user is even allowed to move around in the remote environment (Figure 1.10). Current VR technology allows us to virtually visit far away places and interact in most of the ways that were previously possible only while physically present. This leads to improved opportunities for telecommuting to work. This could ultimately help reverse the urbanization trend sparked by the 19th-century industrial revolution, leading to *deurbanization* as we distribute more uniformly around the Earth.

Virtual societies Whereas telepresence makes us feel like we are in another part of the physical world, VR also allows us to form entire societies that remind us of the physical world, but are synthetic worlds that contain avatars connected to real people. Figure 1.11 shows a Second Life scene in which people interact in a fantasy world through avatars; such experiences were originally designed to view on a screen but can now be experienced through VR. Groups of people could spend time together in these spaces for a variety of reasons, including common

(a) (b)

Figure 1.8: A simple VR experience that presents Google Street View images through a VR headset: (a) A familiar scene in Paris. (b) Left and right eye views are created inside the headset, while also taking into account the user's looking direction.

Figure 1.9: Jaunt captured a panoramic video of Paul McCartney performing Live and Let Die, which provides a VR experience where users felt like they were on stage with the rock star.

(a) (b)

Figure 1.10: Examples of robotic avatars: (a) The DORA robot from the University of Pennsylvania mimics the users head motions, allowing him to look around in a remote world while maintaining a stereo view (panoramas are monoscopic). (b) The Plexidrone, a low-cost flying robot that is designed for streaming panoramic video.

Figure 1.11: Virtual societies develop through interacting avatars that meet in virtual worlds that are maintained on a common server. A snapshot from Second Life is shown here.

Figure 1.12: In Clouds Over Sidra, film producer Chris Milk offers a first-person perspective on the suffering of Syrian refugees.

special interests, educational goals, or simple an escape from ordinary life.

Empathy The first-person perspective provided by VR is a powerful tool for causing people to feel *empathy* for someone else's situation. The world continues to struggle with acceptance and equality for others of different race, religion, age, gender, sexuality, social status, and education, while the greatest barrier to progress is that most people cannot fathom what it is like to have a different identity. Figure 1.12 shows a VR project sponsored by the United Nations to yield feelings of empathy for those caught up in the Syrian crisis of 2015. Some of us may have compassion for the plight of others, but it is a much stronger feeling to understand their struggle because you have been there before. Figure 1.13 shows a VR system that allows men and women to swap bodies. Through virtual societies, many more possibilities can be explored. What if you were 10cm shorter than everyone else? What if you teach your course with a different gender? What if you were the victim of racial discrimination by the police? Using VR, we can imagine many "games of life" where you might not get as far without being in the "proper" group.

Education In addition to teaching empathy, the first-person perspective could revolutionize many areas of education. In engineering, mathematics, and the sciences, VR offers the chance to visualize geometric relationships in difficult concepts or data that is hard to interpret. Furthermore, VR is naturally suited for practical

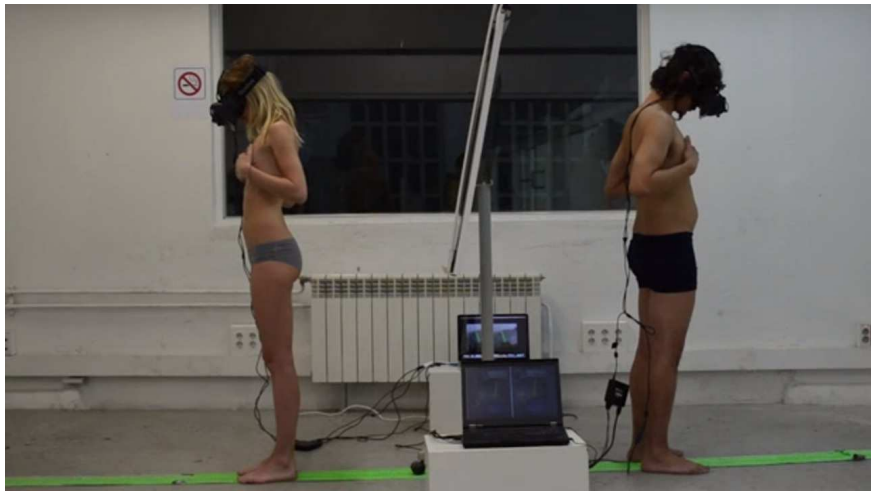


Figure 1.13: Students in Barcelona made an experience where you can swap bodies with the other gender. Each person wears a VR headset that has cameras mounted on its front. Each therefore sees the world from the approximate viewpoint of the other person. They were asked to move their hands in coordinated motions so that they see their new body moving appropriately.

training because skills developed in a realistic virtual environment may transfer naturally to the real environment. The motivation is particularly high if the real environment is costly to provide or poses health risks. One of the earliest and most common examples of training in VR is *flight simulation* (Figure 1.14). Other examples include firefighting, nuclear power plant safety, search-and-rescue, military operations, and medical procedures.

Beyond these common uses of VR, perhaps the greatest opportunities for VR education lie in the humanities, including history, anthropology, and foreign language acquisition. Consider the difference between reading a book on the Victorian era in England and being able to roam the streets of 19th-century London, in a simulation that has been painstakingly constructed by historians. We could even visit an ancient city that has been reconstructed from ruins (Figure 1.15). Fascinating possibilities exist for either touring *physical* museums through a VR interface or scanning and exhibiting artifacts directly in *virtual* museums.

Virtual prototyping In the real world, we build prototypes to understand how a proposed design feels or functions. Thanks to 3D printing and related technologies, this is easier than ever. At the same time, *virtual prototyping* enables designers to inhabit a virtual world that contains their prototype (Figure 1.16). They can quickly interact with it and make modifications. They also have the opportunities to bring clients into their virtual world so that they can communicate

Figure 1.14: A flight simulator used by the US Air Force (photo by Javier Garcia). The user sits in a physical cockpit while being surrounded by displays that show the environment.

Figure 1.15: A tour of the Nimrud palace of Assyrian King Ashurnasirpal II, a VR experience developed by Learning Sites Inc. and the University of Illinois.

Figure 1.16: Architecture is a prime example of where a virtual prototype is invaluable. This demo, called Ty Hedfan, was created by designer Olivier Demangel. The real kitchen is above and the virtual kitchen is below.

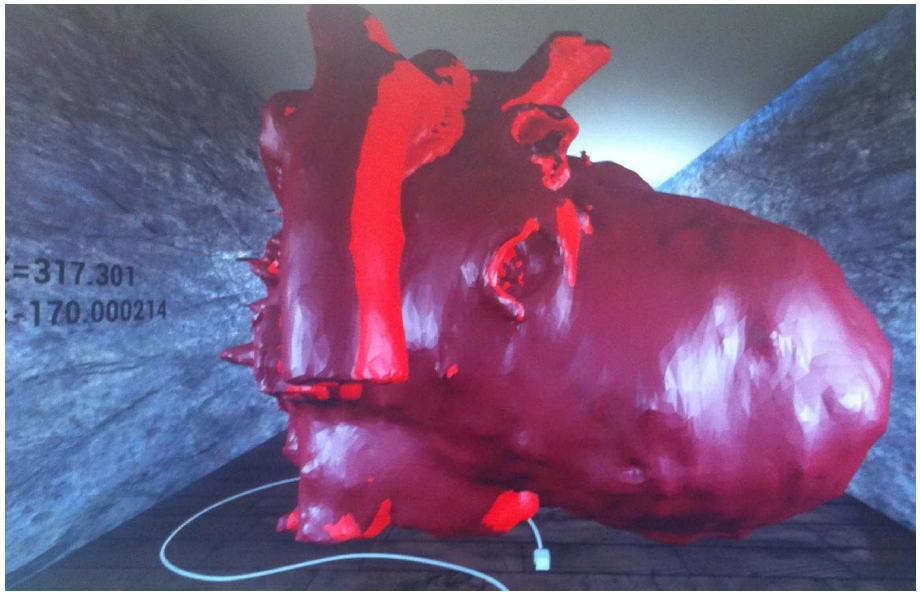


Figure 1.17: A heart visualization system based on images of a real human heart. This was developed by the Jump Trading Simulation and Education Center and the University of Illinois.

their ideas. Imagine you want to remodel your kitchen. You could construct a model in VR and then explain to a contractor exactly how it should look. Virtual prototyping in VR has important uses in many businesses, including real estate, architecture, and the design of aircraft, spacecraft, cars, furniture, clothing, and medical instruments.

Health care Although health and safety are challenging VR issues, the technology can also help to improve our health. There is an increasing trend toward distributed medicine, in which doctors train people to perform routine medical procedures in remote communities around the world. Doctors can provide guidance through telepresence, and also use VR technology for training. In another use of VR, doctors can immerse themselves in 3D organ models that were generated from medical scan data (Figure 1.17). This enables them to better plan and prepare for a medical procedure by studying the patient's body shortly before an operation. They can also explain medical options to the patient or his family so that they may make more informed decisions. In yet another use, VR can directly provide therapy to help patients. Examples include overcoming phobias and stress disorders through repeated exposure, improving or maintaining cognitive skills in spite of aging, and improving motor skills to overcome balance, muscular, or nervous system disorders.

(a)

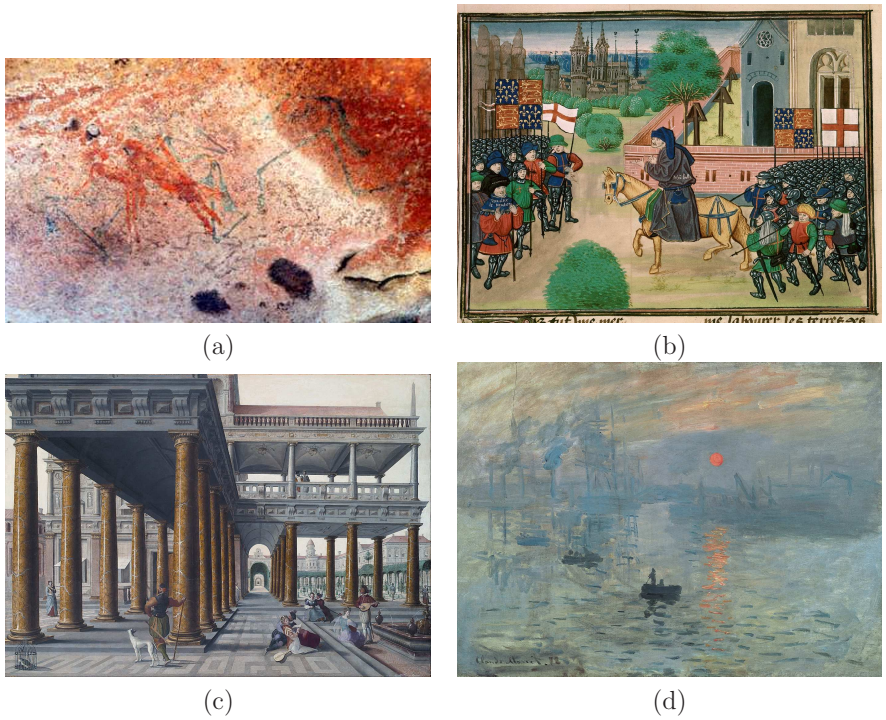
(b)

Figure 1.18: (a) Epic Games created a wild roller coaster ride through virtual living room. (b) A guillotine simulator was made by Andre Berlemont, Morten Brunbjerg, and Erkki Trummal. Participants were hit on the neck by friends as the blade dropped, and they could see the proper perspective as their heads rolled.

New human experiences Finally, the point might be to simply provide a new human experience. Through telepresence, people can try experiences through the eyes of robots or other people. However, we can go further by giving people experiences that are impossible (or perhaps deadly) in the real world. Most often, artists are the ones leading this effort. The Birdly experience of human flying (Figure 1.1) was an excellent example. Figure 1.18 shows two more. What if we change our scale? Imagine being 2mm tall and looking ants right in the face. Compare that to being 50m tall and towering over a city while people scream and run from you. What if we simulate the effect of drugs in your system? What if you could become your favorite animal? What if you became a piece of food? The creative possibilities for artists seem to be endless. We are limited only by what our bodies can comfortably handle. Exciting adventures lie ahead!

1.3 History Repeats

Staring at rectangles How did we arrive to VR as it exists today? We start with a history that predates what most people would consider to be VR, but includes many aspects crucial to VR that have been among us for tens of thousands of years. Long ago, our ancestors were trained to look at the walls and imagine a 3D world that is part of a story. Figure 1.19 shows some examples of this. Cave paintings, such as the one shown in Figure 1.19(a) from 30,000 years ago. Figure 1.19(b) shows a painting from the European Middle Ages. Similar to the cave painting, it relates to military conflict, a fascination of humans regardless of the era or technology. There is much greater detail in the newer painting, leaving



(a) (b) (c) (d)

Figure 1.19: (a) A 30,000-year-old painting from the Bhimbetka rock shelters in India (photo by Archaelogical Survey of India). (b) An English painting from around 1470 that depicts John Ball encouraging Wat Tyler rebels (unknown artist). (c) A painting by Hans Vredeman de Vries in 1596. (d) An impressionist painting by Claude Monet in 1874.

less to the imagination; however, the drawing perspective is comically wrong. Some people seem short relative to others, rather than being further away. The rear portion of the fence looks incorrect. Figure 1.19(c) shows a later painting in which the perspective have been meticulously accounted for, leading to a beautiful palace view that requires no imagination for us to perceive it as "3D". By the 19th century, many artists had grown tired of such realism and started the controversial *impressionist* movement, an example of which is shown in Figure 1.19(d). Such paintings leave more to the imagination of the viewer, much like the earlier cave paintings.

Moving pictures Once humans were content with staring at rectangles on the wall, the next step was to put them into motion. The phenomenon of *stroboscopic apparent motion* is the basis for what we call *movies* or *motion pictures* today.

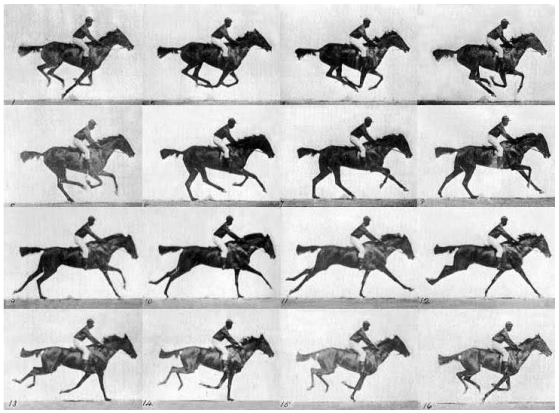
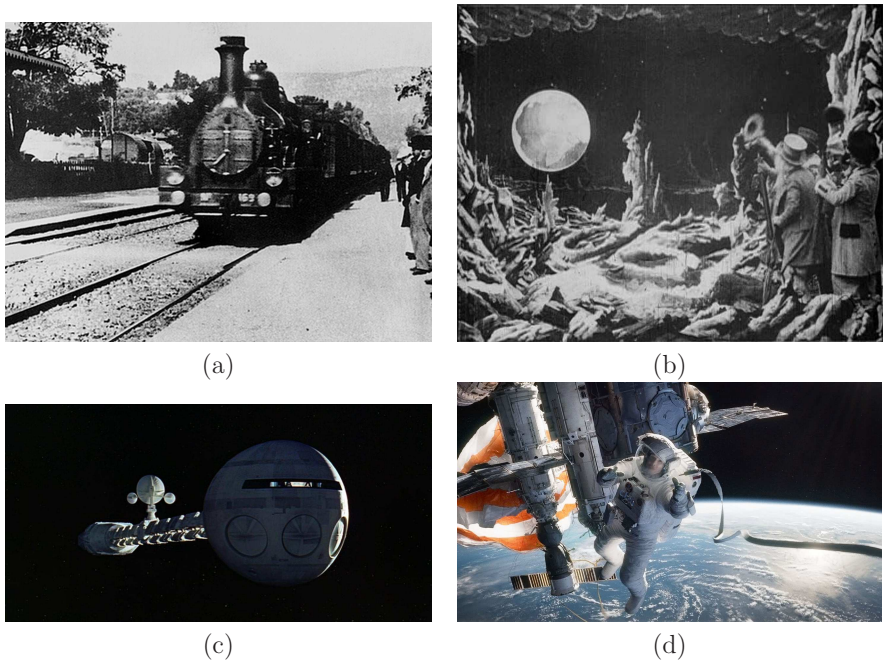


Figure 1.20: This 1878 *Horse in Motion* motion picture by Eadward Muybridge, was created by evenly spacing 24 cameras along a track and triggering them by trip wire as the horse passes. The animation was played on a zoopraxiscope, which was a precursor to the movie projector, but was mechanically similar to a record player.

Flipping quickly through a sequence of pictures gives the illusion of motion, even at a rate as low as two pictures per second. Above ten pictures per second, the motion even appears to be continuous, rather than perceived as individual pictures. One of the earliest examples of this effect is the race horse movie created by Eadward Muybridge in 1878 at the request of Leland Stanford (yes, that one!); see Figure 1.20.

Motion picture technology quickly improved, and by 1896, a room full of spectators in a movie theater screamed in terror as a short film of a train pulling into a station convinced them that the train was about to crash into them (Figure 1.21(a)). There was no audio track. Such a reaction seems ridiculous for anyone who has been to a modern movie theater. As audience expectations increased, so has the degree of realism produced by special effects. In 1902, viewers were inspired by a Journey to the Moon (Figure 1.21(b)), but by 2013, an extremely high degree of realism seemed necessary to keep viewers believing (Figure 1.21(c) and 1.21(d)).

At the same time, motion picture audiences have been willing to accept lower degrees of realism. One motivation, as for paintings, is to leave more to the imagination. The popularity of *animation* (also called *anime* or *cartoons*) is a prime example (Figure 1.22). Even within the realm of animations, a similar trend has emerged as with motion pictures in general. Starting from simple line drawings in 1908 with Fantasmagorie (Figure 1.22(a)), greater detail appears in 1928 with the introduction of Mickey Mouse(Figure 1.22(b)). By 2003, animated films achieved a much higher degree of realism (Figure 1.22(c)); however, excessively simple animations have also enjoyed widespread popularity (Figure 1.22(d)).

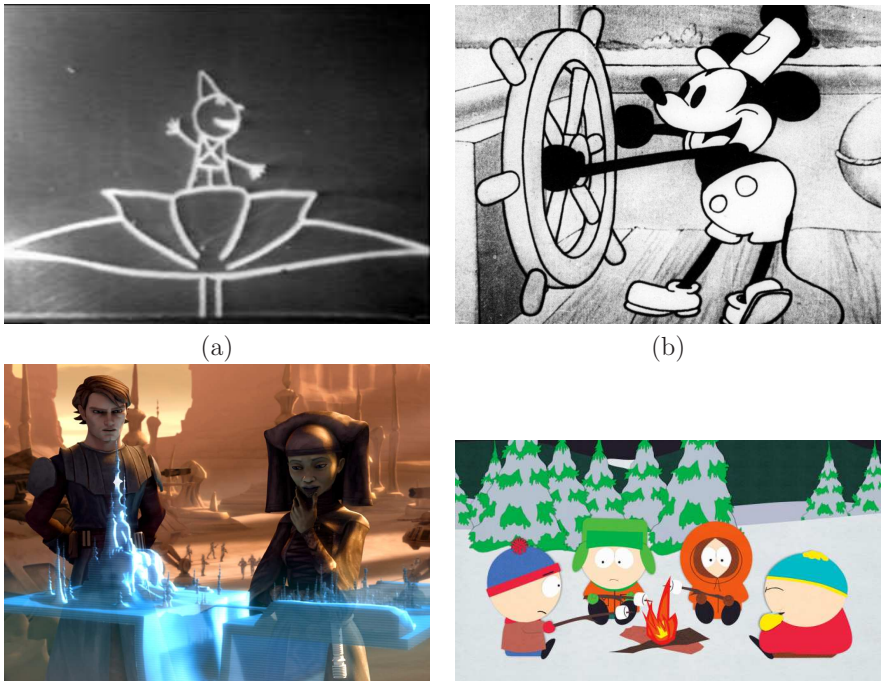


(a) (b) (c) (d)

Figure 1.21: A progression of special effects: (a) Arrival of a Train at La Ciotat Station, 1896. (b) A Trip to the Moon, 1902. (c) The movie 2001, from 1968. (d) Gravity, 2013.

Toward convenience and portability Another motivation for accepting lower levels of realism is cost and portability. As shown in Figure 1.23, families were willing to gather in front of a television to watch free broadcasts in their homes, even though they could go to theaters and watch high-resolution, color, panoramic, and 3D movies at the time. Such tiny, blurry, black-and-white television sets seem comically intolerable with respect to our current expectations. The next level of portability is to carry the system around with you. Thus, the progression is from: 1) having to go somewhere to watch it, to 2) being able to watch it in your home, to 3) being able to carry it anywhere. Whether pictures, movies, phones, computers, or video games, the same progression continues. We can therefore expect the same for VR systems. At the same time, note that the gap is closing between these levels: The quality we expect from a portable device is closer than ever before to the version that requires going somewhere to experience it.

Video games Motion pictures yield a passive, third-person experience, in contrast to video games which are closer to a first-person experience by allowing us to interact with him. Recall from Section 1.1 the differences between open-loop



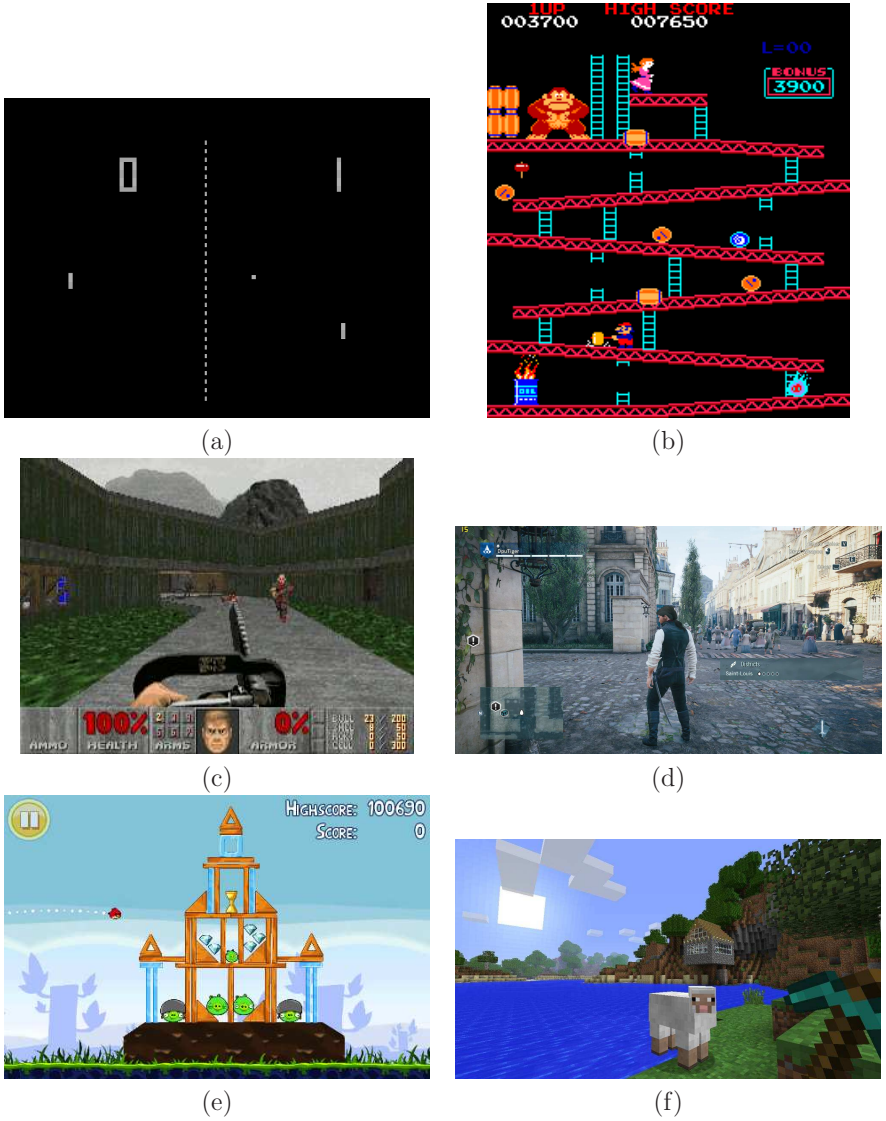
(a) (b) (c) (d)

Figure 1.22: A progression of cartoons: (a) Emile Cohl, Fantasmagorie, 1908. (b) Mickey Mouse in Steamboat Willie, 1928. (c) The Clone Wars Series, 2003. (d) South Park, 1997.

and closed-loop VR. Video games are an important step closer to closed-loop VR, whereas motion pictures are open-loop. As shown in Figure 1.24, we see the same trend from simplicity to improved realism and then back to simplicity. The earliest games, such as Pong and Donkey Kong, left much to the imagination. *First-person shooter (FPS)* games such as Doom gave the player a first-person perspective and launched a major campaign over the following decade toward higher quality graphics and realism. Assassin's Creed shows a typical scene from a modern, realistic video game. At the same time, wildly popular games have emerged by focusing on simplicity. Angry Birds looks reminiscent of games from the 1980s, and Minecraft allows users to create and inhabit worlds composed of course blocks. Note that reduced realism often leads to simpler engineering requirements; in 2015, an advanced FPS game might require a powerful PC and graphics card, while simpler games would run on a basic smartphone. Repeated lesson: Don't assume that more realistic is better!

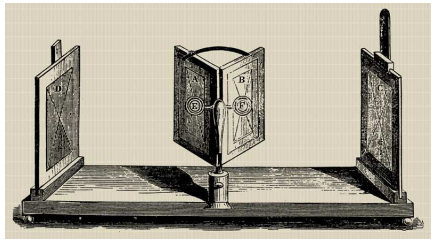
Figure 1.23: Although movie theaters with large screens were available, families were also content to gather around television sets that produced a viewing quality that would be unbearable by current standards, as shown in this photo from 1958.

Beyond staring at a rectangle The concepts so far are still closely centered on staring at a rectangle that is fixed on a wall. Two important steps come next: 1) Presenting a separate picture to each eye to induce a "3D" effect. 2) Increasing the field of view so that the user is not distracted by anything but the stimulus. One way our brains infer the distance of objects from our eyes is by *stereopsis*. Information is gained by observing and matching features in the world that are visible to both the left and right eyes. The differences between their images on the retina yield cues about distances; keep in mind that there are many more such cues, which we discuss in Section 6.1. The first experiment that showed this 3D effect of stereopsis was performed in 1838 by Charles Wheatstone in a system called the *stereoscope* (Figure 1.25(a)). By the 1930s, a portable version became a successful commercial product known to this day as the View-Master (Figure 1.25(b)). Pursuing this idea further led to Sensorama, which added motion pictures, sound, vibration, and even smells to the experience (Figure 1.25(c)). An unfortunate limitation of these designs is requiring that the viewpoint is fixed with respect to the picture. If the device is too large, then the user's head also becomes fixed in the world. An alternative has been available in movie theaters since the 1950s. Stereopsis is achieved when participants wore special glasses that select



(a) (b) (c) (d) (e) (f)

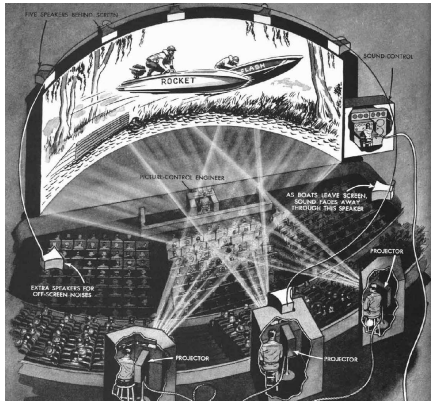
Figure 1.24: A progression of video games: (a) Atari's Pong, 1972. (b) Nintendo's Donkey Kong, 1981. (c) id Software's Doom, 1993. (d) Ubisoft's Assassin's Creed Unity, 2014. (e) Rovio Entertainment's Angry Birds, 2009. (f) Markus "Notch" Persson's Minecraft, 2011.



(a)

(b)

(c)



(d)

Figure 1.25: (a) The first stereoscope, developed by Wheatstone in 1838, used mirrors to present a different image to each eye; the mirrors were replaced by lenses soon afterward. (b) The View-Master is a mass-produced stereoscope that has been available since the 1930s. (c) In 1957, Sensorama added motion pictures, sound, vibration, and even smells to the experience. (d) In competition to stereoscopic viewing, Cinerama offered a larger field of view. Larger movie screens caused the popularity of 3D movies to wane in the 1950s.

a different image for each eye using polarized light filters. This popularized *3D movies*, which are viewed the same way in the theaters today.

Another way to increase the sense of immersion and depth is to increase the field of view. The Cinerama system from the 1950s offered a curved, wide field of view that is similar to the curved, large LED (Light-Emitting Diode) displays offered today (Figure 1.25(d)). Along these lines, we could place screens all around us. This idea led to one important family of VR systems called the CAVE, which was introduced in 1992 at the University of Illinois [30] (Figure 1.26(a)). The user enters a room in which video is projected onto several walls. The CAVE system also offers stereoscopic viewing by presenting different images to each eye using polarized light and special glasses. Often, head tracking is additionally performed to allow viewpoint-dependent video to appear on the walls.

VR headsets Once again, the trend toward portability appears. An important step for VR was taken in 1968 with the introduction of Ivan Sutherland's Sword of Damocles, which leveraged the power of modern displays and computers (Figure 1.26(b)). He constructed what is widely considered to be the first VR headset. As the user turns his head, the images presented on the screen are adjusted to compensate so that the virtual objects appear to be fixed in space. This yielded the first glimpse of an important concept in this book: The *perception of stationarity*. To make an object appear to be stationary while you move your sense organ, the device producing the stimulus must change its output to compensate for the motion. This requires sensors and tracking systems to become part of the VR system. Commercial VR headsets started appearing in the 1980s with Jaron Lanier's company VPL, thereby popularizing the image of *goggles and gloves*; Figure 1.26(c). In the 1990s, VR-based video games appeared in arcades (Figure 1.26(d)) and at home units (Figure 1.26(e). The experiences were not compelling or comfortable enough to attract mass interest. However, the current generation of VR headset leverages the widespread availability of high resolution screens and sensors, due to the smartphone industry, to offer lightweight, low-cost, high-field-of-view headsets, such as the Oculus Rift (Figure 1.26(f)). This has greatly improved the quality of VR experiences while significantly lowering the barrier of entry for developers and hobbyists. This also caused a recent flood of interest in VR technology and applications.

Bringing people together We have so far neglected an important aspect, which is human-to-human or social interaction. We use formats such as a live theater performance, a classroom, or a lecture hall for a few people to communicate with or entertain a large audience. We write and read novels to tell stories to each other. Prior to writing, skilled storytellers would propagate experiences to others, including future generations. We have communicated for centuries by writing letters to each other. More recent technologies have allowed us to interact directly without delay. The audio part has been transmitted through telephones for over a century, and now the video part is transmitted as well through videocon-

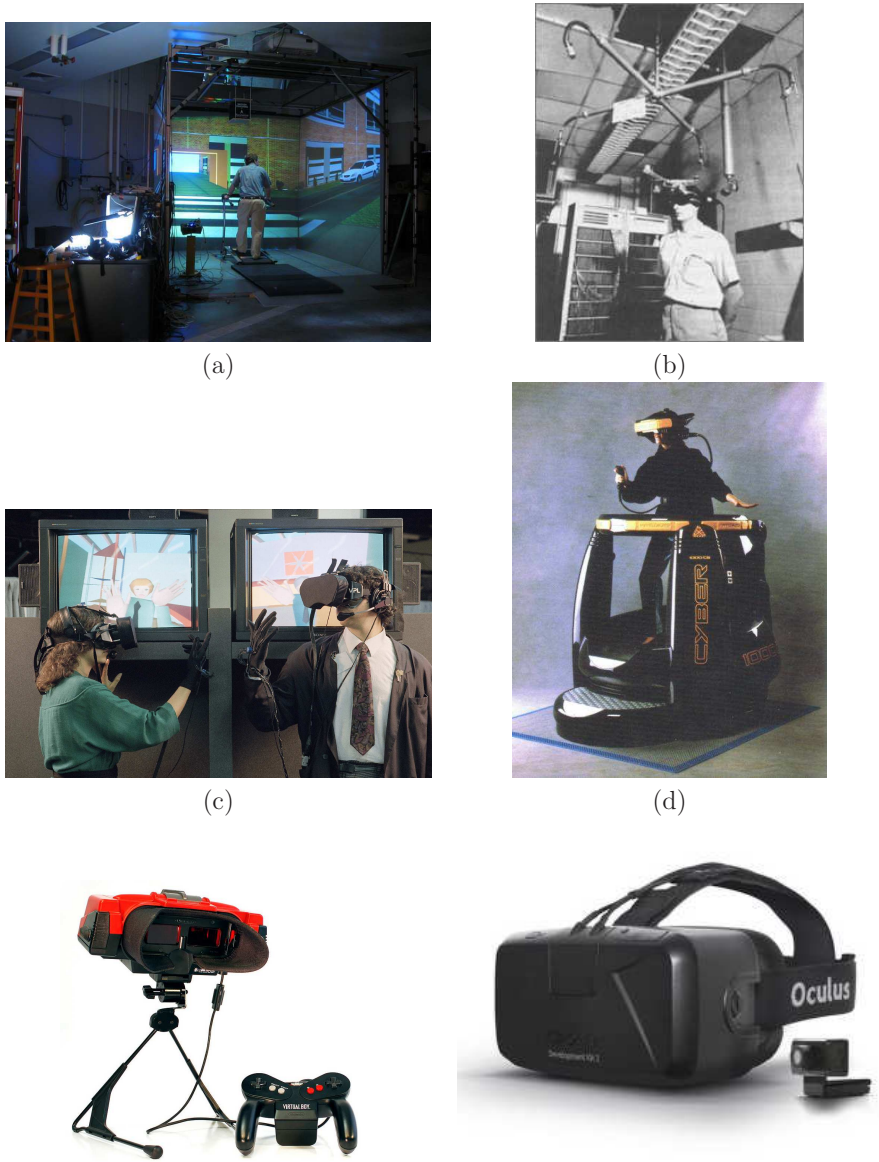


Figure 1.26: (a) CAVE VR, 1992. (b) Sword of Damocles, 1968. (c) VPL Eyephones, 1980s. (d) Virtuality gaming, 1990s. (e) Nintendo Virtual Boy, 1995. (f) Oculus Rift DK2, 2014.

Figure 1.27: Second Life was introduced in 2003 as a way for people to socialize through avatars and essentially build a virtual world to live in. Shown here is the author giving a keynote address at the 2014 Opensimulator Community Conference. The developers build open source software tools for constructing and hosting such communities of avatars with real people behind them.

ferencing over the Internet. At the same time, simple text messaging has become a valuable part of our interaction, providing yet another example of a preference for decreased realism. Communities of online users who interact through text messages over the Internet have been growing since the 1970s. In the context of games, early Multi-User Dungeons (MUDs) grew into Massively Multiplayer Online Games (MMORPGs) that we have today. In the context of education, the PLATO system from the University of Illinois was the first computer-assisted instruction sytsen, which included message boards, instant messaging, screen sharing, chat roms, and emoticons. This was a precursor to many many community-based, on-line learning systems, such as the Khan Academy and Coursera. The largest amount of online social interaction today occurs through Facebook apps, which involve direct communication through text along with the sharing of pictures, videos, and links.

Returning to VR, we can create *avatar* representations of ourselves and "live" together in virtual environments, as is the case with Second Life and Opensimulator 1.27. Without being limited to staring at rectangles, what kinds of societies will emerge with VR? Popular science fiction novels have painted a thrilling, yet dystopian future of a world where everyone prefers to interact through VR [28, 48, 141]. It remains to be seen what the future will bring.

As the technologies evolve over the years, keep in mind the power of simplicity when making a VR experience. In some cases, maximum realism may be important; however, leaving much to the imagination of the users is also valuable. Although the technology changes, one important invariant is that humans are still

designed the same way. Understanding how our senses, brains, and bodies work is crucial to understanding the fundamentals of VR systems.

Further reading

Aghajan, Z., Acharya, L., Moore, J., Cushman, J., Vuong, C., Mehta, M. (2014). Impaired spatial selectivity and intact phase precession in two-dimensional virtual reality Nature Neuroscience, 18 (1), 121-128

Nitz, D. (2014). A place for motion in mapping Nature Neuroscience

Chapter 2

Bird's-Eye View

Chapter Status	Taken from *Virtual Reality*, S. M. LaValle
	This online chapter is not the final version! Check **http://vr.cs.uiuc.edu/** for information on the latest draft version. This draft was compiled on June 29, 2016.

This chapter presents an overview of VR systems from hardware (Section 2.1 to software 2.2 to human perception 2.3. The purpose is to quickly provide a complete perspective so that the detailed subjects in the remaining chapters are understood within the larger context.

2.1 Hardware

The first step to understanding how VR works is to consider what constitutes the entire *VR system*. It is tempting to think of it as being merely the hardware components, such as computers, headsets, and controllers. This would be woefully incomplete. As shown in Figure 2.1, it is equally important to account for the organism, which in this chapter will exclusively refer to a human *user*. The hardware produces stimuli that override the senses of the user. In the Sword of Damocles from Section 1.3 (Figure 1.26(b)), recall that tracking is needed to adjust the stimulus based on human motions. The VR hardware accomplishes this by using its own sensors, thereby *tracking* motions of the user. Head tracking is the most important, but tracking also may include button presses, controller movements, eye movements, or the movements of any other body parts. Finally, it is also important to consider the surrounding physical world as part of the VR system. In spite of stimulation provided by the VR hardware, the user will always have other senses that respond to stimuli from the real world. She also has the ability to change her environment through body motions. The VR hardware might also track objects other than the user, especially if interaction with them is

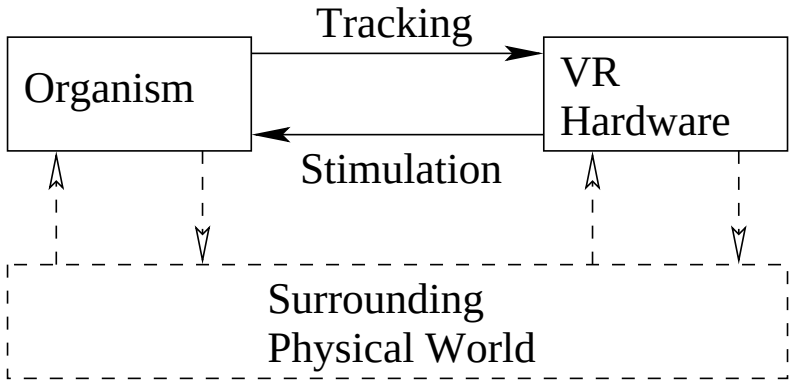


Figure 2.1: A third-person perspective of a VR system. It is wrong to assume that the engineered hardware and software are the complete VR system: The organism and its interaction with the hardware are equally important. Furthermore, interactions with the surrounding physical world continue to occur during a VR experience.

part of the VR experience. Through a robotic interface, the VR hardware might also change the real world. One example is teleoperation of a robot through a VR interface.

Sensors and sense organs How is information extracted from the physical world? Clearly this is crucial to a VR system. In engineering, a *transducer* refers to a device that converts energy from one form to another. A *sensor* is a special transducer that converts the energy it receives into a signal for an electrical circuit. This may be an analog or digital signal, depending on the circuit type. A sensor typically has a *receptor* that collects the energy for conversion. Organisms work in a similar way. The "sensor" is called a *sense organ*, with common examples being eyes and ears. Because our "circuits" are formed from interconnected neurons, the sense organs convert energy into *neural impulses*. As you progress through this book, keep in mind the similarities between engineered sensors and natural sense organs. They are measuring the same things and sometimes even function in a similar manner. This should not be surprising because we and our engineered devices share the same physical world: The laws of physics and chemistry remain the same.

Configuration space of sense organs As the user moves through the physical world, his sense organs move along with him. Furthermore, some sense organs move relative to the body skeleton, such as our eyes rotating within their sockets. Each sense organ has a *configuration space*, which corresponds to all possible ways it can be transformed or configured. The most important aspect of this is the

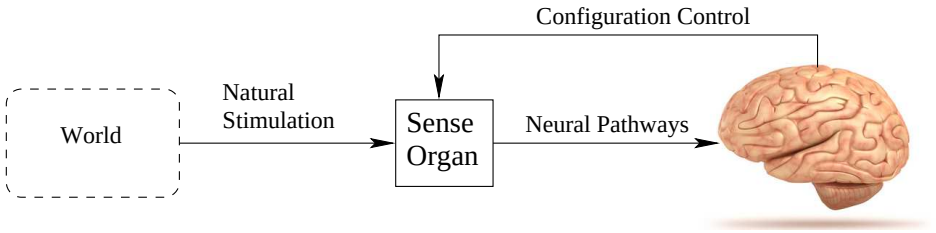


Figure 2.2: Under normal conditions, the brain (and body parts) control the configuration of sense organs (eyes, ears, fingertips) as they receive natural stimulation from the surrounding, physical world.

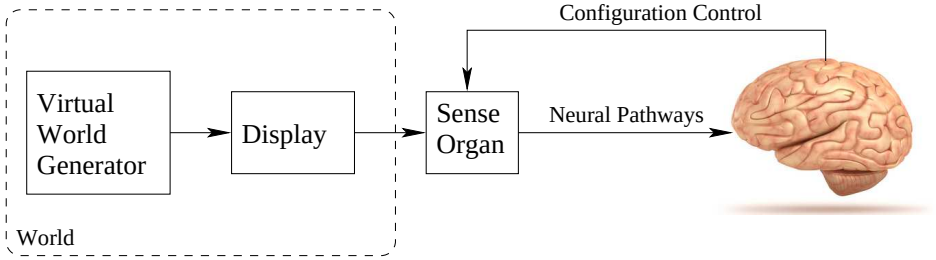


Figure 2.3: In comparison to Figure 2.2, a VR system "hijacks" each sense by replacing the natural stimulation with artificial stimulation that is provided by hardware called a display. Using a computer, a virtual world generator maintains a coherent, virtual world. Appropriate "views" of this virtual world are rendered to the display.

number of *degrees of freedom* or *DOFs* of the sense organ. Chapter 3 will cover this thoroughly, but for now note that a rigid object that moves through ordinary space has six DOFs. Three DOFs correspond to its changing position in space: 1) side-to-side motion, 2) vertical motion, and 3) closer-further motion. The other three DOFs correspond to possible ways the object could be rotated; in other words, exactly three independent parameters are needed to specify how the object is oriented. These are called yaw, pitch, and roll, and are covered in Section 3.2.

As an example, consider your left ear. As you rotate your head or move your body through space, the position of the ear changes, as well as its orientation. This yields six DOFs. The same is true for your right eye. Keep in mind that our bodies have many more degrees of freedom, which affect the configuration of our sense organs. A tracking system may be necessary to determine the position and orientation of each sense organ that receives artificial stimuli, which will be explained shortly.

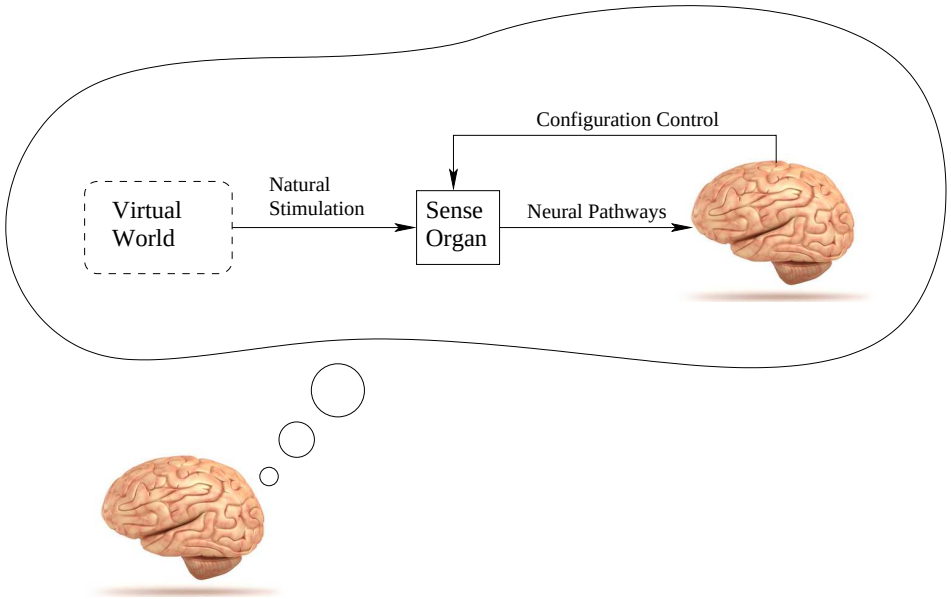


Figure 2.4: If done well, the brain is "fooled" into believing that the virtual world is in fact the surrounding physical world and natural stimulation is resulting from it.

An abstract view Figure 2.2 illustrates the normal operation of one of our sense organs without interference from VR hardware. The brain controls its configuration, while the sense organ converts natural stimulation from the environment into neural impulses that are sent to the brain. Figure 2.3 shows how it appears in a VR system. The VR hardware contains several components that will be discussed shortly. A *Virtual World Generator (VWG)* runs on a computer and produces "another world", which could be many possibilities, such as a pure simulation of a synthetic world, a recording of the real world, or a live connection to another part of the real world. The human perceives the virtual world through each targeted sense organ using a *display*, which emits energy that is specifically designed to mimic the type of stimulus that would appear without VR. The process of converting information from the VWG into output for the display is called *rendering*. In the case of human eyes, the display might be a smartphone screen or the screen of a video projector. In the case of ears, the display is referred to as a *speaker*. (A display need not be visual, even though this is the common usage in everyday life.) If the VR system is effective, then the brain is hopefully "fooled" in the sense shown in Figure 2.4. The user should believe that the stimulation of the senses is natural and comes from a plausible world, being consistent with at least some past experiences.

Aural: world-fixed vs. user-fixed Recall from Section 1.3 the trend of having to go somewhere for an experience, to having it in the home, and then finally to having it be completely portable. To understand these choices for VR systems and their implications on technology, it will be helpful to compare a simpler case: Audio or *aural* systems.

Figure 2.5 shows the speaker setup and listener location for a Dolby 7.1 Surround Sound theater system, which could be installed at a theater or a home family room. Seven speakers distributed around the room periphery generate most of the sound, while a subwoofer (the "1" of the "7.1") delivers the lowest frequency components. The aural displays are therefore *world-fixed*. Compare this to a listener wearing headphones, as shown in Figure 2.6. In this case, the aural displays are *user-fixed*. Hopefully, you have already experienced settings similar to these many times.

What are the key differences? In addition to the obvious portability of headphones the following quickly come to mind:

- In the surround-sound system, the generated sound (or stimulus) is far away from the ears, whereas it is quite close for the headphones.
- One implication of the difference in distance is that much less power is needed for the headphones to generate an equivalent perceived loudness level compared with distant speakers.
- Another implication based on distance is the degree of privacy allowed by the wearer of headphones. A surround-sound system at high volume levels could generate a visit by angry neighbors.

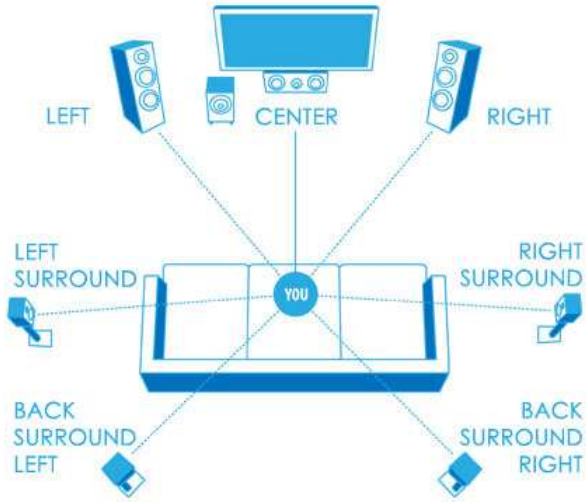


Figure 2.5: In a surround-sound system, the aural displays (speakers) are world-fixed while the user listens from the center.

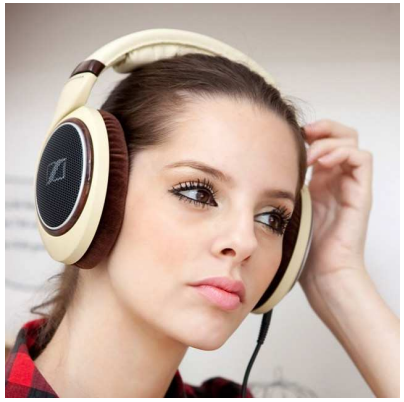


Figure 2.6: Using headphones, the displays are user-fixed, unlike the case of a surround-sound system.

- Wearing electronics on your head could be uncomfortable over long periods of time, causing a preference for surround sound over headphones.

- Several people can enjoy the same experience in a surround-sound system (although they cannot all sit in the optimal location). Using headphones, they would need to split the audio source across their individual headphones simultaneously.

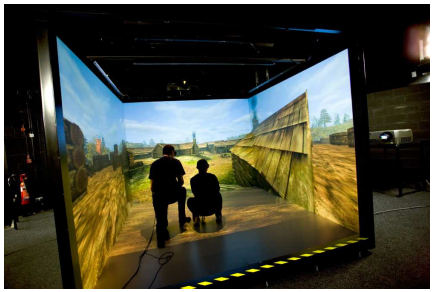
- They are likely to have different costs, depending on the manufacturing costs and available component technology. At present, headphones are favored by costing much less than a set of surround-sound speakers (although one can spend a large amount of money on either).

All of these differences carry over to VR systems. This should not be too surprising because we could easily consider a pure audio experience to be a special kind of VR experience based on our definition from Section 1.1.

While listening to music, close your eyes and imagine you are at a live performance with the artists surrounding you. Where do you perceive the artists and their instruments to be located? Are they surrounding you, or do they seem to be in the middle of your head? Using headphones, it is most likely that they seem to be inside your head. In a surround-sound system, if recorded and displayed properly, the sounds should seem to be coming from their original locations well outside of your head. They probably seem constrained, however, into the horizontal plane that you are sitting in.

This shortcoming of headphones is not widely recognized at present, but nevertheless represents a problem that becomes much larger for VR systems that include visual displays. If you want to preserve your perception of where sounds are coming from, then headphones would need to take into account the configuration of your ears in space so that audio is adjusted accordingly. For example, if you nod your head back and forth in a "no" gesture, then the sound being presented to each ear needs to be adjusted so that the simulated sound source is rotated in the opposite direction. In the surround-sound system, the speaker does not follow your head and therefore does not need to rotate. If the speaker rotates with your head, then a counter-rotation is needed to "undo" your head rotation so that the sound source location is perceived to be stationary.

Visual: world-fixed vs. user-fixed Now consider adding a visual display. You might not worry much about the perceived location of artists and instruments while listening to music, but you will quickly notice if their locations do not appear correct to your eyes. Our vision sense is much more powerful and complex than our sense of hearing. Figure 2.7(a) shows a CAVE system, which parallels the surround-sound system in many ways. The user again sits in the center while displays around the periphery present visual stimuli to your eyes. The speakers are replaced by video screens. Figure 2.7(b) shows a user wearing a VR headset, which parallels the headphones.



(a)

(b)

Figure 2.7: (a) A CAVE VR system developed at Teesside University, UK. (b) A woman wearing the first-generation Oculus Rift headset.

Suppose the screen in front of the user's eyes shows a fixed image in the headset. If the user rotates his head, then the image will be perceived as being attached to the head. This would occur, for example, if you rotate your head while using the Viewmaster (recall Figure 1.25(b)). If you would like to instead perceive the image as part of a fixed world around you, then the image inside the headset must change to compensate as you rotate your head. The surrounding virtual world should be counter-rotated, the meaning of which should become more precise after reading Section 3.4. Once we agree that such transformations are necessary, it becomes a significant engineering challenge to estimate the amount of head and eye movement that has occurred and apply the appropriate transformation in a timely and accurate manner. If this is not handled well, then the user could have a poor or unconvincing experience. Worse yet, they could fall prey to VR sickness. This is one of the main reasons why the popularity of VR headsets waned in the 1990s. The component technology was not good enough yet. Fortunately, the situation is much improved at present. For audio, few seemed to bother with this transformation, but for the visual counterpart, it is absolutely critical. One final note is that tracking and applying transformations also becomes necessary in CAVE systems if we want the images on the screens to be altered according to changes in the eye *positions* inside of the room.

Now that you have a high-level understanding of the common hardware arrangements, we will take a closer look at hardware components that are widely available for constructing VR systems. These are expected to change quickly, with costs decreasing and performance improving. We also expect many new devices to appear in the marketplace in the coming years. In spite of this, the fundamentals in this book remain unchanged. Knowledge of the current technology provides concrete examples to make the fundamental VR concepts clearer.

The hardware components of VR systems are conveniently classified as:

- **Displays (output):** Devices that stimulate a sense organ.

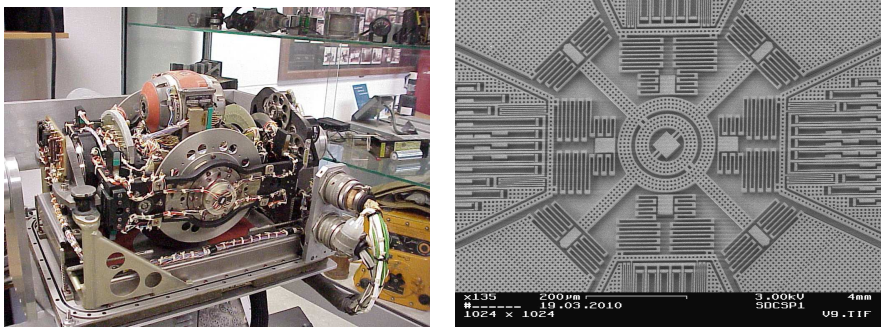
(a) (b)

Figure 2.8: Two examples of haptic feedback devices. (a) The Geomagic Phantom allows the user to feel strong resistance when poking into a virtual object with a real stylus. A robot arm provides the appropriate forces. (b) Some game controllers occasionally vibrate.

- **Sensors (input):** Devices that extract information from the real world.
- **Computers:** Devices that process inputs and outputs sequentially.

Displays The purpose of a display is to generate a stimulus for a targeted sense organ. Vision is our dominant sense, and any display constructed for the eye must cause the desired image to be formed on the retina. Because of this importance, Chapters 4 and 5 will explain optical systems and the human vision system, respectively. For CAVE systems, some combination of digital projectors and mirrors are used. Due to the plummeting costs, an array of large-panel displays may alternatively be employed. For headsets, a smartphone display can be placed close to the eyes and brought into focus using one magnifying lens for each eye. Screen manufacturers are currently making custom displays for VR headsets by leveraging the latest LED display technology from the smartphone industry. Some are targeting one display per eye with frame rates above 90Hz and over two megapixels per eye. Reasons for this are explained in Chapter 5. More exotic displays, which are primarily in a research-and-development stage include pico projectors [**?**, **?**], light-field displays [75, 89], and multi-focal-plane optical systems [2].

Now imagine displays for other sense organs. Sound is displayed to the ears using classic speaker technology. Bone conduction methods may also be used, which vibrate the skull and propagate the waves to the inner ear; this method appeared recently in Google Glass []. Chapter 11 covers the auditory part of VR in detail. For the sense of touch, we have *haptic displays*. Two examples are pictured in Figure 2.8. Haptic feedback can be given in the form of vibration, pressure, or temperature. Displays have even been developed for providing smell and taste in a VR experience [].

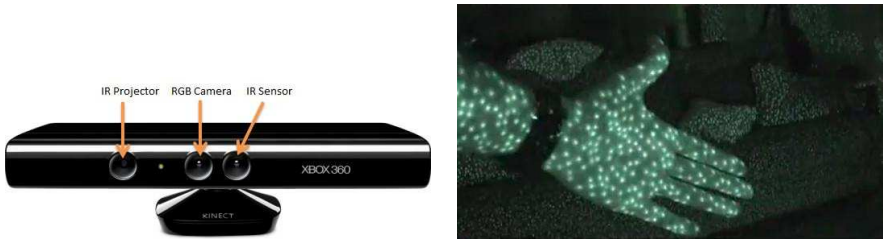


(a) (b)

Figure 2.9: Inertial measurement units (IMUs) have gone from large, heavy mechanical systems to cheap, microscopic MEMS circuits. (a) The LN-3 Inertial Navigation System, developed in the 1960s by Litton Industries. (b) The internal structures of a MEMS gyroscope, for which the total width is less than 1mm.

Sensors Consider the input side of the VR hardware. A brief overview is given here, until Chapter 9 covers sensors and tracking systems in detail. For visual and auditory body-mounted displays, the position and orientation of the sense organ must be tracked by sensors to appropriately adapt the stimulus. The orientation part is usually accomplished by an *inertial measurement unit* or *IMU*. The main component is a *gyroscope*, which measures its own rate of rotation; the rate is referred to as *angular velocity* and has three components. Measurements from the gyroscope are integrated over time to obtain an estimate of the cumulative change in orientation. The resulting error, called *drift error*, would gradually grow unless other sensors are used. To reduce drift error, IMUs also contain an *accelerometer* and possibly a *magnetometer*. Over the years, IMUs have gone from existing only as large mechanical systems in aircraft and missiles to being tiny devices inside of our smartphones; see Figure 2.9. Due to their small size, weight, and cost, IMUs can be easily embedded in wearable devices. They are one of the most important enabling technologies for the current generation of VR headsets and are mainly used for tracking the user's head orientation.

Digital cameras provide another critical source of information for tracking systems. Like IMUs, they have become increasingly cheap and portable due to the smartphone industry, while at the same time improving in image quality. Cameras enable tracking approaches that exploit *line-of-sight visibility*. The idea is to place markers on the object to be tracked and find them in the image. Such *visibility constraints* severely limit the possible object positions and orientations. Standard cameras passively form an image focusing the light through an optical system, much like the human eye. Once the camera calibration parameters are known, an observed marker is known to lie along a ray in space. Cameras are commonly



(a) (b)

Figure 2.10: (a) The Microsoft Kinect sensor gathers both an ordinary RGB image and a depth map (the distance away from the sensor for each pixel). (b) The depth is determined by observing the locations of projected IR dots in an image obtained from an IR camera.

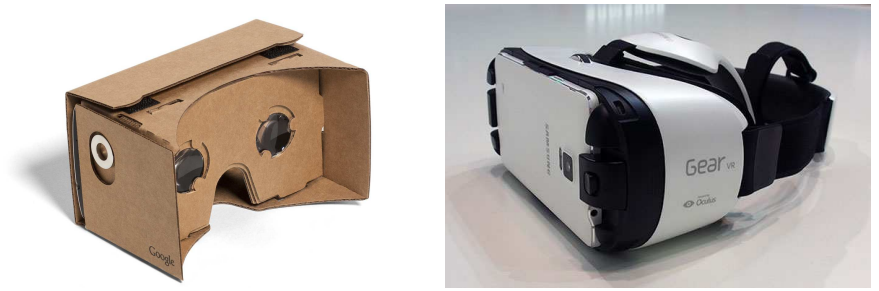
used to track eyes, heads, hands, entire human bodies, and any other objects in the physical world. One of the main difficulties at present is to obtain reliable and accurate performance without placing special markers on the user or objects around the scene.

As opposed to standard cameras, *depth cameras* work actively by projecting light into the scene and then observing its reflection in the image. This is typically done in the infrared (IR) spectrum so that humans do not notice; see Figure 2.10.

In addition to these sensors, we rely heavily on good-old mechanical switches and potientiometers to create keyboards and game controllers. An optical mouse is also commonly used. One advantage of these familiar devices is that users can rapidly input data or control their characters by leveraging their existing training. A disadvantage is that they might be hard to find or interact with if their faces are covered by a headset.

Computers A computer executes the virtual world generator (VWG). Where should this computer be? Although unimportant for world-fixed displays, the location is crucial for body-fixed displays. If a separate PC is needed to power the system, then fast, reliable communication must be provided between the headset and the PC. This connection is currently made by wires, leading to an awkward tether; current wireless speeds are not sufficient. As you have noticed, most of the needed sensors exist on a smartphone, as well as a moderately powerful computer. Therefore, a smartphone can be dropped into a case with lenses to provide a VR experience with little added costs (Figure 2.11). The limitation, though, is that the VWG must be simpler than in the case of a separate PC so that it runs on less-powerful computing hardware.

In addition to the main computing systems, specialized computing hardware may be utilized. Graphical processing units (GPUs) have been optimized for quickly rendering graphics to a screen and they are currently being adapted to



(a) (b)

Figure 2.11: Two headsets that create a VR experience by dropping a smartphone into a case. (a) Google Cardboard works with a wide variety of smartphones. (b) Samsung Gear VR is optimized for one particular smartphone (in this case, the Samsung S6).

handle the specific performance demands of VR. Also, a display interface chip converts an input video into display commands. Finally, microcontrollers are frequently used to gather information from sensing devices and send them to the main computer using standard protocols, such as USB.

To conclude with hardware, Figure 2.12 shows the hardware components for the Oculus Rift DK2, which became available in late 2014. In the lower left corner, you can see a smartphone screen that serves as the display. Above that is a circuit board that contains the IMU, display interface chip, a USB driver chip, a set of chips for driving LEDs on the headset for tracking, and a programmable microcontroller. The lenses, shown in the lower right, are placed so that the smartphone screen appears to be "infinitely far" away, but nevertheless fills most of the field of view of the user. The upper right shows flexible circuits that deliver power to IR LEDs embedded in the headset (they are hidden behind IR-transparent plastic). A camera is used for tracking, and its parts are shown in the center.

2.2 Software

From a developer's standpoint, it would be ideal to program the VR system by providing high-level descriptions and having the software determine automatically all of the low-level details. In a perfect world, there would be a *VR engine*, which serves a purpose similar to the game engines available today for creating video games. If the developer follows patterns that many before him have implemented already, then many complicated details can be avoided by simply calling functions from a well-designed software library. However, if the developer wants to try something relatively original, then she would have to design the functions from scratch. This requires a deeper understanding of the VR fundamentals, while also

Figure 2.12: Disassembly of the Oculus Rift DK2 headset (image by ifixit).

being familiar with lower-level system operations.

Unfortunately, we are currently a long way from having fully functional, general-purpose VR engines. As applications of VR broaden, specialized VR engines are also likely to emerge. For example, one might be targeted for immersive cinematography while another is geared toward engineering design. Which components will become more like part of a VR "operating system" and which will become higher level "engine" components? Given the current situation, developers will likely be implementing much of the functionality of their VR systems from scratch. This may involve utilizing a *software development kit (SDK)* for particular headsets that handles the lowest level operations, such as device drivers, head tracking, and display output. Alternatively, they might find themselves using a game engine that has been recently adapted for VR, even though it was fundamentally designed for video games on a screen. This can avoid substantial effort at first, but then may be cumbersome when someone wants to implement ideas that are not part of standard video games.

What software components are needed to produce a VR experience? Figure 2.13 presents a high-level view that highlights the central role of the Virtual World Generator (VWG). The VWG receives inputs from low-level systems that indicate what the user is doing in the real world. A head tracker provides timely estimates of the user's head position and orientation. Keyboard, mouse, and game controller events arrive in a queue that are ready to be processed. The key role of the VWG is to maintain enough of an internal "reality" so that renderers can extract the

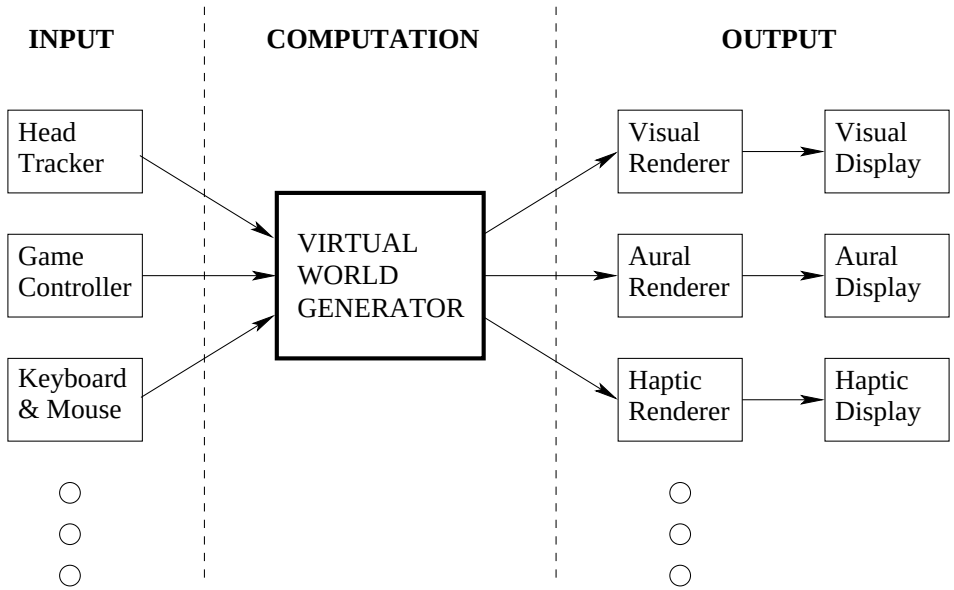


Figure 2.13: The Virtual World Generator (VWG) maintains another world, which could be synthetic, real, or some combination. From a computational perspective, the inputs are received from the user and his surroundings, and appropriate views of the world are rendered to displays.

information they need to calculate outputs for their displays.

Virtual world: real vs. synthetic At one extreme, the virtual world could be completely synthetic. In this case, numerous triangles are defined in a 3D space, along with material properties that indicate how they interact with light, sound, forces, and so on. The field of *computer graphics* addresses computer-generated images from synthetic models [], and it remains important for VR; see Chapter 7. At the other extreme, the virtual world might be a recorded physical world that was captured using modern cameras, computer vision, and Simultaneous Localization and Mapping (SLAM) techniques; Figure 2.14. Many possibilities exist between the extremes. For example, camera images may be taken of a real object, and then mapped onto a synthetic object in the virtual world. This is called *texture mapping*, a common operation in computer graphics; see Section 7.2.

Matched motion The most basic operation of the VWG is to maintain a correspondence between user motions in the real world and the virtual world; see Figure 2.15. In the real world, the user's motions are confined to a safe region, which we will call the *matched zone*. Imagine the matched zone as a place where the real and virtual worlds perfectly align. One of the greatest challenges is the mismatch

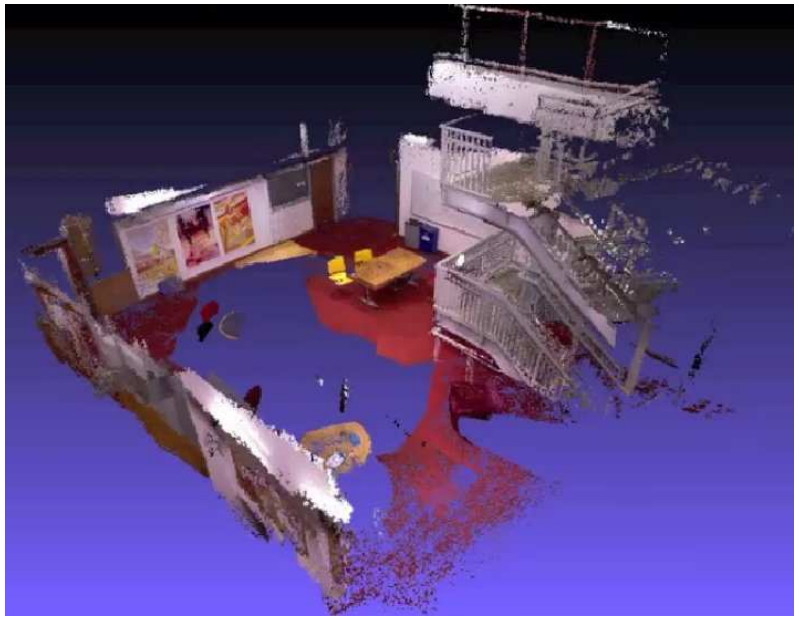


Figure 2.14: Using both color and depth information from cameras, a 3D model of the world can be extracted automatically using Simultaneous Localization and Mapping (SLAM) techniques. Figure from [63].

of obstacles: What if the user is blocked in the virtual world but not in the real world? The reverse is also possible. In a seated experience, the user sits in a chair while wearing a headset. The matched zone in this case is a small region, such as one cubic meter, in which users can move their heads. Head motions should be matched between the two worlds. If the user is not constrained to a seat, then the matched zone could be an entire room or an outdoor field. Note that safety becomes an issue because the user might spill a drink, hit walls, or fall into pits that exist only in the real world, but are not visible in the virtual world. Larger matched zones tend to lead to greater safety issues. Users must make sure that the matched zone is cleared of dangers in the real world, or the developer should make them visible in the virtual world.

Which motions from the real world should be reflected in the virtual world? This varies among VR experiences. In a VR headset that displays images to the eyes, head motions must be matched so that the visual renderer uses the correct viewpoint in the virtual world. Other parts of the body are less critical, but may become important if the user needs to perform hand-eye coordination or looks at other parts of her body and expects them to move naturally.

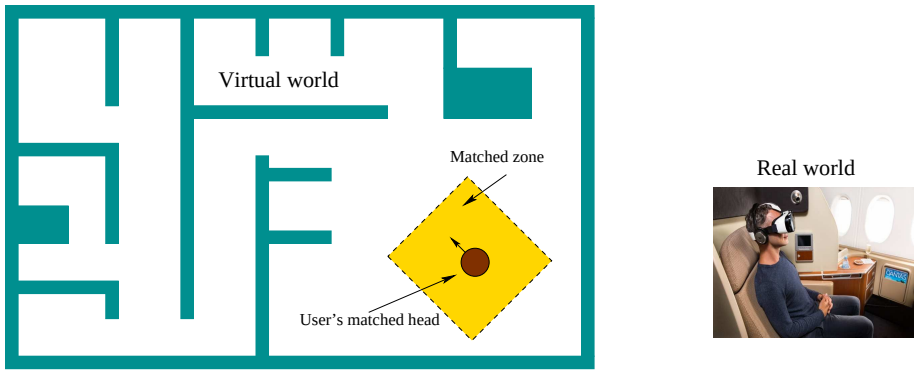


Figure 2.15: A matched zone is maintained between the user in their real world and his representation in the virtual world. The matched zone could be moved in the virtual world by using an interface, such as a game controller, while the user does not correspondingly move in the real world.

Locomotion In many VR experiences, users want to move well outside of the matched zone. This motivates *locomotion*, which means moving oneself in the virtual world, while this motion is not matched in the real world. Imagine you want to explore a virtual city while remaining seated in the real world. How should this be achieved? You could pull up a map and point to where you want to go, with a quick teleportation operation sending you to the destination. A popular option is to move oneself in the virtual world by operating a game controller, mouse, or keyboard. By pressing buttons or moving knobs, your self in the virtual world could be walking, running, jumping, swimming, flying, and so on. You could also climb aboard a vehicle in the virtual world and operate its controls to move yourself. These operations are certainly convenient, but often lead to sickness because of a mismatch between your balance and visual senses. See Sections 2.3 and 10.2.

Physics The VWG handles the *geometric* aspects of motion by applying the appropriate mathematical transformations. In addition, the VWG usually implements some *physics* so that as time progresses, the virtual world behaves like the real world. In most cases, the basic laws of mechanics should govern how objects move in the virtual world. For example, if you drop an object, it should accelerate to the ground due to gravitational force acting on it. One important component is a *collision detection algorithm*, which determines whether two or more bodies are intersecting in the virtual world. If a new collision occurs, then an appropriate response is needed. For example, suppose the user pokes his head through a wall in the virtual world. Should the head in the virtual world be stopped, even though it continues to move in the real world? To make it more complex, what should happen if you unload a dump truck full of basketballs into a busy street in the

virtual world? Simulated physics can become quite challenging, and is a discipline in itself. There is no limit to the complexity.

In addition to handling the motions of moving objects, the physics must also take into account how potential stimuli for the displays are created and propagate through the virtual world. How does light propagate through the environment? How does light interact with the surfaces in the virtual world? What are the sources of light? How do sound and smells propagate? These correspond to rendering problems, which are covered in Chapters 7 and 11 for visual and audio cases.

Networked experiences In the case of a networked VR experience, a shared virtual world is maintained by a server. Each user has a distinct matched zone. Their matched zones might overlap in a real world, but one must then be careful so that they avoid unwanted collisions. Most often, these zones are disjoint and distributed around the Earth. Within the virtual world, user interactions, including collisions, must be managed by the VWG. If multiple users are interacting in a social setting, then the burdens of matched motions may increase. As users see each other, they could expect to see eye motions, facial expressions, and body language; see Section 10.4.

Developer choices for VWGs To summarize, a developer could start with a basic Software Development Kit (SDK) from a VR headset vendor and then build her own VWG from scratch. The SDK should provide the basic drivers and an interface to access tracking data and make calls to the graphical rendering libraries. In this case, the developer must build the physics of the virtual world from scratch, handling problems such as avatar movement, collision detection, lighting models, and audio. This gives the developer the greatest amount of control and ability to optimize performance; however, it may come in exchange for a difficult implementation burden. In some special cases, it might not be too difficult. For example, in the case of the Google Street viewer (recall Figure 1.8), the "physics" is simple: The viewing location needs to jump between panoramic images in a comfortable way while maintaining a sense of location on the Earth. In the case of telepresence using a robot, the VWG would have to take into account movements in the physical world. Failure to handle collision detection could result in a broken robot (or human!).

At the other extreme, a developer may use a ready-made VWG that is customized to make a particular VR experience by choosing menu options and writing high-level scripts. Examples available today are OpenSimulator, Vizard by WorldViz, Unity 3D, and Unreal Engine by Epic Games. The latter two are game engines that were adapted to work for VR, and are by far the most popular among current VR developers. The first one, OpenSimulator, was designed as an open-source alternative to Second Life for building a virtual society of avatars. Using such higher-level engines make it easy for developers to make a VR experience in little time; however, the drawback is that it is harder to make unusual experiences that

were not imagined by the engine builders.

2.3 Human Physiology and Perception

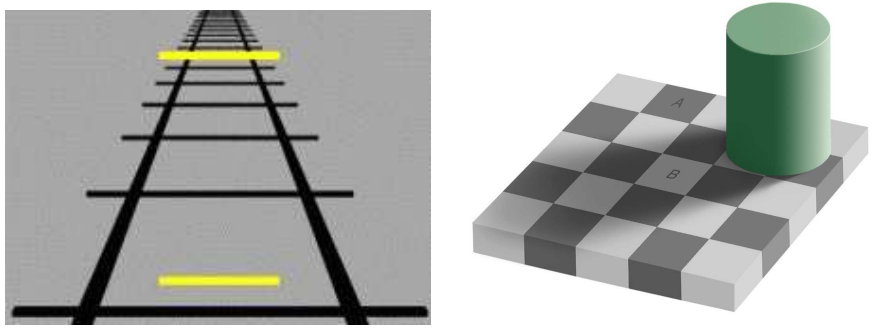
Our bodies were not designed for VR. By applying artificial stimulation to the senses, we are disrupting the operation of biological mechanisms that have taken hundreds of millions of years to evolve in a natural environment. We are also providing input to the brain that is not exactly consistent with all of our other life experiences. In some instances, our bodies may adapt to the new stimuli. This could cause us to become unaware of flaws in the VR system. In other cases, we might develop heightened awareness or the ability to interpret 3D scenes that were once difficult or ambiguous. Unfortunately, there are also many cases where our bodies react by increased fatigue or headaches, partly because the brain is working harder than usual to interpret the stimuli. Finally, the worst case is the onset of VR sickness, which typically involves symptoms of dizziness and nausea.

Perceptual psychology is the science of understanding how the brain converts sensory stimulation into perceived phenomena. Here are some typical questions that arise in VR and fall under this umbrella:

- How far away does that object appear to be?
- How much video resolution is needed to avoid seeing pixels?
- How many frames per second are enough to perceive motion as continuous?
- Is the user's head appearing at the proper height in the virtual world?
- Where is that virtual sound coming from?
- Why am I feeling nauseated?
- Why is one experience more tiring than another?
- What is presence?

To answer these questions and more, we must understand several things: 1) basic physiology of the human body, including sense organs and neural pathways, 2) the key theories and insights of experimental perceptual psychology, and 3) the interference of the engineered VR system with our common perceptual processes and the resulting implications or side-effects.

The perceptual side of VR often attracts far too little attention among developers. In the real world, perceptual processes are mostly invisible to us. Think about how much effort it requires to recognize a family member. When you see someone you know well, the process starts automatically, finishes immediately, and seems to require no effort. Scientists have conducted experiments that reveal how much work actually occurs in this and other perceptual processes. Through brain lesion studies, they are able to see the effects when a small part of the brain



(a) (b)

Figure 2.16: Optical illusions present an unusual stimulus that highlights limitations of our vision system. (a) The *Ponzo illusion* causes the upper line segment to appear larger than the lower one, even though they are the same length. (b) The *checker shadow illusion* causes the B tile to appear lighter than the A tile, even though they are the exactly the same shade of gray (figure by Adrian Pingstone).

is not functioning correctly. Some people suffer from prosopagnosia, which makes them unable to recognize the faces of familiar people, including themselves in a mirror, even though nearly everything else functions normally. Scientists are also able to perform *single-unit recordings*, mostly on animals, which reveal the firings of a single neuron in response to sensory stimuli. Imagine, for example, a single neuron that fires whenever you see a sphere.

Optical illusions One of the most popular ways to appreciate the complexity of our perceptual processing is to view optical illusions. These yield surprising results and are completely unobtrusive. Each one is designed to reveal some shortcoming of our visual system by providing a stimulus that is not quite consistent with ordinary stimuli in our everyday lives. Figure 2.16 shows two. These should motivate you to appreciate the amount of work that our sense organs and neural structures are doing to fill in missing details and make interpretations based on the context of our life experiences and existing biological structures. Interfering with these without understanding them is not wise!

Classification of senses Perception and illusions are not limited to our eyes. Figure 2.17 shows a classification of our basic senses. Recall that a sensor converts an energy source into signals in a circuit. In the case of our bodies, this means that a stimulus is converted into neural impulses. For each sense, Figure 2.17 indicates the type of energy for the stimulus and the *receptor* that converts the stimulus into neural impulses. Think of each receptor as a sensor that targets a particular kind of stimulus. This is referred to as *sensory system selectivity*. In each

Sense	Stimulus	Receptor	Sense Organ
Vision	Electromagnetic energy	Photoreceptors	Eye
Auditory	Air pressure waves	Mechanoreceptors	Ear
Touch	Tissue distortion	Mechanoreceptors	Skin, muscles
		Thermoreceptors	Skin
Balance	Gravity, acceleration	Mechanoreceptors	Vestibular organs
Taste/smell	Chemical composition	Chemoreceptors	Mouth, nose

Figure 2.17: A classification of the human body senses.

eye, over 100 million photoreceptors target electromagnetic energy precisely in the frequency range of visible light. Different kinds even target various colors and light levels; see Section 5.1. The auditory, touch, and balance senses involve motion, vibration, or gravitational force; these are sensed by mechanoreceptors. The sense of touch additionally involves thermoreceptors to detect change in temperature. Our *balance sense* helps us to know which way our head is oriented, including sensing the direction of "up". Finally, our sense of taste and smell is grouped into one category that relies on chemoreceptors, which provide signals based on chemical composition of matter appearing on our tongue or in our nasal passages.

Note that senses have engineering equivalents, most of which appear in VR systems. Imagine you a designing a humanoid telepresence robot, which you expect to interface with through a VR headset. You could then experience life through your surrogate robotic self. Digital cameras would serve as its eyes, and microphones would be the ears. Pressure sensors and thermometers could be installed to give a sense of touch. For balance, we can install an IMU. In fact, the human *vestibular organs* and modern IMUs bear a striking resemblance in terms of the signals they produce; see Section 8.2. We could even install chemical sensors, such as a pH meter, to measure aspects of chemical composition to provide taste and smell.

Big brains Perception happens after the sense organs convert the stimuli into neural impulses. According to latest estimates [10], human bodies contain around 86 billion neurons. Around 20 billion are devoted to the part of the brain called the *cerebral cortex*, which handles perception and many other high-level functions such as attention, memory, language, and consciousness. It is a large sheet of neurons around three millimeters thick and is heavily folded so that it fits into our skulls. In case you are wondering where we lie among other animals, a roundworm, fruit fly, and rat have 302, 100 thousand, and 200 million neurons, respectively. An elephant has over 250 billion neurons, which is more than us!

Only mammals have a cerebral cortex. The cerebral cortex of a rat has around 20 million neurons. Cats and dogs are at 300 and 160 million, respectively. A gorilla has around 4 billion. A type of dolphin called the long-finned pilot whale has an estimated 37 billion neurons in its cerebral cortex, making it roughly twice as many as in the human cerebral cortex; however, scientists claim this does not

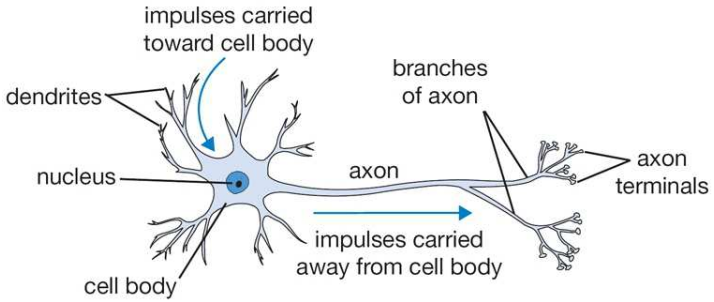


Figure 2.18: A typical neuron receives signals through dendrites, which interface to other neurons. It outputs a signal to other neurons through axons.

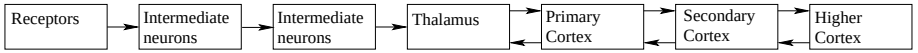


Figure 2.19: The stimulus captured by receptors works its way through a hierarchical network of neurons. In the early stages, signals are combined from multiple receptors and propagated upward. At later stages, information flows bidirectionally.

imply superior cognitive abilities [103].

Another important factor in perception and overall cognitive ability is the interconnection between neurons. Imagine an enormous directed graph, with the usual nodes and directed edges. The nucleus or cell body of each neuron is a node that does some kind of "processing". Figure 2.18 shows a neuron. The dendrites are essentially input edges to the neuron, whereas the axons are output edges. Through a network of dendrites, the neuron can aggregate information from numerous other neurons, which themselves may have aggregated information from others. The result is sent to one or more neurons through the axon. For a connected axon-dendrite pair, communication occurs in a gap called the *synapse*, where electrical or chemical signals are passed along. Each neuron in the human brain has on average about 7000 synaptic connections to other neurons, which results in about 10^{15} edges in our enormous brain graph!

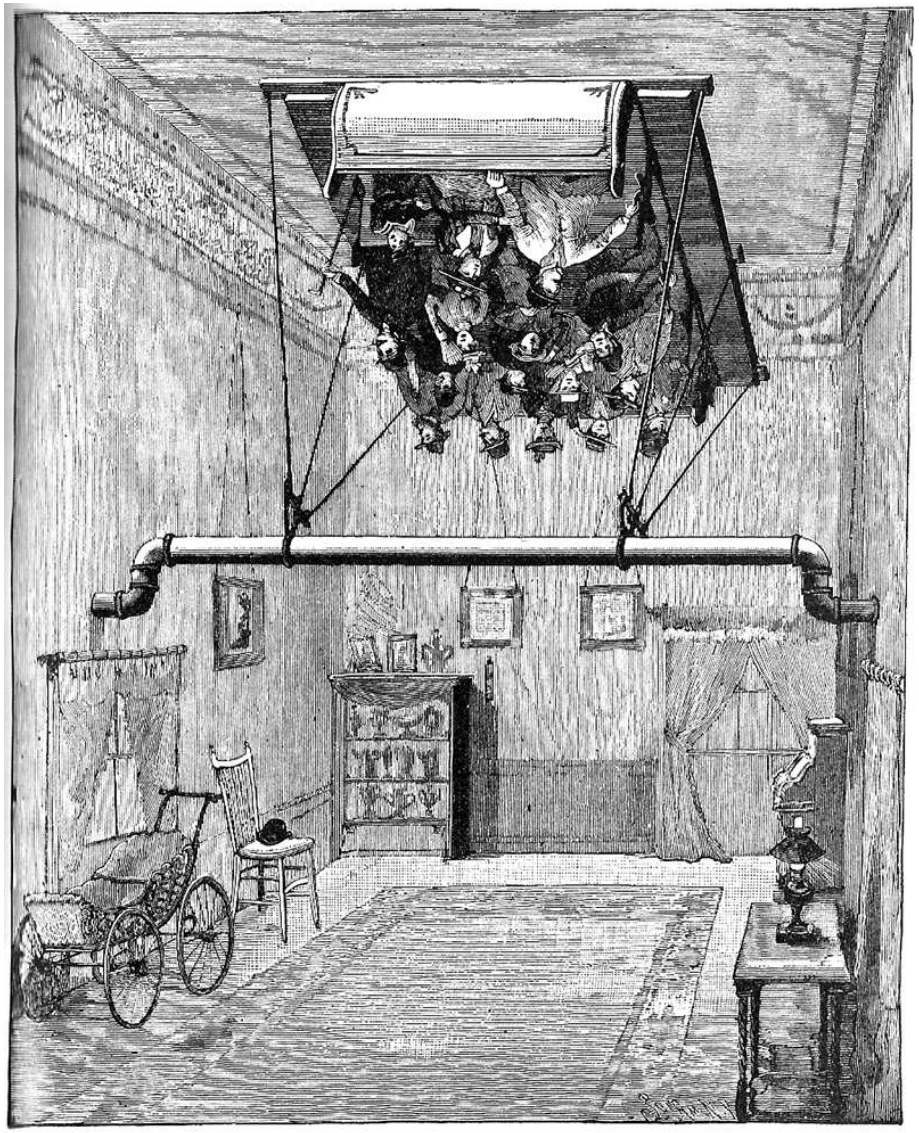
Hierarchical processing Upon leaving the sense-organ receptors, signals propagate among the neurons to eventually reach the cerebral cortex. Along the way, *hierarchical processing* is performed; see Figure 2.19. Through selectivity, each receptor responds to a narrow range of stimuli, across time, space, frequency, and so on. After passing through several neurons, signals from numerous receptors are simultaneously taken into account. This allows increasingly complex patterns to be detected in the stimulus. In the case of vision, feature detectors appear in the early hierarchical stages, enabling us to detect features such as edges, corners, and motion. Once in the cerebral cortex, the signals from sensors are combined

with anything else from our life experiences that may become relevant for making an interpretation of the stimuli. Various *perceptual phenomena* occur, such as recognizing a face or identifying a song. Information or concepts that appear in the cerebral cortex tend to represent a global picture of the world around us. Surprisingly, *topographic mapping* methods reveal that spatial relationships among receptors are maintained in some cases among the distribution of neurons. Also, recall from Section 1.1 that place cells and grid cells encode spatial maps of familiar environments.

Proprioception In addition to information from senses and memory, we also use *proprioception*, which is the ability to sense the relative positions of parts of our bodies and the amount of muscular effort being involved in moving them. Close your eyes and move your arms around in an open area. You should have an idea of where your arms are located, although you might not be able to precisely reach out and touch your fingertips together without using your eyes. This information is so important to our brains that the *motor cortex*, which controls body motion, sends signals called *efference copies* to other parts of the brain to communicate what motions have been executed. Proprioception is effectively another kind of sense. Continuing our comparison with robots, it corresponds to having *encoders* on joints or wheels, to indicate how far they have moved. One interesting implication of proprioception is that you cannot tickle yourself because you know where your fingers are moving; however, if someone else tickles you, then you do not have access to their efference copies. The lack of this information is crucial to the tickling sensation.

Fusion of senses Signals from multiple senses and proprioception are being processed and combined with our experiences by our neural structures throughout our lives. In ordinary life, without VR or drugs, our brains interpret these combinations of inputs in coherent, consistent, and familiar ways. Any attempt to interfere with these operations is likely to cause a mismatch among the data from our senses. The brain may react in a variety of ways. It could be the case that we are not consciously aware of the conflict, but we may become fatigued or develop a headache. Even worse, we could develop symptoms of dizziness or nausea. In other cases, the brain might react by making us so consciously aware of the conflict that we immediately understand that the experience is artificial. This would correspond to a case in which the VR experience is failing to convince someone that they are present in a virtual world. To make an effective and comfortable VR experience, trials with human subjects are essential to understand how the brain reacts. It is practically impossible to predict what would happen in an unknown scenario, unless it is almost identical to other well-studied scenarios.

One of the most important examples of bad sensory conflict in the context of VR is *vection*, which is the illusion of self motion. The conflict arises when your vision sense reports to your brain that you are accelerating, but your balance sense reports that you are motionless. As you walk down the street, the balance and



ILLUSION PRODUCED BY A RIDE IN THE SWING.

Figure 2.20: In the 1890s, a virtual swinging experience was made by spinning the surrounding room instead of the swing. This is known as the *haunted swing illusion*. People who tried it were entertained, but they experienced an extreme version of vection.

vision senses are in harmony. You might have experienced vection before, even without VR. If you are stuck in traffic or stopped at a train station, you might have felt as if you are moving backwards while seeing a vehicle in your periphery that is moving forward. In the 1890s, Amariah Lake constructed an amusement park ride that consisted of a swing that remains at rest while the entire room surrounding the swing rocks back-and-forth (Figure 2.20). In VR, vection is caused by the locomotion operation described in Section 2.2. For example, if you accelerate yourself forward using a controller, rather than moving forward in the real world, then you perceive acceleration with your eyes, but not your vestibular organ. For strategies to alleviate this problem, see Section 10.2.

Adaptation A universal feature of our sensory systems is *adaptation*, which means that the perceived effect of stimuli changes over time. This may happen with any of our senses and over a wide spectrum of time intervals. For example, the perceived loudness of motor noise in an aircraft or car decreases within minutes. In the case of vision, the optical system of our eyes and the photoreceptor sensitivities adapt to change perceived brightness. Over long periods of time, *perceptual training* can lead to adaptation. In military training simulations, sickness experienced by soldiers is greatly reduced by regular exposure []. Anecdotally, the same seems to be true of experienced video game players. Those who have spent many hours and days in front of large screens playing first-person-shooter games apparently experience less vection when locomoting themselves in VR.

Adaptation therefore becomes a crucial factor for VR. Through repeated exposure, developers may become comfortable with an experience that is nauseating to a newcomer. This gives them a terrible bias while developing an experience; recall from Section 1.1 the problem of confusing the scientist with the lab subject in the VR experiment. On the other hand, through repeated, targeted training they may be able to improve their debugging skills by noticing flaws in the system that an "untrained eye" would easily miss. Common examples include:

- A large amount of tracking latency has appeared, which interferes with the *perception of stationarity*.
- The left and right eye views are swapped.
- Objects appear to one eye but not the other.
- One eye view has significantly more latency than the other.
- Straight lines are slightly curved due to uncorrected warping in the optical system.

This disconnect between the actual stimulus and your perception of the stimulus leads to the next topic.

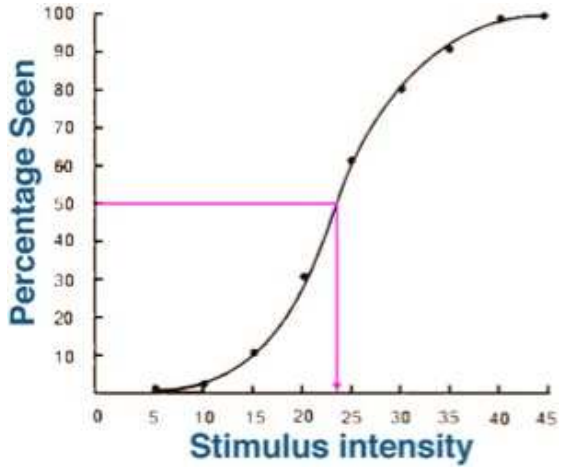


Figure 2.21: The most basic psychometric function. For this example, as the stimulus intensity is increased, the percentage of people detecting the phenomenon increases. The point along the curve that corresponds to 50 percent indicates a critical threshold or boundary in the stimulus intensity.

Psychophysics *Psychophysics* is the scientific study of perceptual phenomena that are produced by physical stimuli. For example, under what conditions would someone call an object "red"? The stimulus corresponds to light entering the eye, and the perceptual phenomenon is the concept of "red" forming in the brain. Other examples of perceptual phenomena are "straight", "larger", "louder", "tickles", and "sour". Figure 2.21 shows a typical scenario in a psychophysical experiment. As one parameter is varied, such as the frequency of a light, there is usually a range of values for which subjects cannot reliably classify the phenomenon. For example, there may be a region where they are not sure whether the light is red. At one extreme, they may consistently classify it as "red" and at the other extreme, they consistently classify it as "not red". For the region in between, the *probability of detection* is recorded, which corresponds to the frequency with which it is classified as "red".

Stevens' power law One of the most known results from psychophysics is *Steven's power law*, which characterizes the relationship between the magnitude of a physical stimulus and its *perceived* magnitude. The hypothesis is that an exponential relationship occurs over a wide range of sensory systems and stimuli:

$$p = cm^x \tag{2.1}$$

in which

- m is the magnitude or intensity of the stimulus,

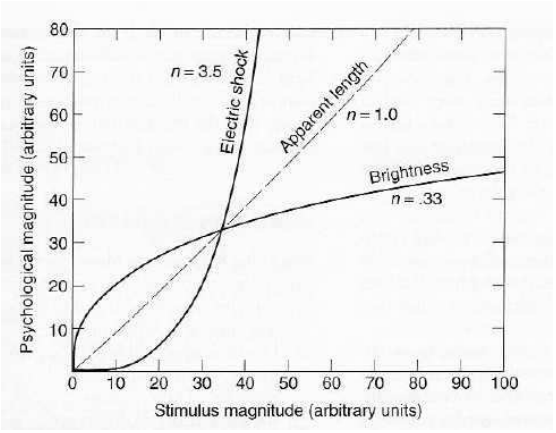


Figure 2.22: Steven's power law captures the relationship between the magnitude of a stimulus and its perceived magnitude. The model is an exponential curve, and the exponent depends on the stimulus type.

- p is the perceived magnitude,
- x relates the actual magnitude to the perceived magnitude, and is the most important part of the equation, and
- c is an uninteresting constant that depends on units.

Note that for $x = 1$, (2.1) is a linear relationship, $p = cm$; see Figure 2.22. An example of this is our perception of the length of an isolated line segment directly in front of our eyes. The length we perceive is proportional to its actual length. The more interesting cases are when $x \neq 1$. For the case of perceiving the brightness of a target in the dark, $x = 0.33$, which implies that a large increase in brightness is perceived as a smaller increase. In the other direction, our perception of electric shock as current through the fingers yields $x = 3.5$. A little more shock is a lot more uncomfortable!

Just noticeable difference Another key psychophysical concept is the *just noticeable difference* (*JND*). This is the amount that the stimulus needs to be changed so that subjects would perceive it to have changed in at least 50 percent of trials. For a large change, all or nearly all subjects would report a change. If the change is too small, then none or nearly none of the subjects would notice. The experimental challenge is to vary the amount of change until the chance of someone reporting a change is 50 percent.

Consider the JND for a stimulus with varying magnitude, such as brightness. How does the JND itself vary as the magnitude varies? This relationship is cap-

tured by *Weber's law*:

$$\frac{\Delta m}{m} = c, \tag{2.2}$$

in which Δm is the JND, m is the magnitude of the stimulus, and c is a constant.

Design of experiments VR disrupts the ordinary perceptual processes of its users. It should be clear from this section that proposed VR systems and experiences need to be evaluated on users to understand whether they are yielding the desired effect while also avoiding unwanted side effects. This amounts to applying the scientific method to make observations, formulate hypotheses, and design experiments that determine their validity. When human subjects are involved, this becomes extremely challenging. How many subjects are enough? What happens if they adapt to the experiment? How does their prior world experience affect the experiment? What if they are slightly sick the day that they try the experiment? What did they eat for breakfast? The answers to these questions could dramatically affect the outcome.

It gets worse. Suppose they already know your hypothesis going into the experiment. This will most likely bias their responses. Also, what will the data from the experiment look like? Will you ask them to fill out a questionnaire, or will you make inferences about their experience from measured data such as head motions, heart rate, and skin conductance? These choices are also critical. See Section 12.3 for more on this topic.

Further Reading

VR sickness survey paper: [66]

Dynimical simulation literature:

More neuroscience: [124]

Basic sensation and perception: [92]

Steven's power law: [142]

Chapter 3

The Geometry of Virtual Worlds

Chapter Status	Taken from *Virtual Reality*, S. M. LaValle
	This online chapter is not the final version! Check **http://vr.cs.uiuc.edu/** for information on the latest draft version. This draft was compiled on June 29, 2016.

Section 2.2 introduced the Virtual World Generator (VWG), which maintains the geometry and physics of the virtual world. This chapter covers the *geometry* part, which is needed to make models and move them around. The models could include the walls of a building, furniture, clouds in the sky, the user's avatar, and so on. Section 3.1 covers the basics of how to define consistent, useful models. Section 3.2 explains how to apply mathematical transforms that move them around in the virtual world. This involves two components: Translation (changing position) and rotation (changing orientation). Section 3.3 presents the best ways to express and manipulate 3D rotations, which are the most complicated part of moving models. Section 3.4 then covers how the virtual world appears if we try to "look" at it from a particular perspective. This is the geometric component of visual rendering, which is covered in Chapter 7. Finally, Section 3.5 puts all of the transformations together, so that you can see how to go from defining a model to having it appear in the right place on the display.

If you work with high-level engines to build a VR experience, then most of the concepts from this chapter might not seem necessary. You might need only to select options from menus and write simple scripts. However, an understanding of the basic transformations, such as how to express 3D rotations or move a camera viewpoint, is essential to making the software do what you want. Furthermore, if you want to build virtual worlds from scratch, or at least want to *understand* what is going on under the hood of a software engine, then this chapter is critical.

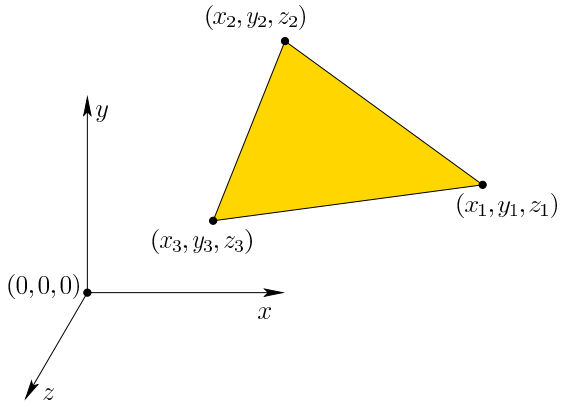


Figure 3.1: Points in the virtual world are given coordinates in a right-handed coordinate system in which the y axis is pointing upward. The origin $(0, 0, 0)$ lies at the point where axes intersect. Also shown is a 3D triangle is defined by its three vertices, each of which is a point in $\mathbb{R}^3$.

3.1 Geometric Models

We first need a virtual world to contain the geometric models. For our purposes, it is enough to have a 3D Euclidean space with Cartesian coordinates. Therefore, let $\mathbb{R}^3$ denote the virtual world, in which every point is represented as a triple of real-valued coordinates: (x, y, z). The coordinate axes of our virtual world are shown in Figure 3.1. We will consistently use right-handed coordinate systems in this book because they represent the predominant choice throughout physics and engineering; however, left-handed systems appear in some places, with the most notable being Microsoft's DirectX graphical rendering library. In these cases, one of the three axes points in the opposite direction in comparison to its direction in a right-handed system. This inconsistency can lead to hours of madness when writing software; therefore, be aware of the differences and their required conversions if you mix software or models that use both. If possible, avoid mixing right-handed and left-handed systems altogether.

Geometric models are made of surfaces or solid regions in $\mathbb{R}^3$ and contain an infinite number of points. Because representations in a computer must be finite, models are defined in terms of *primitives* in which each represents an infinite set of points. The simplest and most useful primitive is a *3D triangle*, as shown in Figure 3.1. A planar surface patch that corresponds to all points "inside" and on the boundary of the triangle is fully specified by the coordinates of the triangle *vertices*:

$$((x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3)). \tag{3.1}$$

To model a complicated object or body in the virtual world, numerous triangles can be arranged into a *mesh*, as shown in Figure 3.2. This provokes many

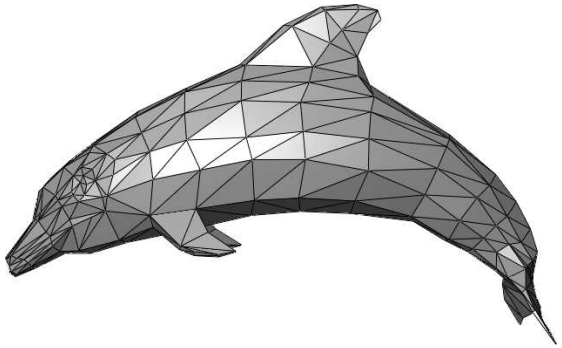


Figure 3.2: A geometric model of a dolphin, formed from a mesh of 3D triangles (from Wikipedia user Chrschn).

important questions:

1. How do we specify how each triangle "looks" whenever viewed by a user in VR?
2. How do we make the object "move"?
3. If the object surface is sharply curved, then should we use curved primitives, rather than trying to approximate the curved object with tiny triangular patches?
4. Is the interior of the object part of the model, or is it represented only by its surface?
5. Is there an efficient algorithm for determining which triangles are adjacent to a given triangle along the surface?
6. Should we avoid duplicating vertex coordinates that are common to many neighboring triangles?

We address these questions in reverse order.

Data structures Consider listing all of the triangles in a file or memory array. If the triangles form a mesh, then most or all vertices will be shared among multiple triangles. This is clearly a waste of space. Another issue is that we will frequently want to perform operations on the model. For example, after moving an object, can we determine whether it is in collision with another object (see Section 8.3)? A typical low-level task might be to determine which triangles share a common vertex or edge with a given triangle. This might require linearly searching through the triangle list to determine whether they share a vertex or two. If there are millions

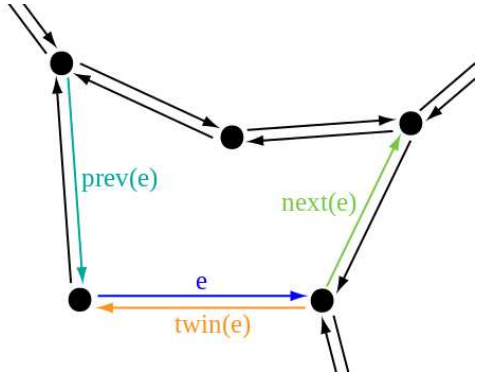


Figure 3.3: Part of a doubly connected edge list is shown here for a face that has five edges on its boundary. Each half-edge structure e stores pointers to the next and previous edges along the face boundary. It also stores a pointer to its twin half-edge, which is part of the boundary of the adjacent face. (Figure from Wikipedia user Accountalive).

of triangles, which is not uncommon, then it would cost too much to perform this operation repeatedly.

For these reasons and more, geometric models are usually encoded in clever data structures. The choice of the data structure should depend on which operations will be performed on the model. One of the most useful and common is the *doubly connected edge list*, also known as *half-edge data structure* [33, 106]. See Figure 3.3. In this and similar data structures, there are three kinds of data elements: *faces*, *edges*, and *vertices*. These represent two, one, and zero-dimensional parts, respectively, of the model. In our case, every face element represents a triangle. Each edge represents the border of one or two, without duplication. Each vertex is shared between one or more triangles, again without duplication. The data structure contains pointers between adjacent faces, edges, and vertices so that algorithms can quickly traverse the model components in a way that corresponds to how they are connected together.

Inside vs. outside Now consider the question of whether the object interior is part of the model (recall Figure 3.2). Suppose the mesh triangles fit together perfectly so that every edge borders exactly two triangles and no triangles intersect unless they are adjacent along the surface. In this case, the model forms a complete barrier between the *inside* and *outside* of the object. If we were to hypothetically fill the inside with a gas, then it could not leak to the outside. This is an example of a *coherent model*. Such models are required if the notion of inside or outside is critical to the VWG. For example, a penny could be inside of the dolphin, but not intersecting with any of its boundary triangles. Would this ever need to be detected? If we remove a single triangle, then the hypothetical gas would leak

out. There would no longer be a clear distinction between the inside and outside of the object, making it difficult to answer the question about the penny and the dolphin. In the extreme case, we could have a single triangle in space. There is clearly no natural inside or outside. Furthermore, the model could be as bad as *polygon soup*, which is a jumble of triangles that do not fit together nicely and could even have intersecting interiors. In conclusion, be careful when constructing models so that the operations you want to perform later will be logically clear. If you are using a high-level design tool, such as Blender or Maya, to make your models, then coherent models will be automatically built.

Why triangles? Continuing upward through the questions above, triangles are used because they are the simplest for algorithms to handle, especially if implemented in hardware. GPU implementations tend to be biased toward smaller representations so that a compact list of instructions can be applied to numerous model parts in parallel. It is certainly possible to use more complicated primitives, such as quadrilaterals, splines, and semi-algebraic surfaces [45, 62, 104]. This could lead to smaller model sizes, but often comes at the expense of greater computational cost for handling each primitive. For example, it is much harder to determine whether two spline surfaces are colliding, in comparison to two 3D triangles.

Stationary vs. movable models There will be two kinds of models in the virtual world $\mathbb{R}^3$:

- *Stationary models*, which keep the same coordinates forever. Typical examples are streets, floors, and buildings.
- *Movable models*, which can be *transformed* into various positions and orientations. Examples include vehicles, avatars, and small furniture.

Motion can be caused in a number of ways. Using a tracking system (Chapter 9), the model might move to match the user's motions. Alternatively, the user might operate a controller to move objects in the virtual world, including a representation of himself. Finally, objects might move on their own according to the laws of physics in the virtual world. Section 3.2 will cover the mathematical operations that move models to the their desired places, and Chapter 8 will describe velocities, accelerations, and other physical aspects of motion.

Choosing coordinate axes One often neglected point is the choice of coordinates for the models, in terms of their placement and scale. If these are defined cleverly at the outset, then many tedious complications can be avoided. If the virtual world is supposed to correspond to familiar environments from the real world, then the axis scaling should match common units. For example, $(1, 0, 0)$ should mean *one meter* to the right of $(0, 0, 0)$. It is also wise to put the origin $(0, 0, 0)$ in a convenient location. Commonly, $y = 0$ corresponds to the floor of a building or

sea level of a terrain. The location of $x = 0$ and $z = 0$ could be in the center of the virtual world so that it nicely divides into quadrants based on sign. Another common choice is to place it in the upper left when viewing the world from above so that all x and z coordinates are nonnegative. For movable models, the location of the origin and the axis directions become extremely important because they affect how the model is rotated. This should become clear in Sections 3.2 and 3.3 as we present rotations.

Viewing the models Of course, one of the most important aspects of VR is how the models are going to "look" when viewed on a display. This problem is divided into two parts. The first part involves determining where the points in the virtual world should appear on the display. This is accomplished by viewing transformations in Section 3.4, which are combined with other transformations in Section 3.5 to produce the final result. The second part involves how each part of the model should appear after taking into account lighting sources and surface properties that are defined in the virtual world. This is the rendering problem, which is covered in Chapter 7.

3.2 Changing Position and Orientation

Suppose that a movable model has been defined as a mesh of triangles. To move it, we apply a single transformation to every vertex of every triangle. This section first considers the simple case of *translation*, followed by the considerably complicated case of *rotations*. By combining translation and rotation, the model can be placed anywhere, and at any orientation in the virtual world $\mathbb{R}^3$.

Translations Consider the following 3D triangle,

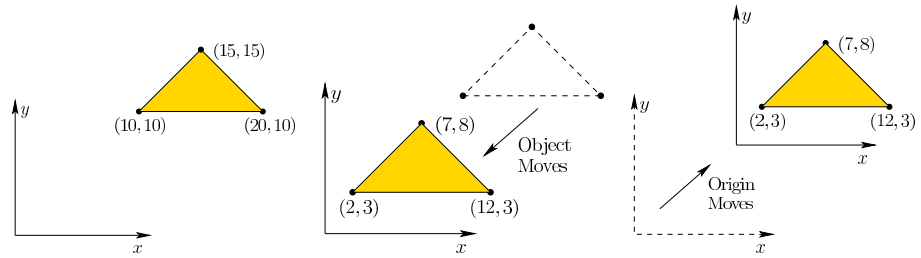
$$((x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3)), \tag{3.2}$$

in which its vertex coordinates are expressed as generic constants.

Let x_t, y_t, and z_t be the amount we would like to change the triangle's position, along the x, y, and z axes, respectively. The operation of changing position is called *translation*, and it is given by

$$\begin{aligned} (x_1, y_1, z_1) &\mapsto (x_1 + x_t, y_1 + y_t, z_1 + z_t) \\ (x_2, y_2, z_2) &\mapsto (x_2 + x_t, y_2 + y_t, z_2 + z_t) \\ (x_3, y_3, z_3) &\mapsto (x_3 + x_t, y_3 + y_t, z_3 + z_t), \end{aligned} \tag{3.3}$$

in which $a \mapsto b$ denotes that a becomes replaced by b after the transformation is applied. Applying (3.3) to every triangle in a model will translate all of it to the desired location. If the triangles are arranged in a mesh, then it is sufficient to apply the transformation to the vertices alone. All of the triangles will retain their size and shape.



(a) Original object (b) Object moves (c) Origin moves

Figure 3.4: Every transformation has two possible interpretations, even though the math is the same. Here is a 2D example, in which a triangle is defined in (a). We could translate the triangle by $x_t = -8$ and $y_t = -7$ to obtain the result in (b). If we instead wanted to hold the triangle fixed but move the origin up by 8 in the x direction and 7 in the y direction, then the coordinates of the triangle vertices change the exact same way, as shown in (c).

Relativity Before the transformations become too complicated, we want to caution you about interpreting them correctly. Figures 3.4(a) and 3.4(b) show an example in which a triangle is translated by $x_t = -8$ and $y_t = -7$. The vertex coordinates are the same in Figures 3.4(b) and 3.4(c). Figure 3.4(b) shows the case we are intended to cover so far: The triangle is interpreted as having moved in the virtual world. However, Figure 3.4(c) shows another possibility: The coordinates of the virtual world have been reassigned so that the triangle is closer to the origin. This is equivalent to having moved the entire world, with the triangle being the only part that does not move. In this case, the translation is applied to the coordinate axes, but they are negated. When we apply more general transformations, this extends so that transforming the coordinate axes results in an *inverse* of the transformation that would correspondingly move the model. Negation is simply the inverse in the case of translation.

Thus, we have a kind of "relativity": Did the object move, or did the whole world move around it? This idea will become important in Section 3.4 when we want to change viewpoints. If we were standing at the origin, looking at the triangle, then the result would appear the same in either case; however, if the origin moves, then we would move with it. A deep perceptual problem lies here as well. If we perceive ourselves as having moved, then VR sickness might increase, even though it was the object that moved. In other words, our brains make their best guess as to which type of motion occurred, and sometimes get it wrong.

Getting ready for rotations How do we make the wheels roll on a car? Or turn a table over onto its side? To accomplish these, we need to change the model's *orientation* in the virtual world. The operation that changes the orientation is called *rotation*. Unfortunately, rotations in three dimensions are much more complicated than translations, leading to countless frustrations for engineers and developers. To improve the clarity of 3D rotation concepts, we first start with a simpler problem: 2D linear transformations.

Consider a 2D virtual world, in which points have coordinates (x, y). You can imagine this as a vertical plane in our original, 3D virtual world. Now consider a generic two-by-two matrix

$$M = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \tag{3.4}$$

in which each of the four entries could be any real number. We will look at what happens when this matrix is multiplied by the point (x, y), when it is written as a column vector.

Performing the multiplication, we obtain

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}, \tag{3.5}$$

in which (x', y') is the transformed point. Using simple algebra, the matrix multiplication yields

$$\begin{aligned} x' &= m_{11}x + m_{12}y \\ y' &= m_{21}x + m_{22}y. \end{aligned} \tag{3.6}$$

Using notation as in (3.3), M is a transformation for which $(x, y) \mapsto (x', y')$.

Applying the 2D matrix to points Suppose we place two points $(1, 0)$ and $(0, 1)$ in the plane. They lie on the x and y axes, respectively, at one unit of distance from the origin $(0, 0)$. Using vector spaces, these two points would be the standard unit basis vectors (sometimes written as $\hat{\imath}$ and $\hat{\jmath}$). Watch what happens if we substitute them into (3.5):

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} m_{11} \\ m_{21} \end{bmatrix} \tag{3.7}$$

and

$$\begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} m_{12} \\ m_{22} \end{bmatrix}. \tag{3.8}$$

These special points simply select the column vectors on M. What does this mean? If M is applied to transform a model, then each column of M indicates precisely how each coordinate axis is changed.

Figure 3.5 illustrates the effect of applying various matrices M to a model. Starting with the upper right, the identity matrix does not cause the coordinates to change: $(x, y) \mapsto (x, y)$. The second example causes a flip as if a mirror were placed at the y axis. In this case, $(x, y) \mapsto (-x, y)$. The second row shows examples of scaling. The matrix on the left produces $(x, y) \mapsto (2x, 2y)$, which doubles the size. The matrix on the right only stretches the model in the y direction, causing an *aspect ratio* distortion. In the third row, it might seem that

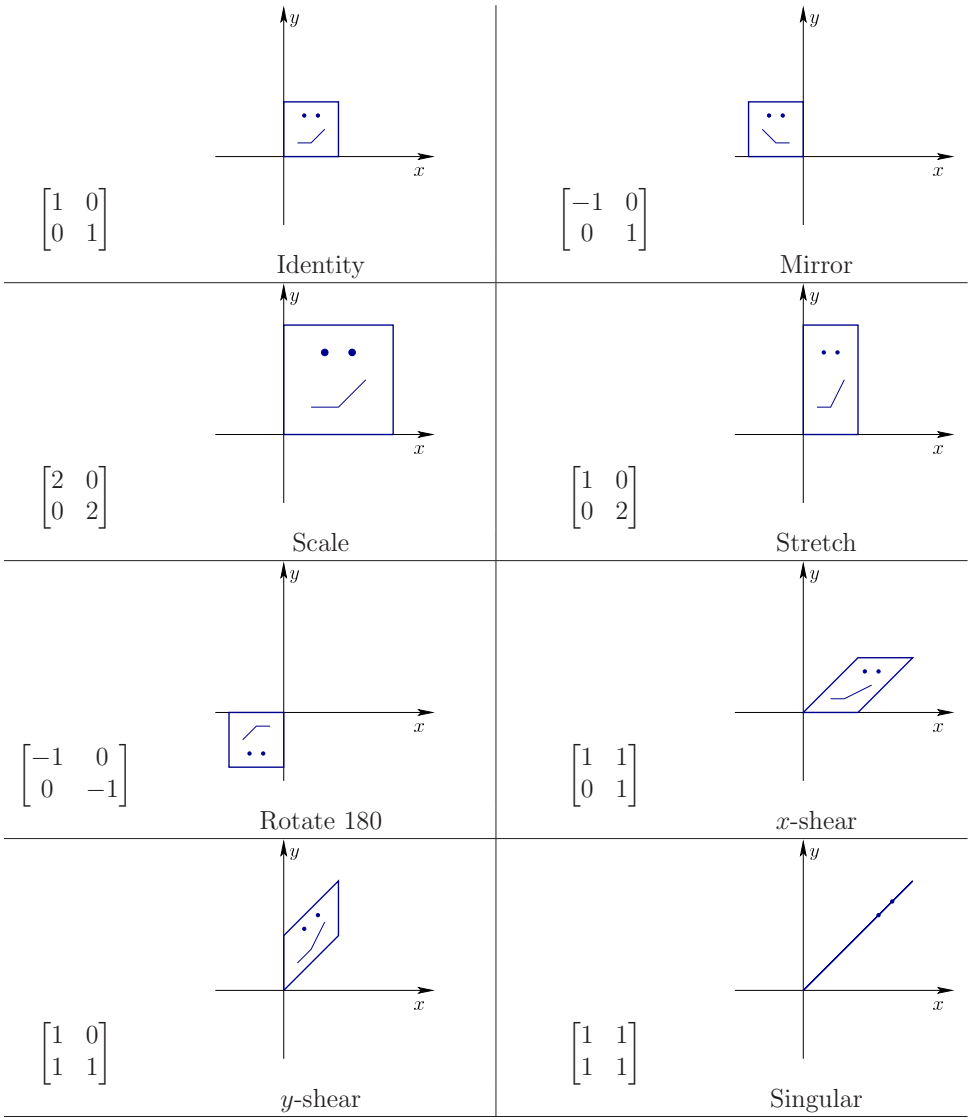


Figure 3.5: Eight different matrices applied to transform a square face. These examples nicely cover all of the possible cases, in a qualitative sense.

the matrix on the left produces a mirror image with respect to both x and y axes. This is true, except that the mirror image of a mirror image restores the original. Thus, this corresponds to the case of a 180-degree (π radians) rotation, rather than a mirror image. The matrix on the right produces a shear along the x direction: $(x, y) \mapsto (x + y, y)$. The amount of displacement is proportional to y. In the bottom row, the matrix on the left shows a skew in the y direction. The final matrix might at first appear to cause more skewing, but it is degenerate. The two-dimensional shape collapses into a single dimension when M is applied: $(x, y) \mapsto (x + y, x + y)$. This corresponds to the case of a *singular* matrix, which means that its columns are not linearly independent (they are in fact identical). A matrix is singular if and only if its determinant is zero.

Only some matrices produce rotations The examples in Figure 3.5 span the main qualitative differences between various two-by-two matrices M. Two of them were rotation matrices: the identity matrix, which is 0 degrees of rotation, and the 180-degree rotation matrix. Among the set of all possible M, which ones are valid rotations? We must ensure that the model does not become distorted. This is achieved by ensuring that M satisfies the following rules:

1. No stretching of axes.
2. No shearing.
3. No mirror images.

If none of these rules is violated, then the result is a rotation.

To satisfy the first rule, the columns of M must have unit length:

$$m_{11}^2 + m_{21}^2 = 1 \text{ and } m_{12}^2 + m_{22}^2 = 1. \tag{3.9}$$

The scaling and shearing transformations in Figure 3.5 violated this.

To satisfy the second rule, the coordinate axes must remain perpendicular. Otherwise, shearing occurs. Since the columns of M indicate how axes are transformed, the rule implies that their inner (dot) product is zero:

$$m_{11}m_{12} + m_{21}m_{22} = 0. \tag{3.10}$$

The shearing transformations in Figure 3.5 violate this rule, which clearly causes right angles in the model to be destroyed.

Satisfying the third rule requires that the determinant of M is positive. After satisfying the first two rules, the only possible remaining determinants are 1 (the normal case) and -1 (the mirror-image case). Thus, the rule implies that:

$$\det \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} = m_{11}m_{22} - m_{12}m_{21} = 1. \tag{3.11}$$

The mirror image example in Figure 3.5 results in $\det M = -1$.

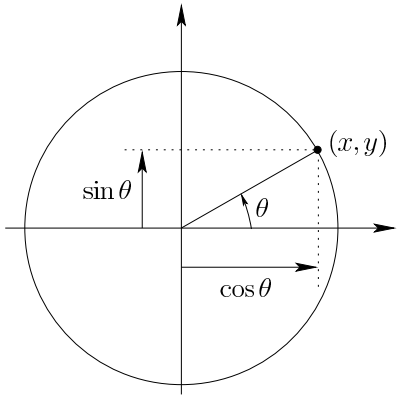


Figure 3.6: For a circle with unit radius, centered at the origin, a single parameter θ reaches all xy points along the circle as it ranges from $\theta = 0$ to $\theta = 2\pi$.

The first constraint (3.9) indicates that each column must be chosen so that its components lie on a unit circle, centered at the origin. In standard planar coordinates, we commonly write the equation of this circle as as $x^2 + y^2 = 1$. Recall the common parameterization of the unit circle in terms of an angle θ that ranges from 0 to 2π radians (see Figure 3.6):

$$x = \cos\theta \text{ and } y = \sin\theta. \tag{3.12}$$

Instead of x and y, we use the notation of the matrix components. Let $m_{11} = \cos\theta$ and $m_{21} = \sin\theta$. Substituting this into M yields

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \tag{3.13}$$

in which m_{12} and m_{22} were uniquely determined by applying (3.10) and (3.11). By allowing θ to range from 0 to 2π, the full range of all allowable rotations is generated.

Think about degrees of freedom. Originally, we could chose all four components of M independently, resulting in 4 DOFs. The constraints in (3.9) each removed a DOF. Another DOF was removed by (3.10). Note that (3.11) does not reduce the DOFs; it instead eliminates exactly half of the possible transformations: The ones that are mirror flips and rotations together. The result is one DOF, which was nicely parameterized by the angle θ. Furthermore, we were lucky that set of all possible 2D rotations can be nicely interpreted as points along a unit circle.

The 3D case Now we try to describe the set of all 3D rotations by following the same general template as the 2D case. The matrix from (3.4) is extended from 2D

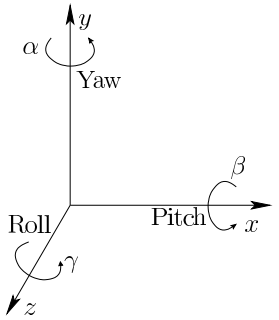


Figure 3.7: Any three-dimensional rotation can be described as a sequence of yaw, pitch, and roll rotations.

to 3D, resulting in 9 components:

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}. \tag{3.14}$$

Thus, we start with 9 DOFs and want to determine what matrices remain as valid rotations. Follow the same three rules from the 2D case. The columns must have unit length. For example, $m_{11}^2 + m_{21}^2 + m_{31}^2 = 1$. This means that the components of each column must lie on a unit sphere. Thus, the unit-length rule reduces the DOFs from 9 to 6. By following the second rule to ensure perpendicular axes result, the pairwise inner products of the columns must be zero. For example, by choosing the first two columns, the constraint is

$$m_{11}m_{12} + m_{21}m_{22} + m_{31}m_{32} = 0. \tag{3.15}$$

We must also apply the rule to the remaining pairs: The second and third columns, and then the first and third columns. Each of these cases eliminates a DOF, resulting in only 3 DOFs remaining. To avoid mirror images, the constraint $\det M = 1$ is applied, which does not reduce the DOFs.

Finally, we arrive at a set of matrices that must satisfy the algebraic constraints; however, they unfortunately do not fall onto a nice circle or sphere. We only know that there are 3 degrees of rotational freedom, which implies that it should be possible to pick three independent parameters for a 3D rotation, and then derive all 9 elements of (3.14) from them.

Yaw, pitch, and roll One of the simplest ways to parameterize 3D rotations is to construct them from "2D-like" transformations, as shown in Figure 3.7. First consider a rotation about the z-axis. Let *roll* be a counterclockwise rotation of γ

about the z-axis. The rotation matrix is given by

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.16}$$

The upper left of the matrix looks exactly like the 2D rotation matrix (3.13), except that θ is replaced by γ. This causes yaw to behave exactly like 2D rotation in the xy plane. The remainder of $R(\gamma)$ looks like the identity matrix, which causes z to remain unchanged after a roll.

Similarly, let *pitch* be a counterclockwise rotation of β about the x-axis:

$$R_x(\beta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta \\ 0 & \sin\beta & \cos\beta \end{bmatrix}. \tag{3.17}$$

In this case, points are rotated with respect to y and z while the x coordinate is left unchanged.

Finally, let *yaw* be a counterclockwise rotation of α about the y-axis:

$$R_y(\alpha) = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix}. \tag{3.18}$$

In this case, rotation occurs with respect to x and z while leaving y unchanged.

Combining rotations Each of (3.16), (3.17), and (3.18) provides a single DOF of rotations. The yaw, pitch, and roll rotations can be combined sequentially to attain any possible 3D rotation:

$$R(\alpha, \beta, \gamma) = R_y(\alpha) R_x(\beta) R_z(\gamma). \tag{3.19}$$

In this case, the range of α is from 0 to 2π; however, the pitch β need only range from $-\pi/2$ to $\pi/2$ while nevertheless reaching all possible 3D rotations.

Be extra careful when combining rotations in a sequence because the operations are not commutative. For example, a yaw by $\pi/2$ followed by a pitch by $\pi/2$ does not produce the same result as the pitch followed by the yaw. You can easily check this by substituting $\pi/2$ into (3.17) and (3.18), and observing how the result depends on the order of matrix multiplication. The 2D case is commutative because the rotation axis is always the same, allowing the rotation angles to additively combine. Having the wrong matrix ordering is one of the most frustrating problems when writing software for VR.

Matrix multiplications are "backwards" Which operation is getting applied to the model first when we apply a product of matrices? Consider rotating a point $p = (x, y, z)$. We have two rotation matrices R and Q. If we rotate p using R, we obtain $p' = Rp$. If we then apply Q, we get $p'' = Qp'$. Now suppose that we instead want to first combine the two rotations and then apply them to p to get p''. Programmers are often temped to combine them as RQ because we read from left to right and also write sequences in this way. However, it is backwards for linear algebra because Rp is already acting from the left side. Thus, it "reads" from right to left.[1] We therefore must combine the rotations as QR to obtain $p'' = QRp$. Later in this chapter, we will be chaining together several matrix transforms. Read them from right to left to understand what they are doing!

Translation and rotation in one matrix It would be convenient to apply both rotation and translation together in a single operation. Suppose we want to apply a rotation matrix R, and follow it with a translation by (x_t, y_t, z_t). Algebraically, this is

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = R \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} x_t \\ y_t \\ z_t. \end{bmatrix} \tag{3.20}$$

Although there is no way to form a single 3 by 3 matrix to accomplish both operations, it can be done by increasing the matrix dimensions by one. Consider the following 4 by 4 *homogeneous transform matrix*:

$$T_{rb} = \begin{bmatrix} & & & x_t \\ & R & & y_t \\ & & & z_t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.21}$$

The notation T_{rb} is used to denote that the matrix is a *rigid body transform*, meaning that it does not distort objects. A homogeneous transform matrix could include other kinds of transforms, which will appear in Section 3.5.

The same result as in (3.20) can be obtained by performing multiplication with (3.21) as follows:

$$\begin{bmatrix} & & & x_t \\ & R & & y_t \\ & & & z_t \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ 1. \end{bmatrix} \tag{3.22}$$

Because of the extra dimension, we extended the point (x, y, z) by one dimension, to obtain $(x, y, z, 1)$. Note that (3.21) represents rotation *followed by* translation, not the other way around. Translation and rotation do not commute; therefore, this is an important point.

Inverting transforms We frequently want to invert (or undo) transformations. For a translation (x_t, y_t, z_t), we simply apply the negation $(-x_t, -y_t, -z_t)$. For a

[1] Perhaps coders who speak Arabic or Hebrew are not confused about this.

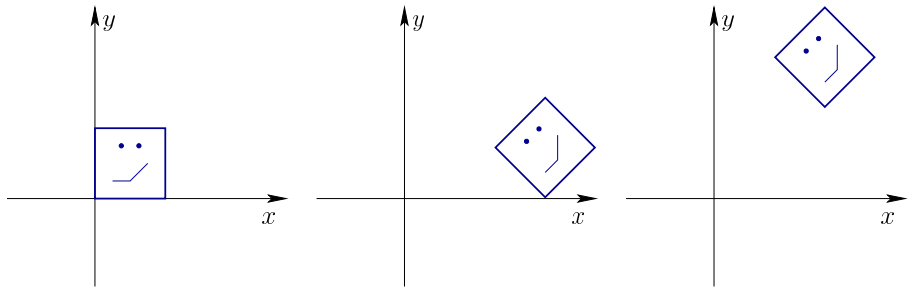


Figure 3.8: (a) A rigid model that is contained in a one-by-one square. (b) The result after rotation by $\pi/4$ (45 degrees), following by translation by $x_t = 2$. (c) The result after reversing the order: Translation by $x_t = 2$, following by rotation by $\pi/4$.

general matrix transform M, we apply the matrix inverse M^{-1} (if it exists). This is often complicated to calculate. Fortunately, inverses are much simpler for our cases of interest. In the case of a rotation matrix R, the inverse is equal to the transpose $R^{-1} = R^T$.[2] To invert the homogeneous transform matrix (3.21), it is tempting to write

$$\begin{bmatrix} & & & -x_t \\ & R^T & & -y_t \\ & & & -z_t \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.23}$$

This will undo both the translation and the rotation; however, the order is wrong. Remember that these operations are not commutative, which implies that order must be correctly handled. See Figure 3.8. The algebra for very general matrices (part of noncommutative group theory) works out so that the inverse of a product of matrices reverses their order:

$$(ABC)^{-1} = C^{-1}B^{-1}A^{-1}. \tag{3.24}$$

This can be seen by putting the inverse next to the original product: $ABCC^{-1}B^{-1}A^{-1}$. In this way, C cancels with its inverse, followed by B and its inverse, and finally A and its inverse. If the order were wrong, then these cancellations would not occur.

The matrix T_{rb} (from 3.21) applies the rotation first, followed by translation. Applying (3.23) undoes the rotation first and then translation, without reversing the order. Thus, the inverse of T_{rb} is

$$\begin{bmatrix} & & & 0 \\ & R^T & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_t \\ 0 & 1 & 0 & -y_t \\ 0 & 0 & 1 & -z_t \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.25}$$

The matrix on the right first undoes the translation (with no rotation). After that, the matrix on the left undoes the rotation (with no translation).

[2]Recall that to transpose a square matrix, we simply swap the i and j indices, which turns columns into rows.

3.3 Axis-Angle Representations of Rotation

As observed in Section 3.2, 3D rotation is complicated for several reasons: 1) Nine matrix entries are specified in terms of only three independent parameters, and with no simple parameterization, 2) the axis of rotation is not the same every time, and 3) the operations or noncommutative, implying that the order of matrices is crucial. None of these problems existed for the 2D case.

Kinematic singularities An even worse problem arises when using yaw, pitch, roll angles (and related Euler-angle variants). Even though they start off being intuitively pleasing, the representation becomes degenerate, leading to *kinematic singularities* that are nearly impossible to visualize. An example will be presented shortly. To prepare for this, recall how we represent locations on the Earth. These are points in $\mathbb{R}^3$, but are represented with longitude and latitude coordinates. Just like the limits of yaw and pitch, longitude ranges from 0 to 2π and latitude only ranges from $-\pi/2$ to $\pi/2$. (Longitude is usually expressed as 0 to 180 degrees west or east, which is equivalent.) As we travel anywhere on the Earth, the latitude and longitude coordinates behave very much like xy coordinates; however, we tend to stay away from the poles. Near the North Pole, the latitude behaves normally, but the longitude could vary a large amount while corresponding to a tiny distance traveled. Recall how a wall map of the world looks near the poles: Greenland is enormous and Antarctica wraps across the entire bottom (assuming it uses a projection that keeps longitude lines straight). The poles themselves are the kinematic singularities: At these special points, you can vary longitude, but the location on the Earth is not changing. One of two DOFs seems to be lost.

The same problem occurs with 3D rotations, but it is harder to visualize due to the extra dimension. If the pitch angle is held at $\beta = \pi/2$, then a kind of "North Pole" is reached in which α and γ vary independently but cause only one DOF (in the case of latitude and longitude, it was one parameter varying but causing DOFs). Here is how it looks when combining the yaw, pitch, and roll matrices:

$$\begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(\alpha-\gamma) & \sin(\alpha-\gamma) & 0 \\ 0 & 0 & -1 \\ -\sin(\alpha-\gamma) & \cos(\alpha-\gamma) & 0 \end{bmatrix}. \tag{3.26}$$

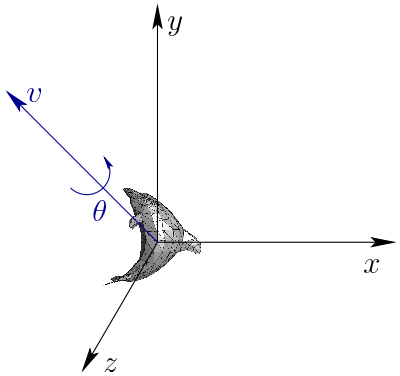


Figure 3.9: Euler's rotation theorem states that every 3D rotation can be considered as a rotation by an angle θ about an axis through the origin, given by the unit direction vector $v = (v_1, v_2, v_3)$.

The second matrix above corresponds to pitch (3.17) with $\beta = \pi/2$. The result on the right is obtained by performing matrix multiplication and applying a subtraction trigonometric identity. You should observe that the resulting matrix is a function of both α and γ, but there is one DOF because only the difference $\alpha - \gamma$ affects the resulting rotation. In the video game industry there has been some back-and-forth battles about whether this problem is crucial. In an FPS game, the avatar is usually not allowed to pitch his head all the way to $\pm\pi/2$, thereby avoiding this problem. In VR, it happens all the time that a user could pitch her head straight up or down. The kinematic singularity often causes the virtual world to apparently spin uncontrollably. This phenomenon occurs when sensing and controlling a spacecraft's orientation using mechanical gimbals; the result is called *gimbal lock*.

The problems can be easily solved with *axis-angle* representations of rotation. They are harder to learn than yaw, pitch, and roll; however, it is a worthwhile investment because it avoids these problems. Furthermore, many well-written software libraries and game engines work directly with these representations. Thus, to use them effectively, you should understand what they are doing.

The most important insight to solving the kinematic singularity problems is Euler's rotation theorem (1775), shown in Figure 3.9. Even though the rotation axis may change after rotations are combined, Euler showed that *any* 3D rotation can be expressed as a rotation θ about some axis that pokes through the origin. This matches the three DOFs for rotation: It takes two parameters to specify the direction of an axis and one parameter for θ. The only trouble is that conversions back and forth between rotation matrices and the axis-angle representation are somewhat inconvenient. This motivates the introduction of a mathematical object that is close to the axis-angle representation, closely mimics the algebra of 3D rotations, and can even be applied directly to rotate models. The perfect

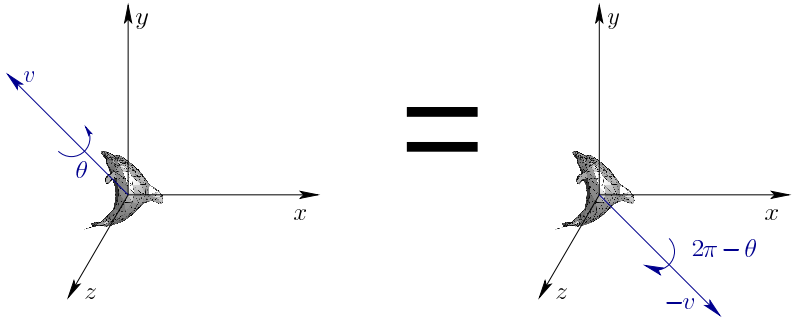


Figure 3.10: There are two ways to encode the same rotation in terms of axis and angle, using either v or $-v$.

representation: *Quaternions*.

Two-to-one problem Before getting to quaternions, it is important point out one annoying problem with Euler's rotation theorem. As shown in Figure 3.10, it does not claim that the axis-angle representation is unique. In fact, for every 3D rotation other than the identity, there are exactly two representations. This is due to the fact that the axis could "point" in either direction. We could insist that the axis always point in one direction, such as positive y, but this does not fully solve the problem because of the boundary cases (horizontal axes). Quaternions, which are coming next, nicely handle all problems with 3D rotations except this one, which is unavoidable.

Quaternions were introduced in 1843 by William Rowan Hamilton. When seeing them the first time, most people have difficulty understanding their peculiar algebra. Therefore, we will instead focus on precisely which quaternions correspond to which rotations. After that, we will introduce some limited quaternion algebra. The algebra is much less important for developing VR systems, unless you want to implement your own 3D rotation library. The correspondence between quaternions and 3D rotations, however, is crucial.

A quaternion h is a 4D vector:

$$q = (a, b, c, d), \tag{3.27}$$

in which a, b, c, and d can take on real values. Thus, q can be considered as a point in $\mathbb{R}^4$. It turns out that we will only use *unit quaternions*, which means that

$$a^2 + b^2 + c^2 + d^2 = 1 \tag{3.28}$$

must always hold. This should remind you of the equation of a unit sphere ($x^2 + y^2 + z^2 = 1$), but it is one dimension higher. A sphere is a 2D surface, whereas the set of all unit quaternions is a 3D "hypersurface", more formally known as a

Quaternion	Axis-Angle	Description
$(1, 0, 0, 0)$	$(\text{undefined}, 0)$	Identity rotation
$(0, 1, 0, 0)$	$((1, 0, 0), \pi)$	Pitch by π
$(0, 0, 1, 0)$	$((0, 1, 0), \pi)$	Yaw by π
$(0, 0, 0, 1)$	$((0, 0, 1), \pi)$	Roll by π
$(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, 0)$	$((1, 0, 0), \pi/2)$	Pitch by $\pi/2$
$(\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}, 0)$	$((0, 1, 0), \pi/2)$	Yaw by $\pi/2$
$(\frac{1}{\sqrt{2}}, 0, 0, \frac{1}{\sqrt{2}})$	$((0, 0, 1), \pi/2)$	Roll by $\pi/2$

Figure 3.11: For these cases, you should be able to look at the quaternion and quickly picture the axis and angle of the corresponding 3D rotation.

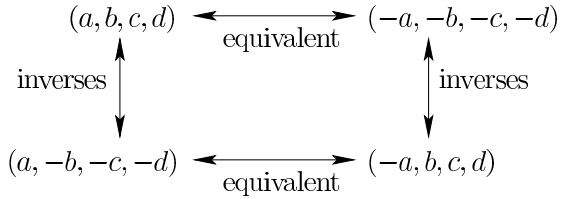


Figure 3.12: Simple relationships between equivalent quaternions and their inverses.

manifold [16, 70]. We will use the space of unit quaternions to represent the space of all 3D rotations. Both have 3 DOFs, which seems reasonable.

Let (v, θ) be an axis-angle representation of a 3D rotation, as depicted in Figure 3.9. Let this be represented by the following quaternion:

$$q = \left(\cos \frac{\theta}{2} \, , \; v_1 \sin \frac{\theta}{2} \, , \; v_2 \sin \frac{\theta}{2} \, , \; v_3 \sin \frac{\theta}{2} \right). \tag{3.29}$$

Think of q as a data structure that encodes the 3D rotation. It is easy to recover (v, θ) from q:

$$\theta = 2 \cos^{-1} a \text{ and } v = \frac{1}{\sqrt{1 - a^2}} (b, c, d). \tag{3.30}$$

If $a = 1$, then (3.30) breaks; however, this corresponds to the case of the identity rotation.

You now have the mappings $(v, \theta) \mapsto q$ and $q \mapsto (v, \theta)$. To test your understanding, Figure 3.11 shows some simple examples, which commonly occur in practice. Furthermore, Figure 3.12 shows some simple relationships between quaternions and their corresponding rotations. The horizontal arrows indicate that q and $-q$ represent the same rotation. This is true because of the double representation issue shown in Figure 3.10. Applying (3.29) to both cases establishes their equivalence. The vertical arrows correspond to inverse rotations. These hold

because reversing the direction of the axis causes the rotation to be be reversed (rotation by θ becomes rotation by $-\theta$.

How do we apply the quaternion $h = (a, b, c, d)$ to rotate the model? One way is to use the following conversion into a 3D rotation matrix:

$$R(h) = \begin{bmatrix} 2(a^2 + b^2) - 1 & 2(bc - ad) & 2(bd + ac) \\ 2(bc + ad) & 2(a^2 + c^2) - 1 & 2(cd - ab) \\ 2(bd - ac) & 2(cd + ab) & 2(a^2 + d^2) - 1 \end{bmatrix}. \tag{3.31}$$

A more efficient way exists which avoids converting into a rotation matrix. To accomplish this, we need to define quaternion multiplication. For any two quaternions, q_1 and q_2, let $q_1 * q_2$ denote the product, which is defined as

$$\begin{aligned} a_3 &= a_1 a_2 - b_1 b_2 - c_1 c_2 - d_1 d_2 \\ b_3 &= a_1 b_2 + a_2 b_1 + c_1 d_2 - c_2 d_1 \\ c_3 &= a_1 c_2 + a_2 c_1 + b_2 d_1 - b_1 d_2 \\ d_3 &= a_1 d_2 + a_2 d_1 + b_1 c_2 - b_2 c_1. \end{aligned} \tag{3.32}$$

In other words, $q'' = q * q'$ as defined in (3.32).

Here is a way to rotate the point (x, y, z) using the rotation represented by h. Let $p = (x, y, z, 1)$, which is done to give the point the same dimensions as a quaternion. Perhaps surprisingly, the point is rotated by applying quaternion multiplication as

$$p' = q * p * q^{-1}, \tag{3.33}$$

in which $q^{-1} = (a, -b, -c, -d)$ (recall from Figure 3.12). The rotated point is (x', y', z'), which is taken from the result $p' = (x', y', z', 1)$.

Here is a simple example for the point $(1, 0, 0)$. Let $p = (1, 0, 0, 1)$ and consider executing a yaw rotation by $\pi/2$. According to Figure 3.11, the corresponding quaternion is $q = (0, 0, 1, 0)$. The inverse is $q^{-1} = (0, 0, -1, 0)$. After tediously applying (3.32) to calculate (3.33), the result is $p' = (0, 1, 0, 1)$. Thus, the rotated point is $(0, 1, 0)$, which is a correct yaw by $\pi/2$.

3.4 Viewing Transformations

This section describes how to transform the models in the virtual world so that they appear on a virtual screen. The main purpose is to set the foundation for graphical rendering, which adds effects due to lighting, material properties, and quantization. Ultimately, the result appears on the physical display. One side effect of these transforms is that they also explain how cameras form images, at least the idealized mathematics of the process. Think of this section as describing a virtual camera that is placed in the virtual world. What should the virtual picture, taken by that camera, look like? To make VR work correctly, the "camera" should actually be one of two virtual human eyes that are placed into the virtual world.

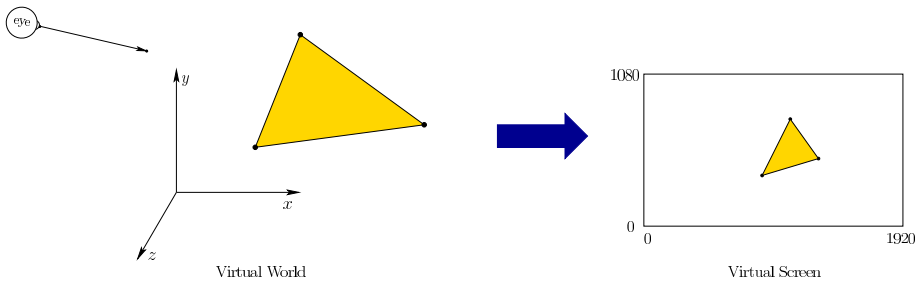


Figure 3.13: If we placed a virtual eye or camera into the virtual world, what would it see? Section 3.4 provides transformations that place objects from the virtual world onto a virtual screen, based on the particular viewpoint of a virtual eye. A flat rectangular shape is chosen for engineering and historical reasons, even though it does not match the shape of our retinas.

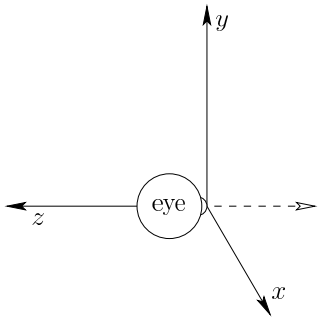


Figure 3.14: Consider an eye that is looking down the z axis in the negative direction. The origin of the model is the point at which light enters the eye.

Thus, what should a virtual eye see, based on its position and orientation in the virtual world? Rather than determine precisely what would appear on the retina, which should become clear after Section 4.4, here we merely calculate where the model vertices would appear on a flat, rectangular screen in the virtual world. See Figure 3.13.

An eye's view Figure 3.14 shows a virtual eye that is looking down the negative z axis. It is placed in this way so that from the eye's perspective, x increases to the right and y is upward. This corresponds to familiar Cartesian coordinates. The alternatives would be: 1) to face the eye in the positive z direction, which makes the xy coordinates appear backwards, or 2) reverse the z axis, which would unfortunately lead to a left-handed coordinate system. Thus, we have made an odd choice that avoids worse complications.

Suppose that the eye is an object model that we want to place into the virtual

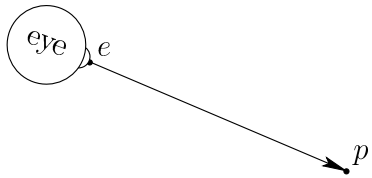


Figure 3.15: The vector from the eye position e to a point p that it is looking at is normalized to form $\hat{c}$ in (3.36).

world $\mathbb{R}^3$ at some position $e = (e_1, e_2, e_3)$ and orientation given by the matrix

$$R_{eye} = \begin{bmatrix} \hat{x}_1 & \hat{y}_1 & \hat{z}_1 \\ \hat{x}_2 & \hat{y}_2 & \hat{z}_2 \\ \hat{x}_3 & \hat{y}_3 & \hat{z}_3 \end{bmatrix}. \tag{3.34}$$

If the eyeball in Figure 3.14 were made of triangles, then rotation by R_{eye} and translation by e would be applied to all vertices to place it in $\mathbb{R}^3$.

This does not, however, solve the problem of how the virtual world should appear to the eye. Rather than moving the eye in the virtual world, we need to move all of the models in the virtual world to the eye's frame of reference. This means that we need to apply the *inverse* transformation. The inverse rotation is R_{eye}^T, the transpose of R_{eye}. The inverse of e is $-e$. Applying (3.25) results in the appropriate transform:

$$T_{eye} = \begin{bmatrix} \hat{x}_1 & \hat{x}_2 & \hat{x}_3 & 0 \\ \hat{y}_1 & \hat{y}_2 & \hat{y}_3 & 0 \\ \hat{z}_1 & \hat{z}_2 & \hat{z}_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -e_1 \\ 0 & 1 & 0 & -e_2 \\ 0 & 0 & 1 & -e_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.35}$$

Note that R_{eye}, as shown in (3.34), has been transposed and placed into the left matrix above. Also, the order of translation and rotation have been swapped, which is required for the inverse, as mentioned in Section 3.2.

Following Figure 3.4, there are two possible interpretations of (3.35). As stated, this could correspond to moving all of the virtual world models (corresponding to Figure 3.4(b)). A more appropriate interpretation in the current setting is that the virtual world's coordinate frame is being moved so that it matches the eye's frame from Figure 3.14. This corresponds to the case of Figure 3.4(c), which was not the appropriate interpretation in Section 3.2.

Starting from a look-at For VR, the position and orientation of the eye in the virtual world are given by a tracking system and possibly controller inputs. By contrast, in computer graphics, it is common to start with a description of where the eye is located and which way it is looking. This is called a *look-at*, and has the following components:

1. Position of the eye: e

2. Central looking direction of the eye: $\hat{c}$

3. Up direction: $\hat{u}$.

Both $\hat{c}$ and $\hat{u}$ are unit vectors. The first direction $\hat{c}$ corresponds to the center of the view. Whatever $\hat{c}$ is pointing at should end up in the center of the display. If we want this to be a particular point p in $\mathbb{R}^3$ (see Figure 3.15), then $\hat{c}$ can be calculated as

$$\hat{c} = \frac{p - e}{\|p - e\|}, \tag{3.36}$$

in which $\| \cdot \|$ denotes the length of a vector. The result is just the vector from e to p, but normalized.

The second direction $\hat{u}$ indicates which way is up. Imagine holding a camera out as if you are about to take a photo and then perform a roll rotation. You can make level ground appear to be slanted or even upside down in the picture. Thus, $\hat{u}$ indicates the up direction for the virtual camera or eye.

We now construct the resulting transform T_{eye} from (3.35). The translation components are already determined by e, which was given in the look-at. We need only to determine the rotation R_{eye}, as expressed in (3.34). Recall from Section 3.2 that the matrix columns indicate how the coordinate axes are transformed by the matrix (refer to (3.7) and (3.8)). This simplifies the problem of determining R_{eye}. Each column vector is calculated as

$$\begin{aligned} \hat{z} &= -\hat{c} \\ \hat{x} &= \hat{u} \times \hat{z} \\ \hat{y} &= \hat{z} \times \hat{x}. \end{aligned} \tag{3.37}$$

The minus sign appears for calculating $\hat{z}$ because the eye is looking down the negative z axis. The $\hat{x}$ direction is calculated using the standard cross product $\hat{z}$. For the third equation, we could use $\hat{y} = \hat{u}$; however, $\hat{z} \times \hat{x}$ will cleverly correct cases in which $\hat{u}$ generally points upward but is not perpendicular to $\hat{c}$. The unit vectors from (3.37) are substituted into (3.34) to obtain R_{eye}. Thus, we have all the required information to construct T_{eye}.

Orthographic projection Let (x, y, z) denote the coordinates any point, after T_{eye} has been applied. What would happen if we took all points and directly projected them into the vertical xy plane by forcing each z coordinate to be 0? In other words, $(x, y, z) \mapsto (x, y, 0)$, which is called *orthographic projection*. If we imagine the xy plane as a virtual display of the models, then there would be several problems:

1. A jumble of objects would be superimposed, rather than hiding parts of a model that are in front of another.

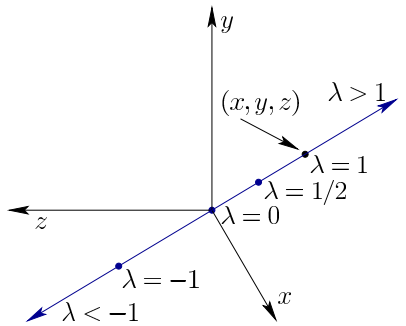


Figure 3.16: Starting with any point (x, y, z), a line through the origin can be formed using a parameter λ. It is the set of all points of the form $(\lambda x, \lambda y, \lambda z)$ for any real value λ. For example, $\lambda = 1/2$ corresponds to the midpoint between (x, y, z) and $(0, 0, 0)$ along the line.

2. The display wound extend infinitely in all directions (except z). If the display is a small rectangle in the xy plane, then the model parts that are outside of its range can be eliminated.

3. Objects that are closer should appear larger than those further away. This happens in the real world. Recall from Section 1.3 (Figure 1.19(c)) paintings that correctly handle perspective.

The first two problems are important graphics operations that are deferred until Chapter 7. The third problem is addressed next.

Perspective projection Instead of using orthographic projection, we define a *perspective projection*. For each point (x, y, z), consider a line through the origin. This is the set of all points with coordinates

$$(\lambda x, \lambda y, \lambda z), \tag{3.38}$$

in which λ can be any real number. In other words λ is a parameter that reaches all points on the line that contains both (x, y, z) and $(0, 0, 0)$. See Figure 3.16.

Now we can place a planar "movie screen" anywhere in the virtual world and see where all of the lines pierce it. To keep the math simple, we pick the $z = -1$ plane to place our virtual screen directly in front of the eye. Using the third component of (3.38), we have $\lambda z = -1$, implying that $\lambda = -1/z$. Using the first two components of (3.38), the coordinates for the points on the screen are calculated as $x' = -x/z$ and $y' = -y/z$. Note that since x and y are scaled by the same amount z for each axis, their aspect ratio is preserved on the screen.

More generally, suppose the vertical screen is placed some location d along the z axis. In this case, we obtain more general expressions for the location of a point

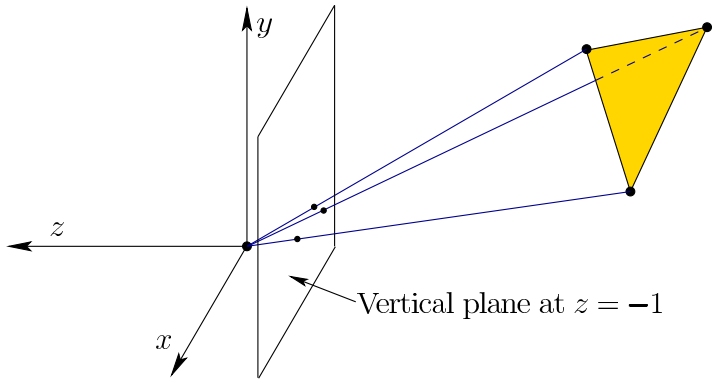


Figure 3.17: An illustration of perspective projection. The model vertices are projected onto a virtual screen by drawing lines through them and the origin $(0, 0, 0)$. The "image" of the points on the virtual screen corresponds to the interesections of the line with the screen.

on the screen:

$$\begin{aligned} x' &= dx/z \\ y' &= dy/z. \end{aligned} \tag{3.39}$$

This was obtained by solving $d = \lambda z$ for λ and substituting it into (3.38).

This is all we need to project the points onto a virtual screen, while respecting the scaling properties of objects at various distances. Getting this right in VR helps in the perception of depth and scale, which are covered in Section 6.1. In Section 3.5, we will adapt (3.39) using transformation matrices. Furthermore, only points that lie within a zone in front of the eye will be projected onto the virtual screen. Points that are too close, too far, or in outside the normal field of view will not be rendered on the virtual screen; this is addressed in Section 3.5 and Chapter 7.

3.5 Chaining the Transformations

This section links all of the transformations of this chapter together while also slightly adjusting their form to match what is currently used in the VR and computer graphics industries. Some of the matrices appearing in this section may seem unnecessarily complicated. The reason is that the expressions are motivated by algorithm and hardware issues, rather than mathematical simplicity. In particular, there is a bias toward putting every transformation into a 4 by 4 homogeneous transform matrix, even in the case of perspective projection which is not even linear (recall (3.39)). In this way, an efficient matrix multiplication algorithm can be iterated over the chain of matrices to produce the result.

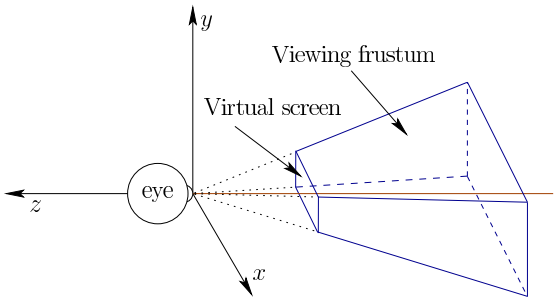


Figure 3.18: The viewing frustrum.

The chain generally appears as follows:

$$T = T_{vp} T_{can} T_{eye} T_{rb}. \tag{3.40}$$

When T is applied to a point $(x, y, z, 1)$, the location of the point on the screen is produced. Remember that these matrix multiplications are not commutative, and the operations are applied from right to left. The first matrix T_{rb} is the rigid body transform (3.21) applied to points on a movable model. For each rigid object in the model, T_{rb} remains the same; however, different objects will generally be placed in various positions and orientations. For example, the wheel of a virtual car will move differently than the avatar's head. After T_{rb} is applied, T_{eye} transforms the virtual world into the coordinate frame of the eye, according to (3.35). At a fixed instant in time, this and all remaining transformation matrices are the same for all points in the virtual world. Here we assume that the eye is positioned at the midpoint between the two virtual human eyes, leading to a *cyclopean viewpoint*. Later in this section, we will extend it to the case of left and right eyes so that stereo viewpoints can be constructed.

Canonical view transform The next transformation, T_{can} performs the perspective projection as described in Section 3.4; however, we must explain how it is unnaturally forced into a 4 by 4 matrix. We also want the result to be in a canonical form that appears to be unitless, which is again motivated by industrial needs. Therefore, T_{can} is called the *canonical view transform*. Figure 3.18 shows a *viewing frustum*, which is based on the four corners of a rectangular *virtual screen*. At $z = n$ and $z = f$ lie a *near plane* and *far plane*, respectively. Note that $z < 0$ for these cases because the z axis points in the opposite direction. The virtual screen is contained in the near plane. The perspective projection should place all of the points inside of the frustum onto a virtual screen that is centered in the near plane. This implies $d = n$ using (3.39).

We now want to try to reproduce (3.39) using a matrix. Consider the result of

applying the following matrix multiplication:

$$\begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} nx \\ ny \\ nz \\ z \end{bmatrix}. \tag{3.41}$$

In the first two coordinates, we obtain the numerator of (3.39). The nonlinear part of (3.39) is the $1/z$ factor. To handle this, the fourth coordinate is used to represent z, rather than 1 as in the case of T_{rb}. From this point onward, the resulting 4D vector is interpreted as a 3D vector that is scaled by dividing out its fourth component. For example, (v_1, v_2, v_3, v_4) is interpreted as

$$(v_1/v_4, v_2/v_4, v_3/v_4). \tag{3.42}$$

Thus, the result from (3.41) is interpreted as

$$(nx/z, ny/z, n), \tag{3.43}$$

in which the first two coordinates match (3.41) with $d = n$, and the third coordinate is the location of the virtual screen along the z axis.

Keeping track of depth for later use The following matrix is commonly used in computer graphics, and will be used here in our chain:

$$T_p = \begin{bmatrix} n & 0 & 0 & 0 \\ 0 & n & 0 & 0 \\ 0 & 0 & n+f & -fn \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{3.44}$$

It is identical to the matrix in (3.41) except in how it transforms the z coordinate. For purposes of placing points on the virtual screen, it is unnecessary because we already know they are all placed at $z = n$. The z coordinate is therefore co-opted for another purpose: Keeping track of the distance of each point from the eye so that graphics algorithms can determine which objects are in front of other objects. The matrix T_p calculates the third coordinate as

$$(n + f)z - fn \tag{3.45}$$

When divided by z, (3.45) does not preserve the exact distance, but the graphics methods (some of which are covered in Chapter 7) require only that the distance *ordering* is preserved. In other words, if point p is further from the eye than point q, then it remains further after the transformation, even if the distances are distorted. It does, however, preserve the distance in two special cases: $z = n$ and $z = f$. This can be seen by substituting these into (3.45) and dividing by z.

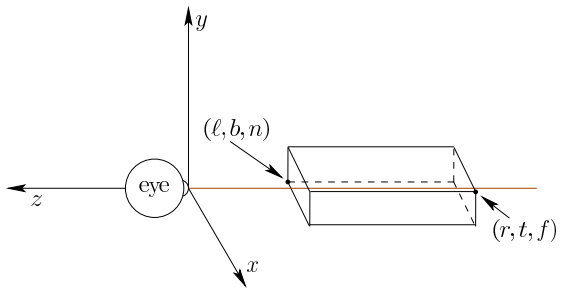


Figure 3.19: The rectangular region formed by the corners of the viewing frustum, after they are transformed by T_p. The coordinates of the selected opposite corners provide the six parameters, ℓ, r, b, t, n, and f, which used in T_{st}.

Additional translation and scaling After T_p is applied, the 8 corners of the frustum are transformed into the corners of a rectangular box, shown in Figure 3.19. The following performs a simple translation of the box along the z axis and some rescaling so that it is centered at the origin and the coordinates of its corners are $(\pm 1, \pm 1, \pm 1)$:

$$T_{st} = \begin{bmatrix} \frac{2}{r-\ell} & 0 & 0 & -\frac{r+\ell}{r-\ell} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.46}$$

If the frustrum is perfectly centered in the xy plane, then the first two components of the last column become 0. Finally, we define the canonical view transform T_{can} from (3.40) as

$$T_{can} = T_{st} T_p. \tag{3.47}$$

Viewport transform The last transform to be applied in the chain (3.40) is the *viewport transform* T_{vp}. After T_{can} has been applied, the x and y coordinates each range from -1 to 1. One last step is required to bring the projected points to the coordinates used to index pixels on a physical display. Let m be the number of horizontal pixels and n be the number of vertical pixels. For example, $n = 1080$ and $m = 1920$ for a 1080p display. Suppose that the display is indexed with rows running from 0 to $n - 1$ and columns from 0 to $m - 1$. Furthermore, $(0, 0)$ is in the lower left corner. In this case, the viewport transform is

$$T_{vp} = \begin{bmatrix} \frac{m}{2} & 0 & 0 & \frac{m-1}{2} \\ 0 & \frac{n}{2} & 0 & \frac{n-1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.48}$$

Left and right eyes We now address how the transformation chain (3.40) is altered for stereoscopic viewing. Let t denote the distance between the left and

right eyes. Its value in the real world varies across people, and its average is around $t = 0.064$ meters. To handle the left eye view, we need to simply shift the cyclopean (center) eye horizontally to the left. Recall from Section 3.4 that the inverse actually gets applied. The models need to be shifted to the right. Therefore, let

$$T_{left} = \begin{bmatrix} 1 & 0 & 0 & \frac{t}{2} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.49}$$

which corresponds to a right shift of the models, when viewed from the eye. This transform is placed after T_{eye} to adjust its output. The appropriate modification to (3.40) is:

$$T = T_{vp} T_{can} T_{left} T_{eye} T_{rb}. \tag{3.50}$$

By symmetry, the right eye is similarly handled by replacing T_{left} in (3.50) with

$$T_{right} = \begin{bmatrix} 1 & 0 & 0 & -\frac{t}{2} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.51}$$

This concludes the explanation of the entire chain of transformations to place and move models in the virtual world and then have them appear in the right place on a display. After reading Chapter 4, it will become clear that one final transformation may be needed after the entire chain has been applied. This is done to compensate for nonlinear optical distortions that occur due to wide-angle lenses in VR headsets.

Further Reading

References on transforming chains of bodies ([77] Chapter 3), and animating articulated structures.

The fact that mesh orientations cannot be consistently labeled for some surfaces is the basis of homology! Should include some topology references.

Euler angle references.

Need quaternion algebra references, more conversions, and derivations of all the given conversions.

Chapter 4

Light and Optics

Chapter Status	Taken from *Virtual Reality*, S. M. LaValle
	This online chapter is not the final version! Check **http://vr.cs.uiuc.edu/** for information on the latest draft version. This draft was compiled on June 29, 2016.

Knowing how light propagates in the physical world is crucial to understanding VR. One reason is the interface between visual displays and our eyes. Light is emitted from displays and arrives on our retinas in a way that convincingly reproduces how light arrives through normal vision in the physical world. In the current generation of VR headsets, a system of both engineered and natural lenses (parts of our eyes) guide the light. Another reason to study light propagation is the construction of virtual worlds. Chapter 3 covered purely *geometric* aspects of modeling. The next logical step is to model how light propagates through virtual worlds to be rendered on a display; this will be continued in Chapter 7. Finally, light propagation is also helpful to understanding how cameras work, which provides another way present a virtual world: Through panoramic videos.

Section 4.1 covers basic physical properties of light, including its interaction with materials and its spectral properties. Section 4.2 provides idealized models of how lenses work. Section 4.3 then shows many ways that lens behavior deviates from the ideal model, thereby degrading VR experiences. Section 4.4 introduces the human eye as an optical system of lenses, before eyes and human vision are covered in much more detail in Chapter 5. Cameras, which can be considered as engineered eyes, are introduced in Section 4.5.

4.1 Basic Behavior of Light

Light can be described in three ways that appear to be mutually incompatible:

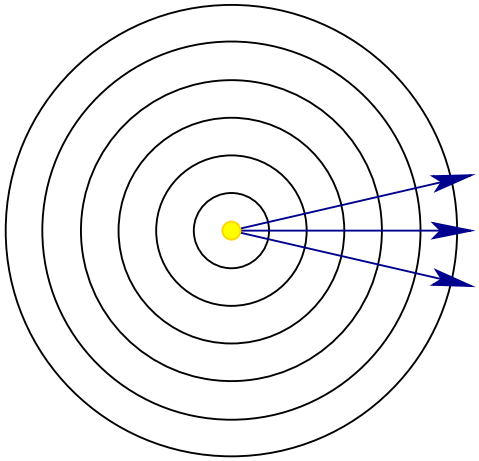


Figure 4.1: Waves and visibility rays emanating from a point light source.

1. Photons: Tiny particles of energy moving through space at high speeds (no need for quantum mechanics in this book!). This interpretation is helpful when considering the amount of light received by a sensor or receptor.

2. Waves: Ripples through space that are similar to waves propagating on the surface of water, but are 3D. The *wavelength* is the distance between peaks. This interpretation is helpful when considering the spectrum of colors.

3. Rays: A ray traces the motion of a single hypothetical photon. The direction is perpendicular to the wavefronts (see Figure 4.1). This interpretation is helpful when explaining lenses and defining the concept of visibility.

Fortunately, modern physics has explained how these interpretations are in fact compatible; each is useful in this book.

Spreading waves Figure 4.1 shows how waves would propagate from a hypothetical point light source. The density would be the same in all directions (radial symmetry), but would decrease as the light source becomes more distant. Recall that the surface area of a sphere with radius r is $4\pi r^2$. Consider centering a spherical screen around the light source. The total number of photons per second hitting a screen of radius 1 should be the same as for a screen of radius 2; however, the density (photons per second per area) should decrease by a factor of $1/4$ because they are distributed over 4 times the area. Thus, photon density decreases quadratically as a function of distance from a point light source.

The curvature of the wavefronts also decreases as the point light source becomes further away. If the waves were to propagate infinitely far away, then they would completely flatten as shown in Figure 4.2. This results in the important case

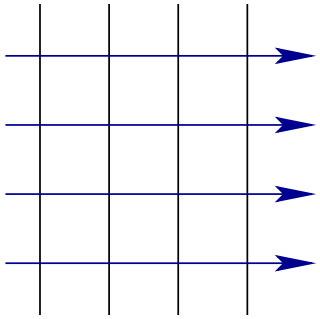


Figure 4.2: If the point light source were "infinitely far" away, then parallel wavefronts would be obtained. Other names for this setting are: Collimated light, parallel rays, rays from infinity, rays to infinity, and zero vergence.

of parallel wavefronts. Without the help of lenses or mirrors, it is impossible to actually obtain this case from a tiny light source in the physical world because it cannot be so far away; however, it serves as both a useful approximation for distant light sources and as an ideal way to describe lenses mathematically. Keep in mind that at any finite distance from a point light source, the rays of light always diverge; it is impossible to make them converge without the help of lenses or mirrors.

Interactions with materials As light strikes the surface of a material, one of three behaviors might occur, as shown in Figure 4.3. In the case of *transmission*, the energy travels through the material and exits the other side. For a transparent material, such as glass, the transmitted light rays are slowed down and bend according to Snell's law, which will be covered in Section 4.2. For a translucent material that is not transparent, the rays scatter into various directions before exiting. In the case of *absorption*, energy is absorbed by the material as the light becomes trapped. The third case is *reflection*, in which the light is deflected from the surface. Along a perfectly smooth or polished surface, the rays reflect in the same way: The exit angle is equal to the entry angle. See Figure 4.4. This case is called *specular reflection*, in contrast to *diffuse reflection*, in which the reflected rays scatter in arbitrary directions. Usually, all three cases of transmission, absorption, and reflection occur simultaneously. The amount of energy divided between the cases depends on many factors, such as the angle of approach, the wavelength, and differences between the materials.

A jumble of wavelengths Figure 4.1 presented an oversimplified view that will make it easy to understand idealized lenses in Section 4.2. Unfortunately, it misses many details that become important in other settings, such as understanding lens aberrations (Section 4.3) or how light interacts with materials in the physical world.

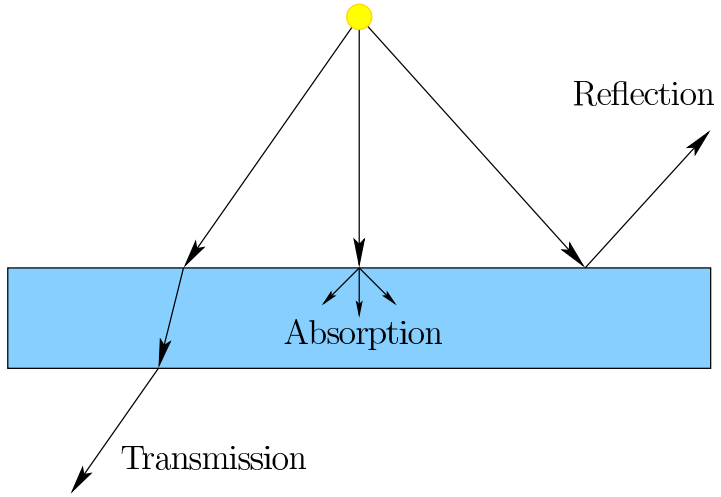


Figure 4.3: As light energy hits the boundary of a different medium, there are three possibilities: transmission, absorption, and reflection.

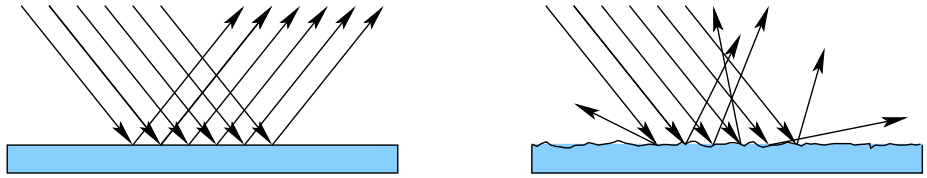


Figure 4.4: Two extreme modes of reflection are shown. Specular reflection means that all rays reflect at the same angle at which they approached. Diffuse reflection means that the rays scatter in a way that could be independent of their approach angle. Specular reflection is common for a polished surface, such as a mirror, whereas diffuse reflection corresponds to a rough surface.

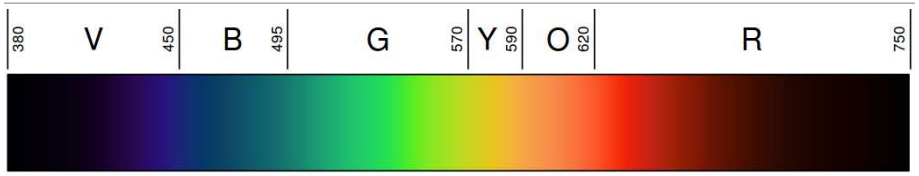


Figure 4.5: Visible light spectrum corresponds to the range of electromagnetic waves that have wavelengths between 400nm and 700nm. (Figure by David Eccles for Wikipedia.)

The remainder of this section therefore considers various realistic complications that arise.

Coherent versus jumbled light The first complication is that light sources usually do not emit *coherent light*, a term that means the wavefronts are perfectly aligned in time and space. A laser is an exceptional case that indeed produces coherent light. It emits parallel waves of a constant wavelength that are also synchronized in time so that their peaks align as they propagate. Common light sources, such as light bulbs and the sun, instead emit a jumble of waves that have various wavelengths and do not have their peaks aligned.

Wavelengths and colors To make sense out of the jumble of waves, we will describe how they are distributed in terms of wavelengths. Figure 4.5 shows the range of wavelengths that are visible to humans. Each wavelength corresponds to a *spectral color*, which is what we would perceive with a coherent light source fixed at that wavelength alone. Wavelengths between 700 and 1000nm are called *infrared*, which are not visible to us, but our cameras can sense them (see Section 9.3). Wavelengths between 100 and 400nm are called *ultraviolet*; they are not part of our visible spectrum, but some birds, insects, and fish can perceive ultraviolet wavelengths over 300nm. Thus, our notion of visible light is already tied to *human* perception.

Spectral power Figure 4.6 shows how the wavelengths are distributed for common light sources. An ideal light source would have all visible wavelengths represented with equal energy, leading to idealized *white* light. The opposite is total darkness, which is *black*. We usually do not allow a light source to propagate light directly onto our retinas (don't stare at the sun!). Instead, we observe light that is reflected from objects all around us, causing us to perceive their color. Each surface has its own distribution of wavelengths that it *reflects*. The fraction of light energy that is reflected back depends on the wavelength, leading to the plots shown in Figure 4.7. For us to perceive an object surface as red, the red wavelengths must be included in the light source *and* the surface must strongly

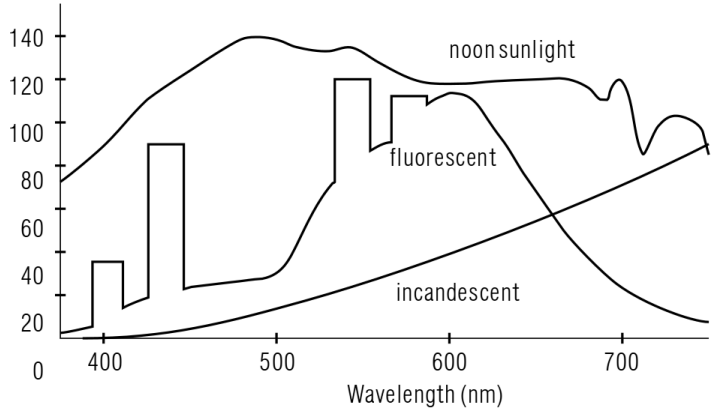


Figure 4.6: The *spectral power distribution* for some common light sources. (Figure from [133]).

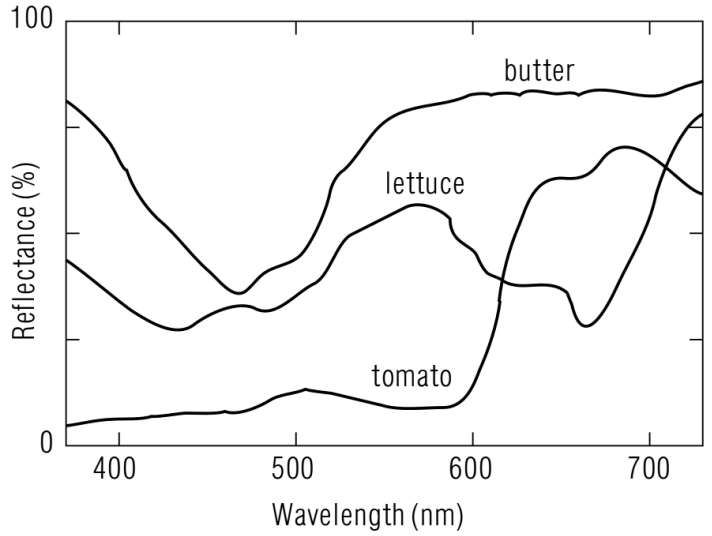
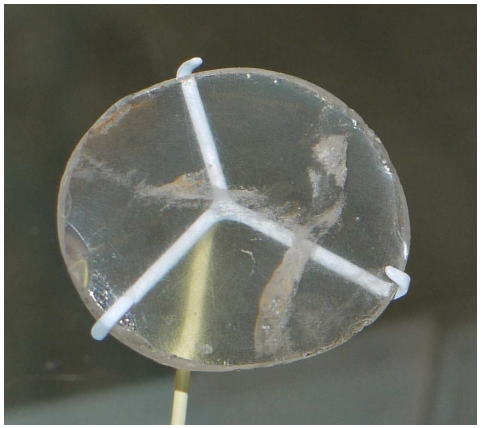
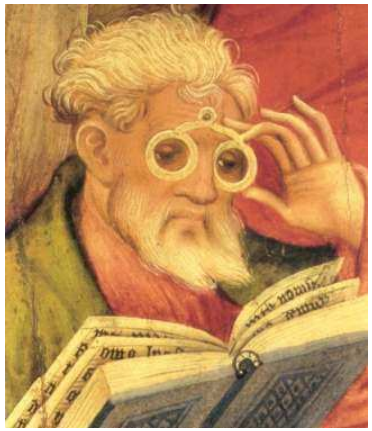


Figure 4.7: The *spectral reflection function* of some common familiar materials. (Figure from [133]).



(a)



(b)

Figure 4.8: (a) The earliest known artificially constructed lens, which was made between 750 and 710 BC in ancient Assyrian Nimrud. It is not known whether this artifact was purely ornamental or used to produce focused images. Picture from the British Museum. (b) A painting by Conrad con Soest from 1403, which shows the use of reading glasses for an elderly male.

reflect red wavelengths. Other wavelengths must also be suppressed. For example, the light source could be white (containing all wavelengths) and the object could strongly reflect all wavelengths, causing the surface to appear white, not red. Section 6.3 will provide more details on color perception.

Frequency Often times, it is useful to talk about frequency instead of wavelength. The *frequency* is the number of times per second that wave peaks pass through a fixed location. Using both the wavelength λ and the speed s, the frequency f is calculated as:

$$f = \frac{s}{\lambda}. \tag{4.1}$$

The speed of light in a vacuum is a universal constant c with value approximately equal to 3×10^8 m/s. In this case, $s = c$ in (4.1). Light propagates roughly 0.03 percent faster in a vacuum than in air, causing the difference to be neglected in most engineering calculations. Visible light in air has a frequency range of roughly 400 to 800 terahertz, which is obtained by applying (4.1). As light propagates through denser media, such as water or lenses, s is significantly smaller; that difference is the basis of optical systems, which are covered next.

4.2 Lenses

Lenses have been made for thousands of years, with the oldest known artifact shown in Figure 4.8(a). It was constructed before 700 BC in Assyrian Nimrud, which was coincidentally mentioned in Figure 1.15 of Chapter 1. Whether constructed from transparent materials or from polished surfaces that act as mirrors, lenses bend rays of light so that a focused image is formed. Over the centuries, their uses have given rise to several well-known devices, such as eyeglasses (Figure 4.8(b)), telescopes, magnifying glasses, binoculars, cameras, and microscopes. Optical engineering is therefore filled with design patterns that indicate how to optimize the designs of these well-designed devices. VR headsets are a newcomer among existing optical devices, leading to many new challenges that are outside of standard patterns that have existed for centuries. Thus, the lens design patterns for VR are still being written. To first step toward addressing the current challenges is to understand how simple lenses work.

Snell's Law Lenses work because of Snell's Law, which expresses how much rays of light bend when entering and exiting a transparent material. Recall that the speed of light in a medium is less than the speed c in an vacuum. For a given material, let its *refractive index* be defined as

$$n = \frac{c}{s}, \tag{4.2}$$

in which s is the speed of light in the medium. For example, $n = 2$ means that light takes twice as long to traverse the medium as through a vacuum. For some common examples, $n = 1.000293$ for air, $n = 1.33$ for water, and $n = 1.523$ for crown glass.

Figure 4.9 shows what happens to incoming light waves and rays. Suppose in this example that the light is traveling from air into glass, so that $n_1 < n_2$. Let θ_1 represent the incoming angle with respect to the surface normal, and let θ_2 represent the resulting angle as it passes through the material. Snell's law relates the four quantities as

$$n_1 \sin\theta_1 = n_2 \sin\theta_2. \tag{4.3}$$

Typically, n_1/n_2 and θ_1 are given, so that (4.3) is solved for θ_2 to obtain

$$\theta_2 = \sin^{-1}\left(\frac{n_1 \sin\theta_1}{n_2}\right). \tag{4.4}$$

If $n_1 < n_2$, then θ_2 is closer to perpendicular than θ_1. If $n_1 > n_2$, then θ_2 is further from perpendicular. The case of $n_1 > n_2$ is also interesting in that light may not penetrate the surface if the incoming angle θ_1 is too large. The range of $\sin^{-1}$ is 0 to 1, which implies that (4.4) provides a solution for θ_2 only if $(n_1/n_2)\sin\theta_1 \leq 1$. If the condition does not hold, then the light rays always reflect from the surface. This situation occurs while under water and looking up at the surface. Rather than being able to see the world above, you might instead see a reflection, depending on the viewing angle.

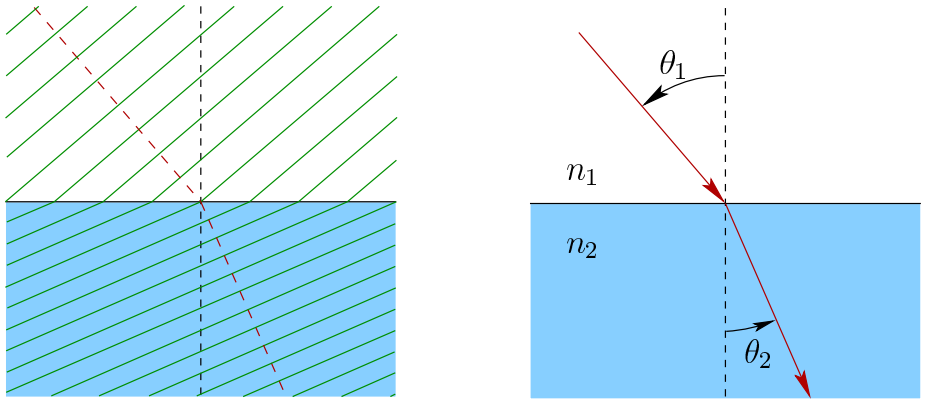


Figure 4.9: Propagating wavefronts from a medium with low refractive index (such as air) to one with a higher index (such as glass). (a) The effect of slower propagation on the wavefronts is shown as they enter the lower medium. (b) This shows the resulting bending of a light ray, which is always perpendicular to the wavefronts. Snell's Law relates the refractive indices and angles as $n_1 \sin\theta_1 = n_2 \sin\theta_2$.

Prisms Imagine shining a laser beam through a prism, as shown in Figure 4.10. Snell's Law can be applied to calculate how the light ray bends after it enters and exits the prism. Note that for the upright prism, a ray pointing slightly upward becomes bent downward. Recall that a larger refractive index inside the prism would cause greater bending. By placing the prism upside down, rays pointing slightly downward are bent upward. Once the refractive index is fixed, the bending depends only on the angles at which the rays enter and exist the surface, rather than the thickness of the prism. To construct a lens, we will exploit this principle and construct a kind of curved version of Figure 4.10.

Simple convex lens Figure 4.11 shows a simple convex lens, which should remind you of the prisms in Figure 4.10. Instead of making a diamond shape, the lens surface is spherically curved so that incoming, parallel, horizontal rays of light converge to a point on the other side of the lens. This special place of convergence is called the *focal point*. Its distance from the lens center is called the *focal depth* or *focal length*.

The incoming rays in Figure 4.11 are special in two ways: 1) They are parallel, thereby corresponding to a source that is infinitely far away, and 2) they are perpendicular to the plane in which the lens is centered. If the rays are parallel but not perpendicular to the lens plane, then the focal point shifts accordingly, as shown in Figure 4.12. In this case, the focal point is not on the optical axis. There are two DOFs of incoming ray directions, leading to a *focal plane* that contains all of the focal points. Unfortunately, this planarity is just an approximation; Section

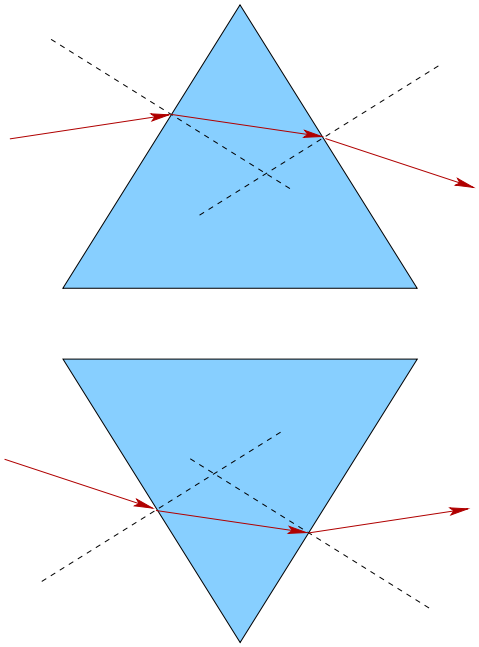


Figure 4.10: The upper part shows how a simple prism bends ascending rays into descending rays, provided that the incoming ray slope is not too high. This was achieved by applying Snell's Law at the incoming and outgoing boundaries. Placing the prism upside down causes descending rays to become ascending. Putting both of these together, we will see that a lens is like a stack of prisms that force diverging rays to converge through the power of refraction.

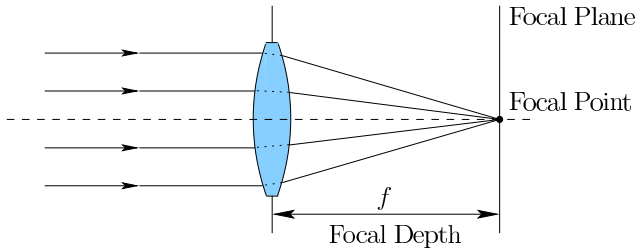


Figure 4.11: A simple convex lens causes parallel rays to converge at the focal point. The dashed line is the *optical axis*, which is perpendicular to the lens and pokes through its center.

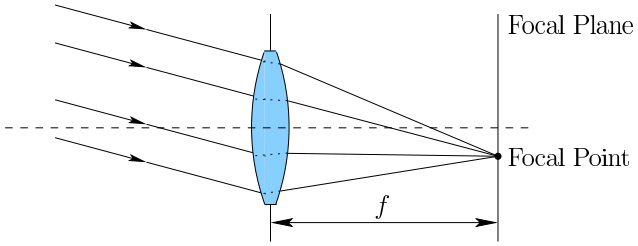


Figure 4.12: If the rays are not perpendicular to the lens, then the focal point is shifted away from the optical axis.

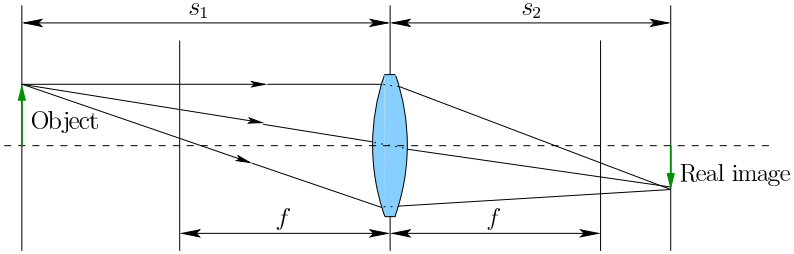


Figure 4.13: In the real world, an object is not infinitely far away. When placed at distance s_1 from the lens, a real image forms in a focal plane at distance $s_2 > f$ behind the lens, as calculated using (4.5).

4.3 explains what really happens. In this idealized setting, a *real image* is formed in the image plane, as if it were a projection screen that is showing how the world looks in front of the lens (assuming everything in the world is very far away).

If the rays are not parallel, then it may still be possible to focus them into a real image, as shown in Figure 4.13. Suppose that a lens is given that has focal length f. If the light source is placed at distance s_1 from the lens, then the rays from that will be in focus if and only if the following equation is satisfied:

$$\frac{1}{s_1} + \frac{1}{s_2} = \frac{1}{f}. \tag{4.5}$$

Figure 4.11 corresponds to the idealized case in which $s_1 = \infty$, for which solving (4.5) yields $s_2 = f$. What if the object being viewed is not completely flat and lying in a plane perpendicular to the lens? In this case, there does not exist a single plane behind the lens that would bring the entire object into focus. We must tolerate the fact that most of it will be approximately in focus. Unfortunately, this is the situation almost always encountered in the real world, including the focus provided by our own eyes (see Section 4.4).

If the light source is placed too close to the lens, then the outgoing rays might be diverging so much that the lens cannot force them to converge. If $s_1 = f$, then the outgoing rays would be parallel ($s_2 = \infty$). If $s_1 < f$, then (4.5) yields $s_2 < 0$.

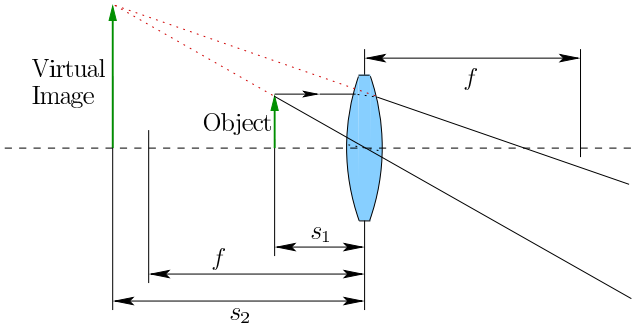


Figure 4.14: If the object is very close to the lens, then the lens cannot force its outgoing light rays to converge to a focal point. In this case, however, a virtual image appears and the lens works as a magnifying glass. This is the way lenses are commonly used for VR headsets.

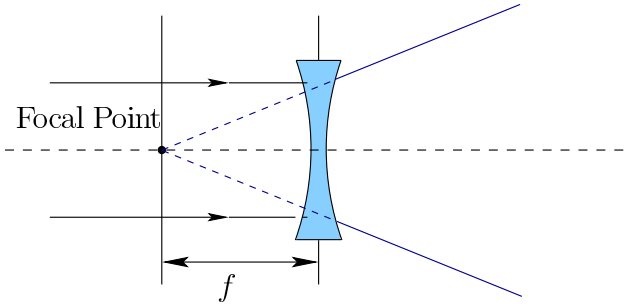


Figure 4.15: In the case of a concave lens, parallel rays are forced to diverge. The rays can be extended backward through the lens to arrive at a focal point on the left side. The usual sign convention is that $f < 0$ for concave lenses.

In this case, a real image is not formed; however, something interesting happens: The phenomenon of *magnification*. A *virtual image* appears when looking into the lens, as shown in Figure 4.14. This exactly what happens in the case of the View-Master and the VR headsets that were shown in Figure 2.11. The screen is placed so that it appears magnified. To the user viewing looking through the screen, it appears as if the screen is infinitely far away (and quite enormous!).

Lensmaker's equation For a given simple lens, the focal length f can be calculated using the Lensmaker's Equation,

$$(n_2 - n_1)\left(\frac{1}{r_1} + \frac{1}{r_2}\right) = \frac{1}{f}, \tag{4.6}$$

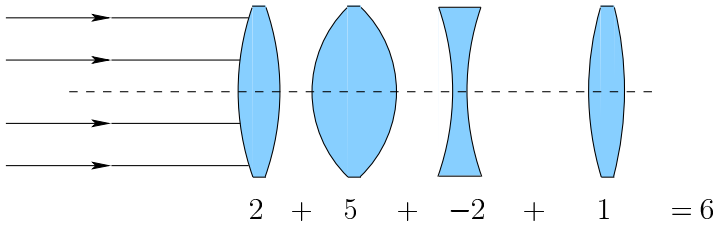


Figure 4.16: To calculate the combined optical power of a chain of lenses, the algebra is simple: Add their diopters. This arrangement of four lenses is equivalent to a 6-diopter lens, which has a focal length of 0.1667m.

which is derived from Snell's law []. The parameters r_1 and r_2 represent the radius of curvature of each of the two lens surfaces (front and back). This version assumes a *thin lens approximation*, which means that the lens thickness is small relative to r_1 and r_2. Also, it is typically assumed that $n_1 = 1$, which is approximately true for air.

Concave lenses For the sake of completeness, we include the case of a *concave* simple lens, shown in Figure 4.15. Parallel rays are forced to diverge, rather than converge; however, a meaningful notion of negative focal length exists by tracing the diverging rays backwards through the lens. The Lensmaker's Equation (4.6) can be slightly adapted to calculate negative f in this case [].

Diopters For optical systems used in VR, several lenses will be combined in succession. What is the effect of the combination? A convenient method to answer this question with simple arithmetic was invented by ophthalmologists. The idea is to define a *diopter*, which is $D = 1/f$. Thus, it is the reciprocal of the focal length. If a lens focuses parallel rays at a distance of 0.2m in behind the lens, then $D = 5$. A larger diopter D means greater converging power. Likewise, a concave lens yields $D < 0$ by using a negative lens. To combine several lenses in succession, we simply add their diopters to determine their equivalent power as a single, simple lens. Figure 4.16 shows a simple example.

4.3 Optical Aberrations

If lenses in the real world behaved *exactly* as described in Section 4.2, then VR systems would be much simpler and more impressive than they are today. Unfortunately, numerous imperfections, called *aberrations*, degrade the images formed by lenses. Because these problems are perceptible in everyday uses, such as viewing content through VR headsets or images from cameras, they are important to understand so that some compensation for them can be designed into the VR system or content.

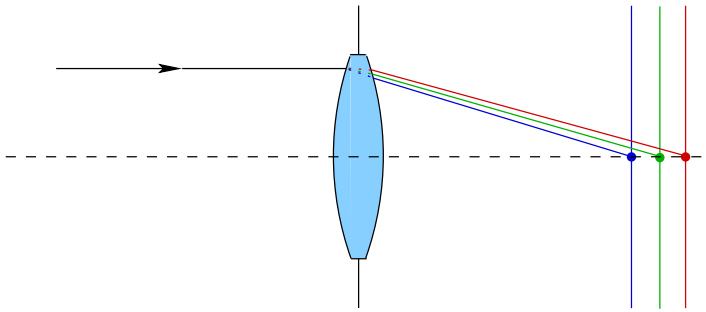


Figure 4.17: Chromatic aberration is caused by longer wavelengths traveling more quickly through the lens. The unfortunate result is that a different focal plane exists for each wavelength or color.

Chromatic aberration Recall from Section 4.1 that light energy is usually a jumble of waves with a spectrum of wavelengths. You have probably seen that the colors of the entire visible spectrum nicely separate when white light is shined through a prism. This is a beautiful phenomenon, but for lenses it is terrible annoyance because it separates the focused image based on color. This problem is called *chromatic aberration*.

The problem is that the speed of light through a medium depends on the wavelength. We therefore should write a material's refractive index as $n(\lambda)$ to indicate that it is a function of λ. Figure 4.17 shows the effect on a simple convex lens. The focal depth becomes a function of wavelength. If we shine red, green, and blue lasers directly into the lens along the same ray, then each color would cross the optical axis in a different place, resulting in red, green, and blue focal points. Recall the spectral power distribution and reflection functions from Section 4.1. For common light sources and materials, the light passing through a lens results in a whole continuum of focal points. Figure 4.18 shows an image with chromatic aberration artifacts. Chromatic aberration can be reduced at greater expense by combining convex and concave lenses of different materials so that the spreading rays are partly coerced into converging [136].

Spherical aberration Figure 4.19 shows *spherical aberration*, which is caused by rays further away from the lens center being refracted more than rays near the center. The result is similar to that of chromatic aberration, but this phenomenon is a *monochromatic* aberration because it is independent of the light wavelength. Incoming parallel rays are focused at varying depths, rather then being concentrated at a single point. The result is some blur that cannot be compensated by moving the object, lens, or image plane. Alternatively, the image might instead focus onto a curved surface, called the *Petzval surface*, rather then the image plane. This aberration arises due to the spherical shape of the lens. An *aspheric lens* is more complex and has non-spherical surfaces that are designed to specifically

Figure 4.18: The upper image is properly focused whereas the lower image suffers from chromatic aberration. (Figure by Stan Zurek.)

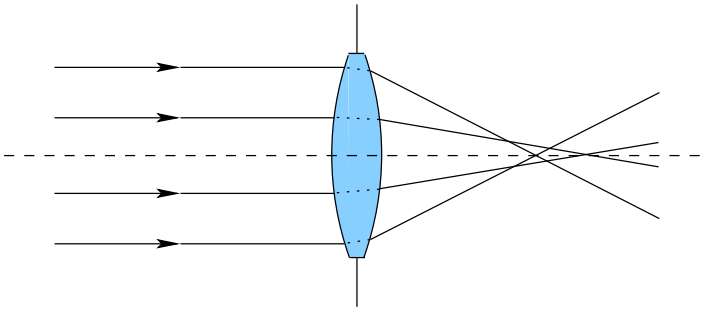


Figure 4.19: Spherical aberration causes imperfect focus because rays away from the optical axis are refracted more than those at the periphery.

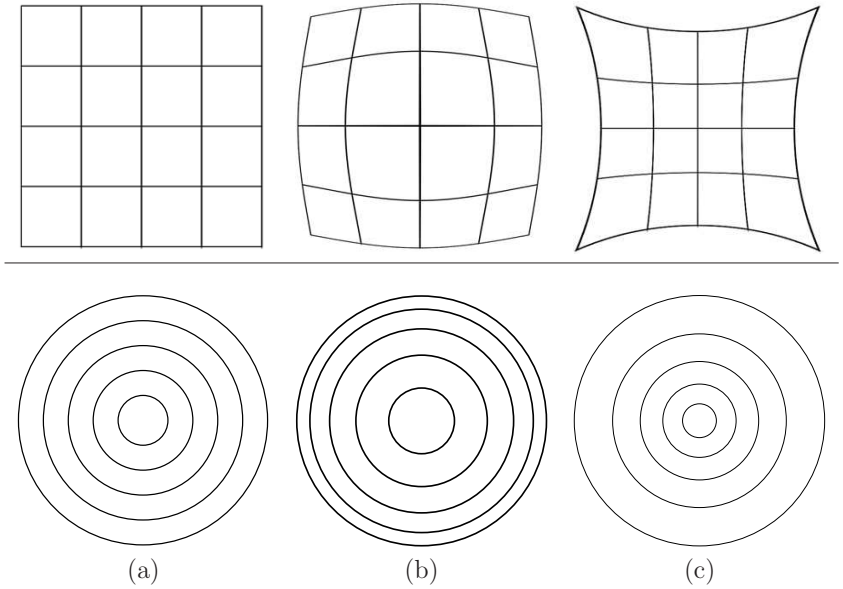


Figure 4.20: Common optical distortions. (a) Original images. (b) Barrel distortion. (c) Pincushion distortion. For the upper row, the grid becomes nonlinearly distorted. For lower row illustrates how circular symmetry is nevertheless maintained.

eliminate the spherical aberration and reduce other aberrations.

Optical distortion Even if the image itself projects onto the image plane it might be distorted at the periphery. Assuming that the lens is radially symmetric, the distortion can be described as a stretching or compression of the image that becomes increasingly severe away from the optical axis. Figure 4.20 shows how this effects the image for two opposite cases: *barrel distortion* and *pincushion distortion*. For lenses that have a wide field-of-view, the distortion is stronger, especially in the extreme case of a *fish-eyed lens*. Figure 4.21 shows an image that has strong barrel distortion. Correcting this distortion is an important component of VR headsets; otherwise, the virtual world would appear to be warped.

Astigmatism Figure 4.22 depicts *astigmatism*, which is a lens aberration that occurs for incoming rays that are not perpendicular to the lens. Up until now, our lens drawings have been 2D; however, a third dimension is needed to understand this new aberration. The rays can be off-axis in one dimension, but aligned in another. By moving the image plane along the optical axis, it becomes impossible to bring the image into focus. Instead, horizontal and vertical focal depths appear,

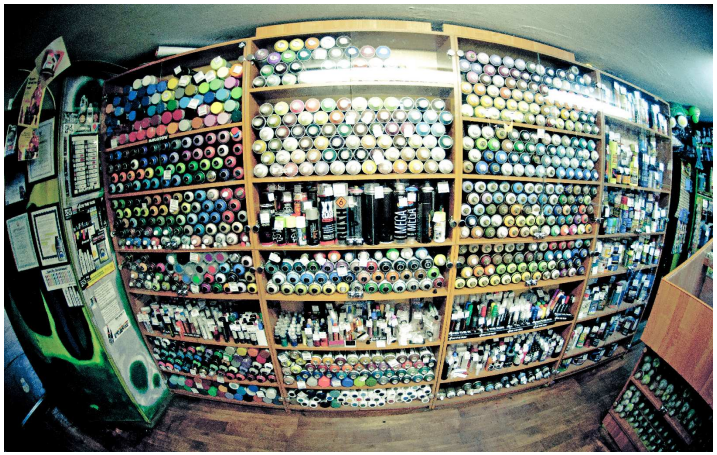


Figure 4.21: An image with barrel distortion, taken by a fish-eyed lens. (Image by Wikipedia user Ilveon.)

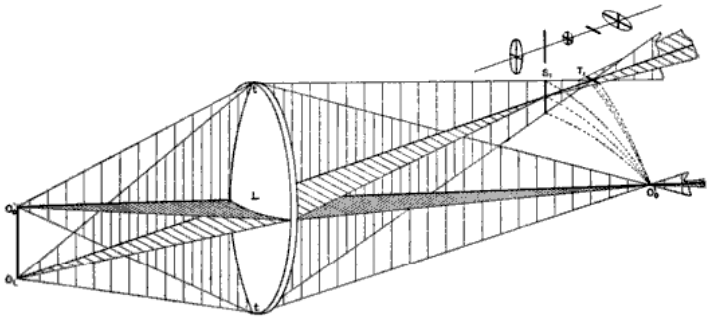


Figure 4.22: Astigmatism is primarily caused by incoming rays being off-axis in one plane, but close to perpendicular in another. (Figure from [160].)

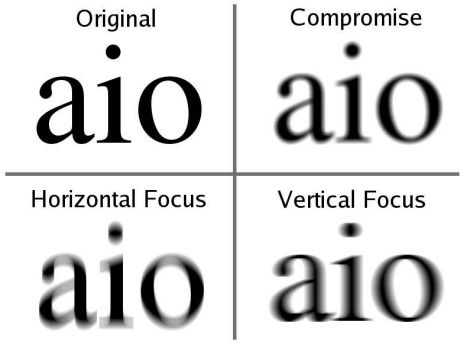


Figure 4.23: Due to astigmatism, it becomes impossible to bring the image perfectly into focus. At one depth, it might be focus horizontally, while at another it is focused vertically. We are forced to chose a compromise.

as shown in Figure 4.23.

Coma Finally, *coma* is yet another aberration. In this case, the image magnification varies dramatically as the rays are far from perpendicular to the lens. The result is a "comet" pattern in the image plane. You might have seen this while tilting a lens outside and observing bright disc patterns produced by direct sunlight. All of the aberrations of this section complicate the system or degrade the experience in a VR headset; therefore, substantial engineering effort is spent on mitigating these problems.

4.4 The Human Eye

We have covered enough concepts in this chapter to describe the basic operation of the human eye, which is clearly an important component in any VR system. Here it will be considered as part of an optical system of lenses and images. The physiological and perceptual parts of human vision are deferred until Chapter 5.

Figure 4.24 shows a cross section of the human eye facing left. Parallel light rays are shown entering from the left; compare to Figure 4.11, which showed a similar situation for an engineered convex lens. Although the eye operation is similar to the engineered setting, several important differences arise at this stage. The focal plane is replaced by a spherically curved surface called the *retina*. The retina contains *photoreceptors* that convert the light into neural pulses; this is covered in Sections 5.1 and 5.2. The interior of the eyeball is actually liquid, as opposed to air. The refractive indices of materials along the path from the outside air to the retina are shown in Figure 4.25.

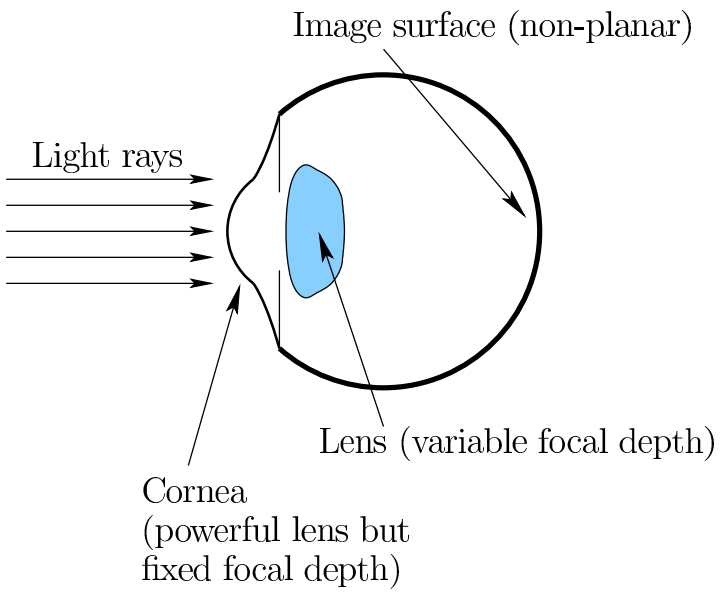
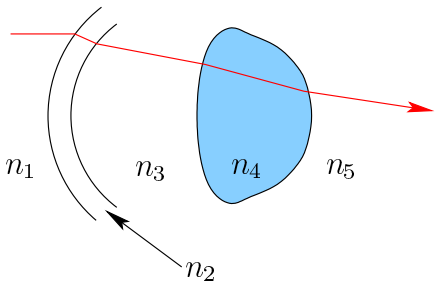


Figure 4.24: A simplified view of the human eye as an optical system.



1. $n_1 = 1.009$ (air)
2. $n_2 = 1.376$ (cornea)
3. $n_3 = 1.336$ (aqueous fluid)
4. $n_4 = 1.413$ (lens)
5. $n_5 = 1.337$ (vitreous fluid)

Figure 4.25: A ray of light travels through five media before hitting the retina. The indices of refraction are indicated. Considering Snell's law, the greatest bending occurs due to the transition from air to the cornea. Note that once the ray enters the eye, it passes through only liquid or solid materials.

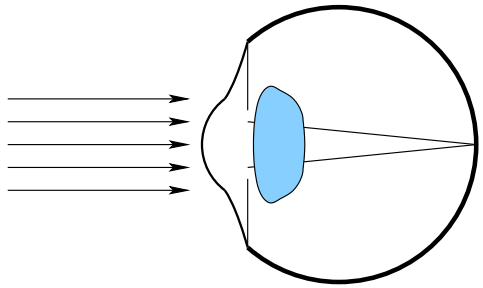


Figure 4.26: Normal eye operation, with relaxed lens.

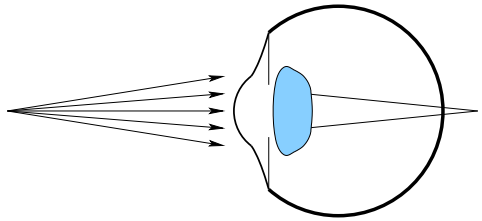


Figure 4.27: A closer object yields diverging rays, but with a relaxed lens, the image is blurry on the retina.

The optical power of the eye The outer diameter of the eyeball is roughly 24mm, which implies that a lens of at least 40D would be required to cause convergence of parallel rays onto the retina center inside of the eye (recall diopters from Section 4.2). There are effectively two convex lenses: The *cornea* and the *lens*. The cornea is the outermost part of the eye where the light first enters and has the greatest optical power, approximately 40D. The eye lens is less powerful and provides an additional 20D. By adding diopters, the combined power of the cornea and lens is 60D, which means that parallel rays are focused onto the retina at a distance of roughly 17mm from the outer cornea. Figure 4.26 shows how this system acts on parallel rays for a human with normal vision. Images of far away objects are thereby focused onto the retina.

Accommodation What happens when we want to focus on a nearby object, rather than one "infinitely far" away? Without any changes to the optical system, the image would be blurry on the retina, as shown in Figure 4.27. Fortunately, and miraculously, the lens changes its diopter to accommodate the closer distance. This process is appropriately called *accommodation*, as is depicted in Figure 4.28. The diopter change is effected through muscles that pull on the lens to change its shape. In young children, the lens can increase its power by an additional 15 to 20D, which explains why a child might hold something right in front of your face and expect you to focus on it; they can! At 20D, this corresponds to focusing on

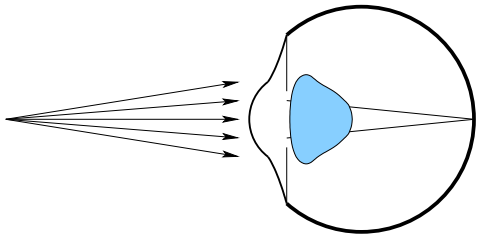


Figure 4.28: The process of accommodation: The eye muscles pull on the lens, causing it to increase the total optical power and focus the image on the retina.

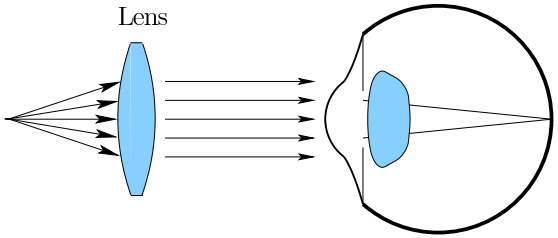


Figure 4.29: Placing a convex lens in front of the eye is another way to increase the optical power so that nearby objects can be brought into focus by the eye. This is the principle of reading glasses.

an object that is only 5cm from the cornea. Young adults already lose this ability and can accommodate up to about 10D. Thus, with normal vision they can read a book down to a distance of about 10cm (with some eye strain). Once adults reach 50 years old, little or no accommodation ability remains. This condition is called *presbyopia*. Figure 4.29 shows the most common treatment, which is to place reading glasses in front of the eye.

Vision abnormalities The situations presented so far represent normal vision throughout a person's lifetime. One problem could be that the optical system simply does not have enough optical power to converge parallel rays onto the retina. This condition is called *hyperopia* or *farsightedness*. Eyeglasses come to the rescue. The simple fix is to place a convex lens (positive diopter) in front of the eye, as in the case of reading glasses. In the opposite direction, some eyes have too much optical power. This case is called *myopia* or *nearsightedness*, and a concave lens (negative diopter) is placed in front of the eye to reduce the optical power appropriately. Recall that we have two eyes, not one. This allows the possibility for each eye to have a different problem, resulting in different lens diopters per eye. Other vision problems may exist beyond optical power. The most common is astigmatism, which was covered in Section 4.3. In human eyes this is caused by the eyeball having an excessively elliptical shape, rather than being perfectly spherical. Specialized, non-simple lenses are needed to correct this condition. You might also wonder whether the aberrations from Section 4.3 occur in the human eye. They do, however, the problems, such as chromatic aberration, are corrected automatically by our brains because we have learned to interpret such flawed images our entire lives.

A simple VR headset Now suppose we are constructing a VR headset by placing a screen very close to the eyes. Young adults would already be unable to bring it into focus it if were closer than 10cm. We want to bring it close so that it fills the view of the user. Therefore, the optical power is increased by using a convex lens, functioning in the same way as reading glasses. See Figure 4.30. This is also the process of magnification, from Section 4.2. The lens is placed at the distance of its focal depth. Using (4.5), this implies that $s_2 = -f$, resulting in $s_1 = \infty$. The screen appears as an enormous virtual image that is infinitely far away. Note, however, that a real image is nevertheless projected onto the retina. We do not perceive the world around us unless real images are formed on our retinas!

To account for people with vision problems, a focusing knob may be appear on the headset, which varies the distance between the lens and the screen. This adjusts the optical power so that the rays between the lens and the cornea are no longer parallel. They can be made to converge, which helps people with hyperopia. Alternatively, they can be made to diverge, which helps people with myopia. Thus, they can focus sharply on the screen without placing their eyeglasses in front of the lens. However, if each eye requires a different diopter, then a focusing knob

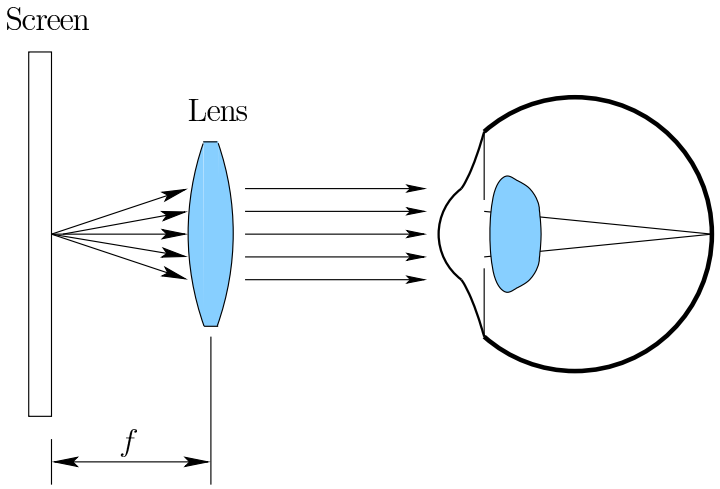


Figure 4.30: In VR headsets, the lens is placed so that the screen appears to be infinitely far away.

would be required for *each* eye. Furthermore, if they have an astigmatism, then it cannot be corrected. Placing eyeglasses inside of the headset may be the only remaining solution, but it may be uncomfortable and will reduce the field of view.

Many details have been skipped or dramatically simplified in this section. One important detail for a VR headset is each lens should be centered perfectly in front of the cornea. If the distance between the two lenses is permanently fixed, then this is impossible to achieve for everyone who uses the headset. The *interpupillary distance*, or IPD, is the distance between human eye centers. The average among humans is around 64mm, but it varies greatly by race, gender, and age (in the case of children). To be able to center the lenses for everyone, the distance between lens centers should be adjustable from around 55 to 75mm. This is a common range for binoculars. Unfortunately, the situation is not even this simple because our eyes also rotate within their sockets, which changes the position and orientation of the cornea with respect to the lens. This amplifies optical aberration problems that were covered in Section 4.3. Eye movements will be covered in Section 5.3. Another important detail is the fidelity of our vision: What pixel density is needed for the screen that is placed in front of our eyes so that we do not notice the pixels? A similar question is how many dots-per-inch (DPI) are needed on a printed piece of paper so that we do not see the dots, even when viewed under a magnifying glass? We return to this question in Section 5.1.

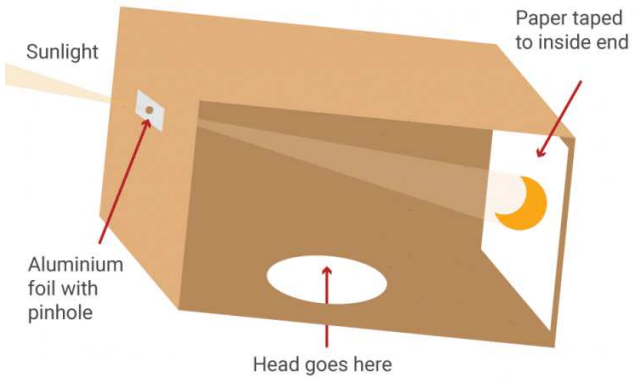
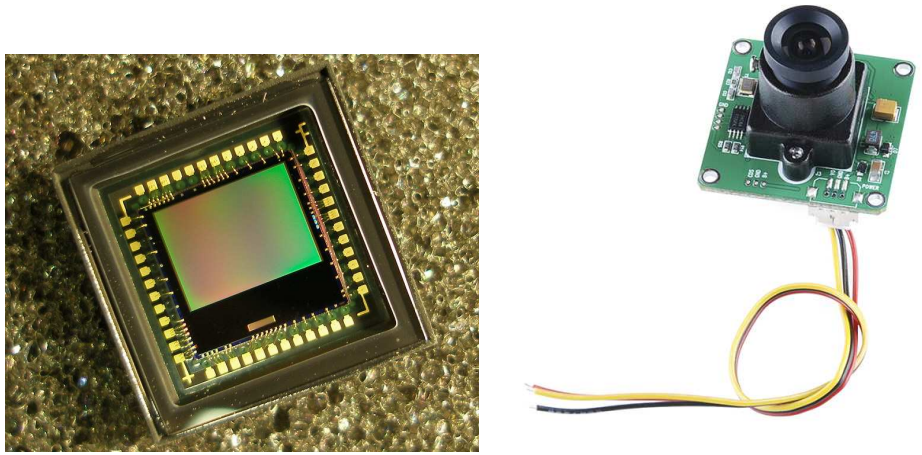


Figure 4.31: A pinhole camera that is recommended for viewing a solar eclipse. (Figure from TimeAndDate.com.)



(a) (b)

Figure 4.32: (a) A CMOS active-pixel image sensor. (b) A low-cost CMOS camera module (SEN-11745), ready for hobbyist projects.

4.5 Cameras

Now that we have covered the human eye, it seems natural to describe an engineered eye, otherwise known as a camera. People have built and used cameras for hundreds of years, starting with a *camera obscura* that allows light to pass through a *pinhole* and onto a surface that contains the real image. Figure 4.31 shows an example that you might have constructed to view a solar eclipse. (Recall the perspective transformation math from Section 3.4.) Eighteenth-century artists incorporated a mirror and tracing paper to un-invert the image and allow it to be perfectly copied. Across the 19th century, various chemically based technologies were developed to etch the image automatically from the photons hitting the imaging surface. Across the 20th century, film was in widespread use, until *digital cameras* avoided the etching process altogether by electronically capturing the image using a sensor. Two popular technologies have been a *Charge-Coupled Device* (*CCD*) array and a *CMOS active-pixel image sensor*, which is shown in Figure 4.32(a). Such digital technologies record the amount of light hitting each pixel location along the image, which directly produces a captured image. The costs of these devices has plummeted in recent years, allowing hobbyists to buy a camera module such as the one shown in Figure 4.32(b) for under $30 US.

Shutters Several practical issues arise when capturing digital images. The image is an 2D array of *pixels*, each of which having red (R), green (G), and blue (B) values that typically range from 0 to 255. Consider the total amount of light energy that hits the image plane. For a higher-resolution camera, there will generally be less photons per pixel because the pixels are smaller. Each sensing element (one per color per pixel) can be imagined as a bucket that collects photons, much like drops of rain. To control the amount of photons, a *shutter* blocks all the light, opens for a fixed interval of time, and then closes again. For a long interval (low *shutter speed*), more light is collected; however, the drawbacks are that moving objects in the scene will become blurry and that the sensing elements could become saturated with too much light. Photographers must strike a balance when determining the shutter speed to account for the amount of light in the scene, the sensitivity of the sensing elements, and the motion of the camera and objects in the scene.

Also relating to shutters, CMOS sensors unfortunately work by sending out the image information sequentially, line-by-line. The sensor is therefore coupled with a *rolling shutter*, which allows light to enter for each line, just before the information is sent. This means that the capture is not synchronized over the entire image, which leads to odd artifacts, such as the one shown in Figure 4.33. Image processing algorithms that work with rolling shutters and motion typically transform the image to correct for this problem. CCD sensors grab and send the entire image at once, resulting in a *global shutter*. Unfortunately, CCDs are more expensive than CMOS sensors, which has resulted in widespread appearance of rolling shutter cameras in smartphones.

Figure 4.33: The wings of a flying helicopter are apparently bent backwards due to the rolling shutter effect.

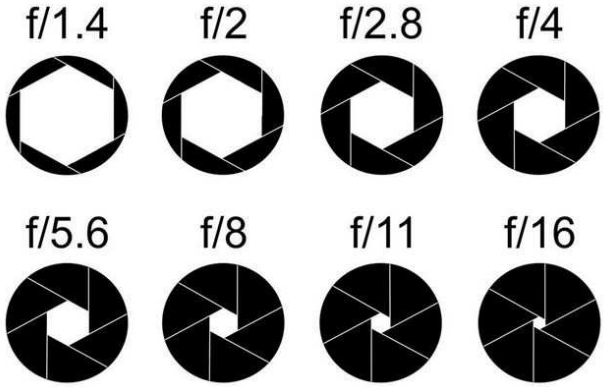


Figure 4.34: A spectrum of aperture settings, which control the amount of light that enters the lens. The values shown are called the *focal ratio* or *f-stop*.

Aperture The optical system also impacts the amount of light. Using a pinhole, as shown in Figure 4.31, light would fall onto the image sensor, but it would not be bright enough for most purposes (other than viewing a solar eclipse). Therefore, a convex lens is used instead so that multiple rays are converged to the same point in the image plane; recall Figure 4.11. This generates more photons per sensing element. The main drawback is that the lens sharply focuses objects at a single depth, while blurring others; recall (4.5). In the pinhole case, all depths are essentially "in focus", but there might not be enough light. Photographers therefore want to tune the optical system to behave more like a pinhole or more like a full lens, depending on the desired outcome. This result is a controllable *aperture* (Figure 4.34), which appears behind the lens and sets the size of the hole through which the light rays enter. A small radius mimics a pinhole by blocking all but the center of the lens. A large radius allows light to pass through the entire lens. Our eyes control the light levels in a similar manner by contracting or dilating our pupils.. Finally, note that the larger the aperture, the more that the aberrations covered in Section 4.3 affect the imaging process.

Further Reading

A classic, popular text on optical engineering: [136].

Chapter 5

The Physiology of Human Vision

Chapter Status	Taken from *Virtual Reality*, S. M. LaValle
	This online chapter is not the final version! Check **http://vr.cs.uiuc.edu/** for information on the latest draft version. This draft was compiled on June 29, 2016.

What you perceive about the world around you is "all in your head". After reading Chapter 4, especially Section 4.4, you should understand that the light around us forms images on our retinas that capture colors, motions, and spatial relationships in the physical world. For someone with normal vision, these captured images may appear to have perfect clarity, speed, accuracy, and resolution, while being distributed over a large field of view. However, we are being fooled. We will see in this chapter that this apparent perfection of our vision is mostly an illusion because neural structures are filling in plausible details to generate a coherent picture in our heads that is consistent with our life experiences. When building VR technology that co-opts these processes, it important to understand how they work. They were designed to do more with less, and fooling these processes with VR produces many unexpected side effects because the display technology is not a perfect replica of the surrounding world.

Section 5.1 continues where Section 4.4 left off by adding some biology of the human eye to the optical system. Most of the section is on photoreceptors, which are the "input pixels" that get paired with the "output pixels" of a digital display for VR. Section 5.2 offers a taste of neuroscience by explaining what is known about the visual information that hierarchically propagates from the photoreceptors up to the visual cortex. Section 5.3 explains how our eyes move, which serves a good purpose, but incessantly interferes with the images in our retinas. Section 5.4 concludes the chapter by applying the knowledge gained about visual physiology to determine VR display requirements, such as the screen resolution.

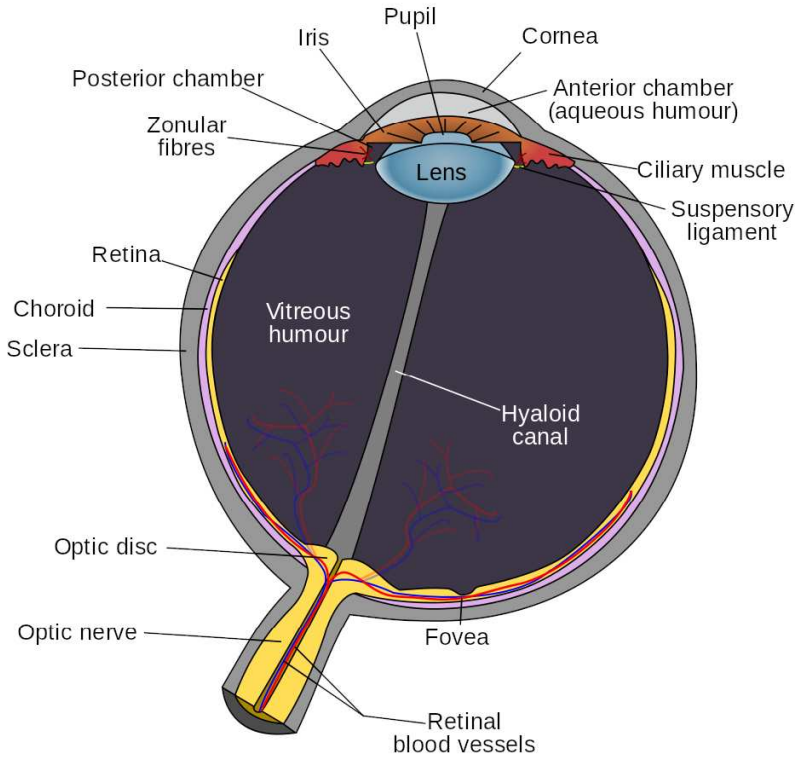


Figure 5.1: Physiology of the human eye. This viewpoint shows how the right eye would appear if sliced horizontally (the nose would be to the left). (From Wikipedia user Rhcastilhos.)

5.1 From the Cornea to Photoreceptors

Parts of the eye Figure 5.1 shows the physiology of a human eye. The shape is approximately spherical, with a diameter of around 24mm and only slight variation among people. The *cornea* is a hard, transparent surface through which light enters and provides the greatest optical power (recall from Section 4.4). The rest of the outer surface of the eye is protected by a hard, white layer called the *sclera*. Most of the eye interior consists of *vitreous humor*, which is a transparent, gelatinous mass that allows light rays to penetrate with little distortion or attenuation.

As light rays cross the cornea, they pass through a small chamber containing *aqueous humour*, which is another transparent, gelatinous mass. After crossing this, rays enter the *lens* by passing through the *pupil*. The size of the pupil is controlled by a disc-shaped structure called the *iris*, which provides an aperture that regulates the amount of light that is allowed to pass. The optical power of the lens is altered by *ciliary muscles*. After passing through the lens, rays pass through

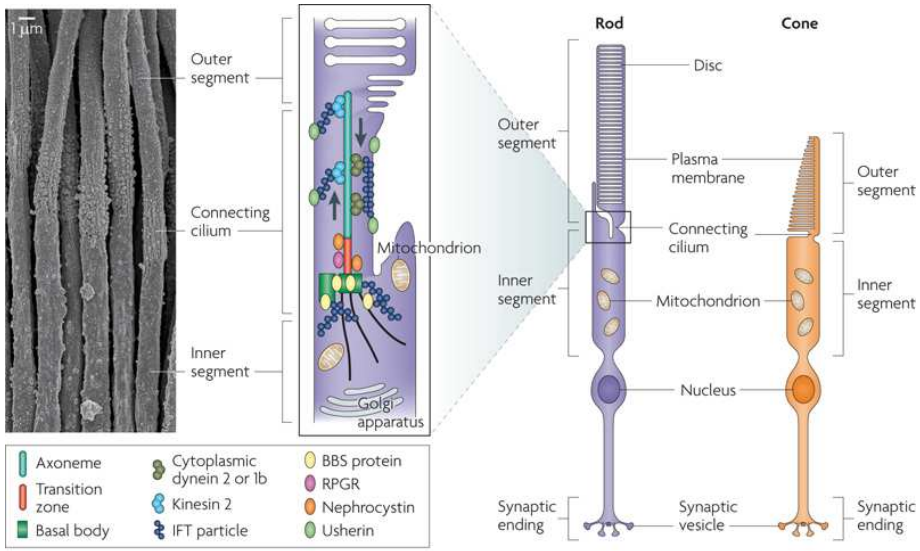


Figure 5.2: On the left is an electron micrograph image of photoreceptors. The right shows the structure and components of rods and cones. The outer segments contain photopigments that electrochemically respond when bombarded by photons. (Figure from [159].)

the vitreous humor and strike the *retina* which lines more than 180° of the inner eye boundary. Since Figure 5.1 shows a 2D cross section, the retina looks more like an arc; however, keep in mind that it is a 2D surface. Imagine it as curved counterpart to a visual display. To catch the light from the output pixels, it is lined with *photoreceptors*, which behave like "input pixels". The most important part of the retina is the *fovea*; the highest *visual acuity*, which is a measure of the sharpness or clarity of vision, is provided for rays that land on it. The *optic disc* is a small hole in the retina through which neural pulses are transmitted outside of the eye through the *optic nerve*. It is on the same side of the fovea as the nose.

Photoreceptors The retina contains two kinds of photoreceptors for vision: 1) *rods*, which are triggered by very low levels of light, and 2) *cones*, which require more light and are designed to distinguish between colors. See Figure 5.2. To understand the scale, the width of the smallest cones is around 1000nm. This is quite close to the wavelength of visible light, implying that photoreceptors need not be much smaller. Each human retina contains about 120 million rods and 6 million cones that are densely packed along the retina. Figure 5.3 shows the detection capabilities of each photoreceptor type. Rod sensitivity peaks at 498nm, between blue and green in the spectrum. There are three categories of cones, based on whether they are designed to sense blue, green, or red light.

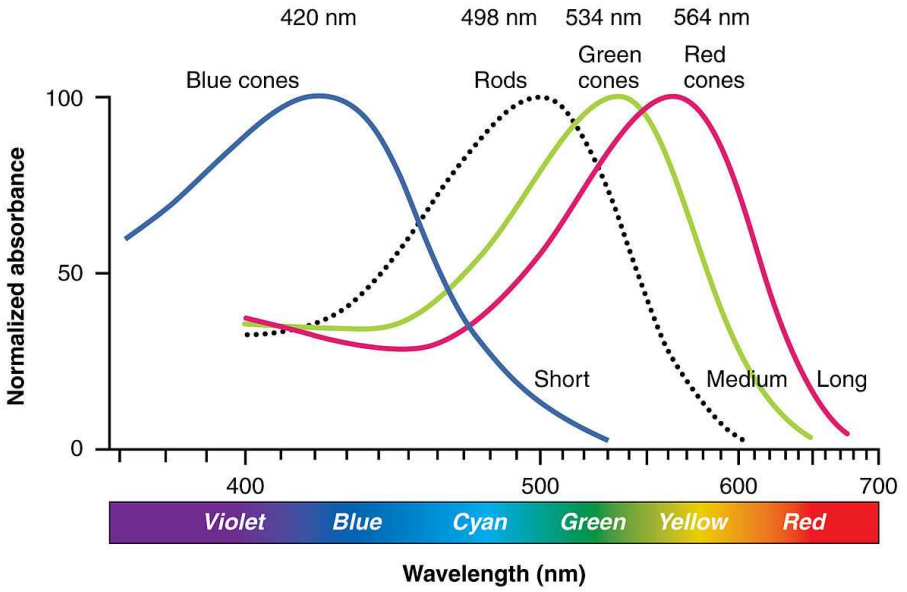


Figure 5.3: The sensitivity of rods and cones as a function of wavelength [18]. (Figure adapted by OpenStax College.)

Light source	**Luminance** (cd/m^2)	**Photons per receptor**
Paper in starlight	0.0003	0.01
Paper in moonlight	0.2	1
Computer monitor	63	100
Room light	316	1000
Blue sky	2500	10,000
Paper in sunlight	40,000	100,000

Figure 5.4: Several familiar settings and the approximate number of photons per second hitting a photoreceptor. (Figure adapted from [74, 92].)

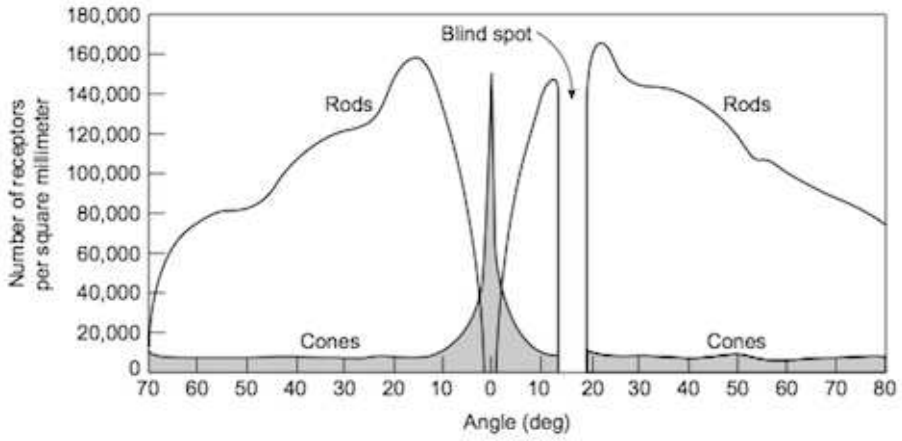


Figure 5.5: Photoreceptor density as a function of angle. The right of the plot is the nasal side (which corresponds to rays entering from the opposite, temporal side). (Figure based on [110])

Photoreceptors respond to light levels over a large dynamic range. Figure 5.4 shows several familiar examples. The luminance is measured in SI units of candelas per square meter, which corresponds directly to the amount of light power per area. The range spans seven orders of magnitude, from 1 photon hitting a photoreceptor every 100 seconds up to 100,000 photons per receptor per second. At low light levels, only rods are triggered. Our inability to distinguish colors at night is caused by the inability of rods to distinguish colors. Our eyes may take up to 35 minutes to fully adapt to low light, resulting in a monochromatic mode called *scotopic vision*. By contrast, our cones become active in brighter light. Adaptation to this trichromatic mode, called *photopic vision*, may take up to ten minutes (you have undoubtedly noticed the adjustment period when someone unexpectedly turns on lights while you are lying in bed at night).

Photoreceptor density The density of photoreceptors across the retina varies greatly, as plotted in Figure 5.5. The most interesting region is the *fovea*, which has the greatest concentration of photoreceptors. The innermost part of the fovea has a diameter of only 0.5mm or an angular range of ± 0.85 degrees, and contains almost entirely cones. This implies that the eye must be pointed straight at a target to perceive a sharp, colored image. The entire fovea has diameter 1.5mm (± 2.6 degrees angular range), with the outer ring having a dominant concentration of rods. Rays that enter the cornea from the sides land on parts of the retina with lower rod density and very low cone density. This corresponds to the case of *peripheral vision*. We are much better at detecting movement in our periphery, but cannot distinguish colors effectively. Peripheral movement detection may have

Figure 5.6: An experiment that reveals your blind spot. Close your right eye and look directly at the "X". Vary the distance of the paper (or screen) from your eye. Over some range, the dot should appear to vanish. You can carry this experiment one step further by writing an "X" and dot on a textured surface, such as graph paper. In that case, the dot disappears and you might notice the surface texture perfectly repeating in the place where the dot once existed. This is caused by your brain filling in the expected texture over the blind spot!

helped our ancestors from being eaten by predators. Finally, the most intriguing part of the plot is the *blind spot*, where there are no photoreceptors. This is due to our retinas being inside-out and having no other way to route the neural signals to the brain; see Section 5.2.

The photoreceptor densities shown in Figure 5.5 leave us with a conundrum. With 20/20 vision, we perceive the world as if our eyes are capturing a sharp, colorful image over a huge angular range. This seems impossible, however, because we can only sense sharp, colored images in a narrow range. Furthermore, the blind spot should place a black hole in our image. Surprisingly, our *perceptual* processes produce an illusion that a complete image is being captured. This is accomplished by filling in the missing details using contextual information, which is described in Section 5.2, and by frequent eye movements, the subject of Section 5.3. If you are still not convinced that your brain is fooling you into seeing a complete image, try the blind spot experiment shown in Figure 5.6.

5.2 From Photoreceptors to the Visual Cortex

Photoreceptors are transducers that convert the light-energy stimulus into an electrical signal called a neural impulse, thereby inserting information about the outside world into our neural structures. Recall from Section 2.3 that signals are propagated upward in a hierarchical manner, from photoreceptors to the visual cortex (Figure 2.19). Think about the influence that each photoreceptor has on the network of neurons. Figure 5.7 shows a simplified model. As the levels increase, the number of influenced neurons grows rapidly. Figure 5.8 shows the same diagram, but highlighted in a different way by showing how the number of photoreceptors that influence a single neuron increases with level. Neurons at the lowest levels are able to make simple comparisons of signals from neighboring photoreceptors. As the levels increase, the neurons may response to a larger patch of the retinal image. This principle will become clear when seeing more neural structures in this section. Eventually, when signals reach the highest levels (beyond these figures), information from the memory of a lifetime of experiences is fused with the information that propagated up from photoreceptors. As the brain

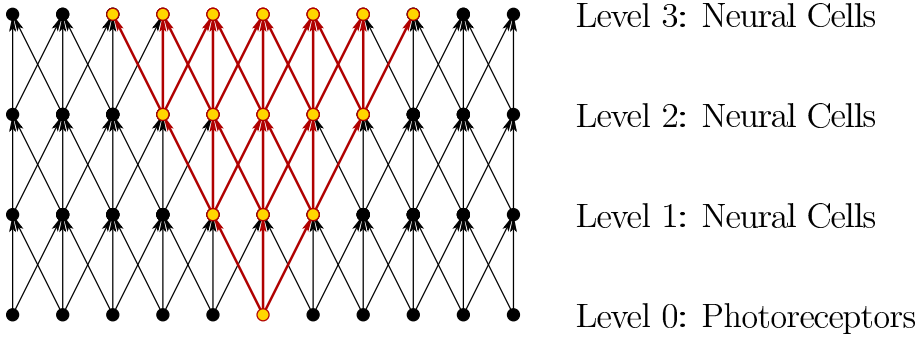


Figure 5.7: Four levels in a simple hierarchy are shown. Each disk corresponds to a neural cell or photoreceptor, and the arrows indicate the flow of information. Photoreceptors generate information at Level 0. In this extremely simplified and idealized view, each photoreceptor and neuron connects to exactly three others at the next level. The red and gold part highlights the growing zone of influence that a single photoreceptor can have as the levels increase.

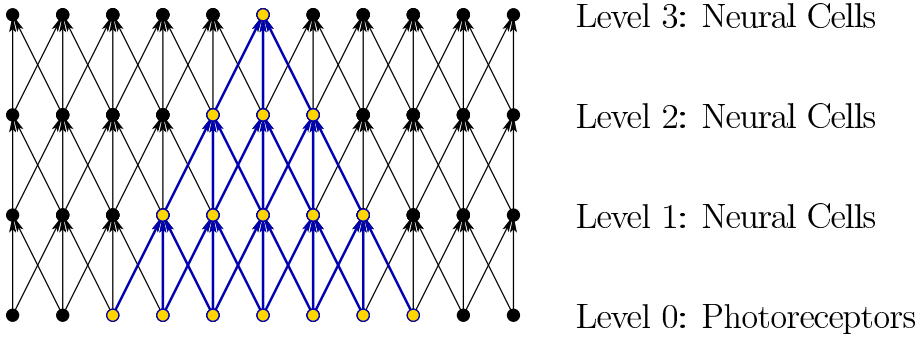


Figure 5.8: This diagram is the same as Figure 5.7 except that the information feeding into a single neuron is highlighted. Consider the set of photoreceptors involved in the reaction of a single neural cell. This is called the *receptive field*. As the level increases, the receptive field size grows dramatically. Due to the spatial arrangement of the photoreceptors, this will imply that each neuron responds to a growing patch in the image on the retina. The patch increases in size at higher levels.

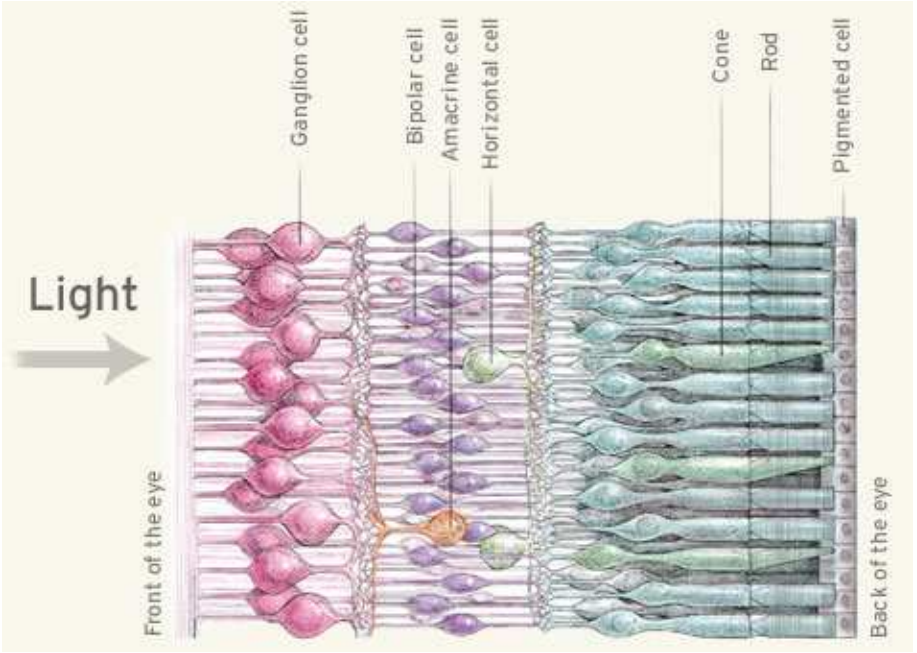


Figure 5.9: Light passes through a few neural layers before hitting the rods and cones. (Figure by the Institute for Dynamic Educational Advancement.)

performs significant perceptual processing, a perceptual phenomenon results, such as recognizing a face or judging the size of a tree. It takes the brain over 100ms to produce a result that enters our consciousness.

Now consider the first layers of neurons in more detail, as shown in Figure 5.9. The information is sent from right to left, passing from the rods and cones to the bipolar, amacrine, and horizontal cells. These three types of cells are in the *inner nuclear layer*. From there, the signals reach the ganglion cells, which form the *ganglion cell layer*. Note that the light appears to be entering from the wrong direction: It passes over these neural cells before reaching the photoreceptors. This is due to the fact that the human retina is inside-out, as shown in Figure 5.10. Evolution got it right with octopuses and other cephalopods, for which the light directly reaches the photoreceptors. One consequence of an inside-out retina is that the axons of the ganglion cells cannot be directly connected to the *optic nerve* (item 3 in Figure 5.10), which sends the signals outside of the eye. Therefore, a hole has been punctured in our retinas so that the "cables" from the ganglion cells can be routed outside of the eye (item 4 in Figure 5.10). This causes the blind spot that was illustrated in Figure 5.6.

Upon studying Figure 5.9 closely, it becomes clear that the neural cells are not arranged in the ideal way of Figure 5.8. The *bipolar cells* transmit signals from the

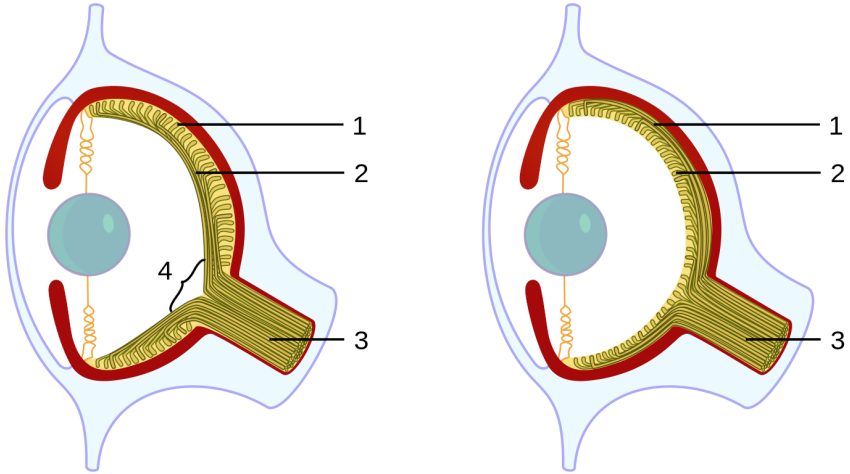


Figure 5.10: Vertebrates (including humans) have inside-out retinas, which lead to a blind spot and photoreceptors aimed away from the incoming light. The left shows a vertebrate eye, and the right shows a cephalopod eye, for which nature got it right: The photoreceptors face the light and there is no blind spot. (Figure by Jerry Crimson Mann.)

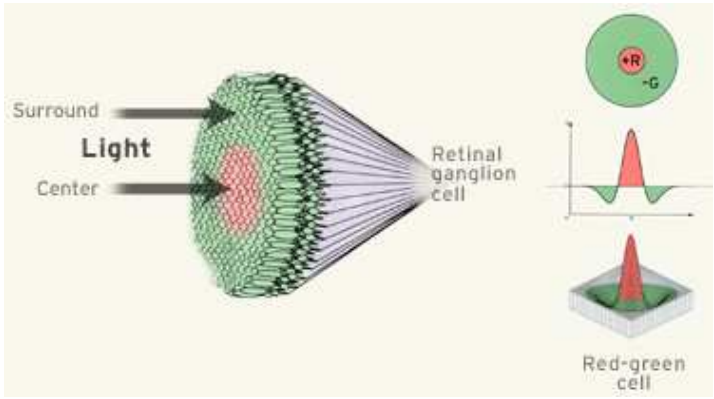


Figure 5.11: The receptive field of an ON-center ganglion cell. (Figure by the Institute for Dynamic Educational Advancement.)

photoreceptors to the ganglion cells. Some bipolars connect only to cones, with the number being between 1 and 10 per bipolar. Others connect only to rods, with about 30 to 50 rods per bipolar. There are two types of bipolar cells based on their function. An *ON bipolar* activates when the rate of photon absorption in its connected photoreceptors *increases*. An *OFF bipolar* activates for *decreasing* photon absorption. The bipolars connected to cones have both kinds; however, the bipolars for rods have only ON bipolars. The bipolar connections are considered to be *vertical* because they connect directly from photoreceptors to the ganglion cells This is in contrast to the remaining two cell types in the inner nuclear layer. The *horizontal cells* are connected by inputs (dendrites) to photoreceptors and bipolar cells within a radius of up to 1mm. Their output (axon) is fed into photoreceptors, causing *lateral inhibition*, which means that the activation of one photoreceptor tends to decrease the activation of its neighbors. Finally, *amacrine cells* connect horizontally between bipolar cells, other amacrine cells, and vertically to ganglion cells. There are dozens of types, and their function is not well understood. Thus, scientists do not have a complete understanding of human vision, even at the lowest layers. Nevertheless, the well understood parts contribute greatly to our ability to design effective VR systems and predict other human responses to visual stimuli.

At the ganglion cell layer, several kinds of cells process portions of the retinal image. Each ganglion cell has a large receptive field, which corresponds to the photoreceptors that contribute to its activation as shown in Figure 5.8. The three most common and well understood types of ganglion cells are called *midget*, *parasol*, and *bistratified*. They perform simple filtering operations over their receptive fields based on spatial, temporal, and spectral (color) variations in the stimulus across the photoreceptors. Figure 5.11 shows one example. In this case, a ganglion cell is triggered when red is detected in the center but not green in the surrounding

area. This condition is an example of *spatial opponency*, for which neural structures are designed to detect local image variations. Thus, consider ganglion cells as tiny image processing units that can pick out local changes in time, space, and/or color. They can detect and emphasize simple image features such as edges. Once the ganglion axons leave the eye through the optic nerve, a significant amount of image processing has already been performed to aid in visual perception. The raw image based purely on photons hitting the photoreceptor never leaves the eye.

The optic nerve connects to a part of the *thalamus* called the *lateral geniculate nucleus* (*LGN*); see Figure 5.12. The LGN mainly serves as a router that sends signals from the senses to the brain, but also performs some processing. The LGN sends image information to the *primary visual cortex* (V1), which is located at the back of the brain. The *visual cortex*, highlighted in Figure 5.13, contains several interconnected areas that each perform specialized functions. Figure 5.14 shows one well-studied operation performed by the visual cortex. Chapter 6 will describe visual perception, which is the conscious result of processing in the visual cortex, based on neural circuitry, stimulation of the retinas, information from other senses, and expectations based on prior experiences. Characterizing how all of these processes function and integrate together remains an active field of research.

5.3 Eye Movements

Eye rotations are a complicated and integral part of human vision. They occur both voluntarily and involuntarily, and allow humans to fixate on features in the world, even as the head or target features are moving. One of the main reasons for movement is to position the feature of interest on the fovea. Recall from Section 5.2 that only the fovea can sense dense, color images, and it unfortunately spans a very narrow field of view. To gain a coherent, detailed view of a large object, the eyes rapidly scan over it while fixating on points of interest. Figure 5.15 shows an example. Another reason for eye movement is that our photoreceptors are slow to respond to stimuli due to their chemical nature. They take up to 10ms to fully respond to stimuli and produce a response for up to 100ms. Eye movements help keep the image fixed on the same set of photoreceptors so that they can fully charge. This is similar to the image blurring problem that occurs in cameras at low light levels and slow shutter speeds. Additional reasons for eye movement are to maintain a stereoscopic view and to prevent adaptation to a constant stimulation. To support the last claim, it has been shown experimentally that when eye motions are completely suppressed, visual perception disappears completely [58]. As movements combine to build a coherent view, it is difficult for scientists to predict and explain how we will interpret some stimuli. For example, the optical illusion in Figure 5.16 appears to be moving when our eyes scan over it.

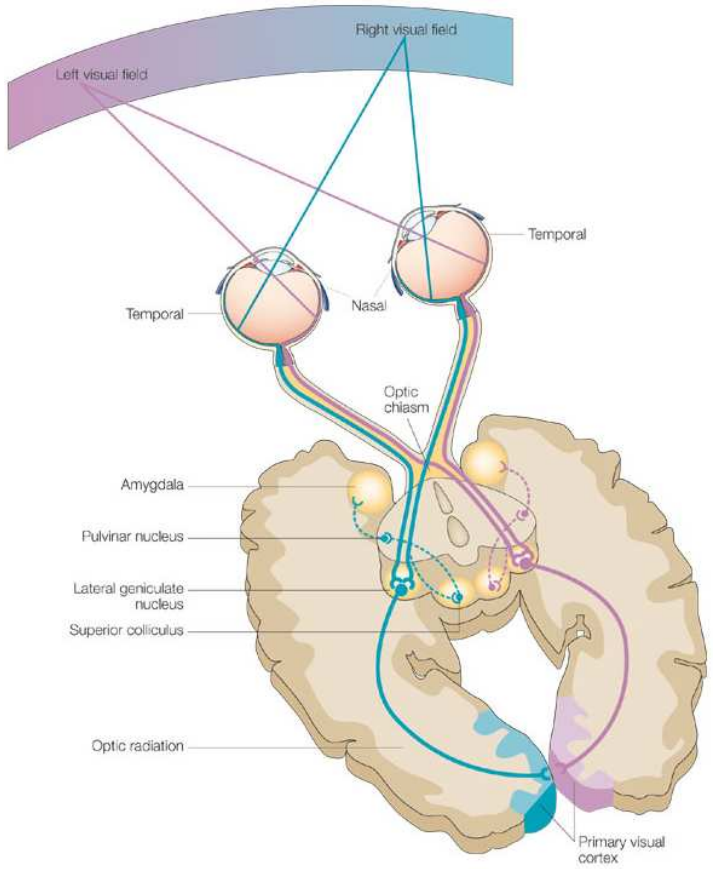


Figure 5.12: The visual pathway from the eyes to the LGN to the visual cortex. Note that information from the right and left sides of the visual field becomes swapped in the cortex. (Figure from Nature Reviews: Neuroscience)

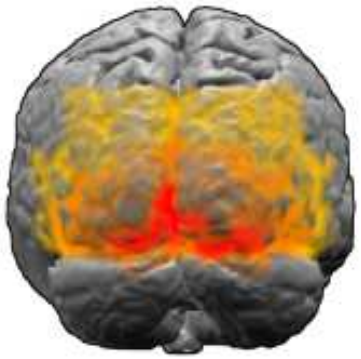


Figure 5.13: The visual cortex is located in the back of the head (Figure by Washington Irving).

Eye muscles The rotation of each eye is controlled by six muscles that are each attached to the sclera (outer eyeball surface) by a tendon. Figures 5.17 and 5.18 show their names and arrangement. The tendons pull on the eye in opposite pairs. For example, to perform a yaw (side-to-side) rotation, the tensions on the medial rectus and lateral rectus are varied while the other muscles are largely unaffected. To cause a pitch motion, four muscles per eye become involved. All six are involved to perform both a pitch and yaw, for example, looking upward and to the right. A small amount of roll can be generated; however, our eyes are generally not designed for much roll motion. Imagine if you could turn your eyeballs upside-down inside of their sockets! Thus, it is reasonable in most cases to approximate eye rotations as a 2D set that includes only yaw and pitch, rather than the full 3 DOFs obtained for rigid body rotations in Section 3.2.

Types of movements We now consider movements based on their purpose, resulting in six categories: 1) saccades, 2) smooth pursuit, 3) vestibulo-ocular reflex, 4) optokinetic reflex, 5) vergence, and 6) microsaccades. All of motions cause both eyes to rotate approximately the same way, except for vergence, which causes the eyes to rotate in opposite directions. We will skip a seventh category of motion, called *rapid eye movements* (REMs), because they only occur while we are sleeping and therefore do not contribute to a VR experience. The remaining six categories will now be discussed in detail.

Saccades The eye can move in a rapid motion called a *saccade*, which lasts less than 45ms and rotates at a rate of about 900° per second. The purpose is to quickly relocate the fovea so that important features in a scene are sensed with highest visual acuity. Figure 5.15 showed an example in which a face is scanned by

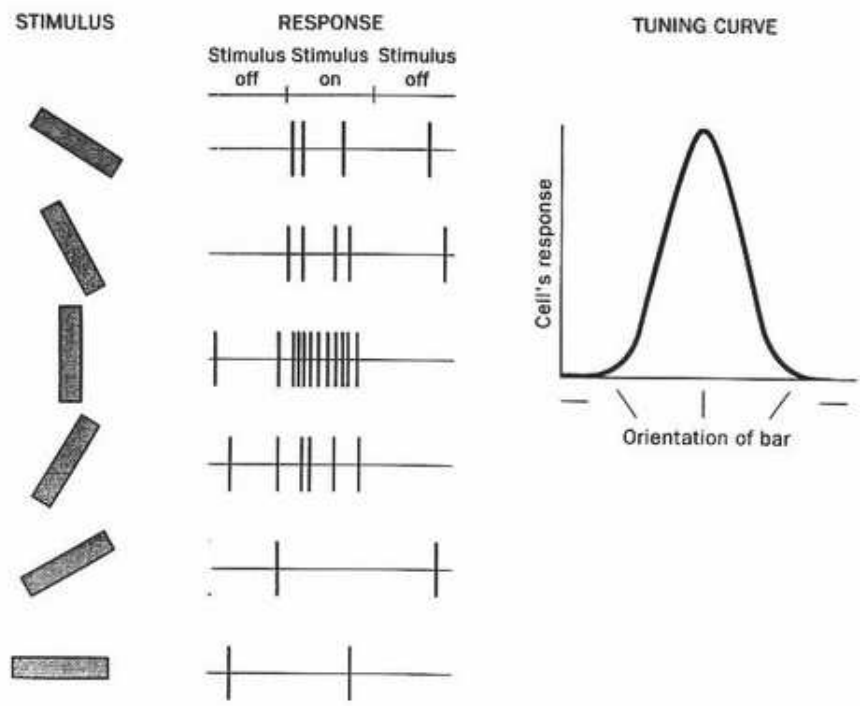


Figure 5.14: A popular example of visual cortex function is *orientation tuning*, in which a single-unit recording is made of a single neuron in the cortex. As the bar is rotated in front of the eye, the response of the neuron varies. It strongly favors one particular orientation.

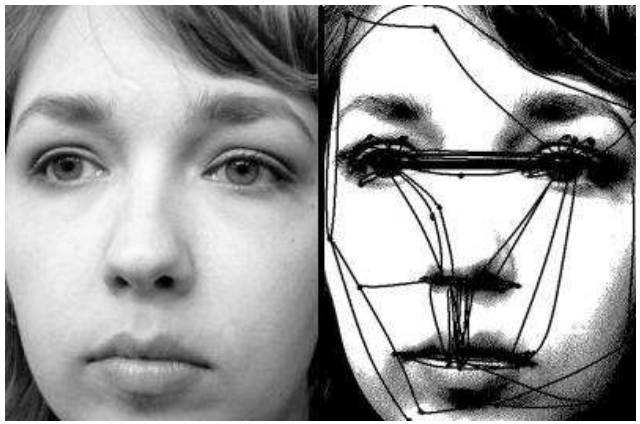


Figure 5.15: The trace of scanning a face using saccades.

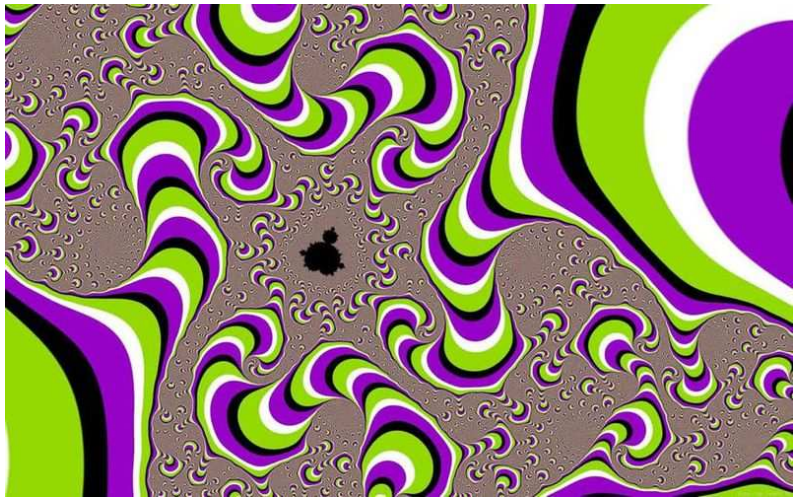


Figure 5.16: The fractal appears to be moving until you carefully fixate on a single part to verify that it is not.

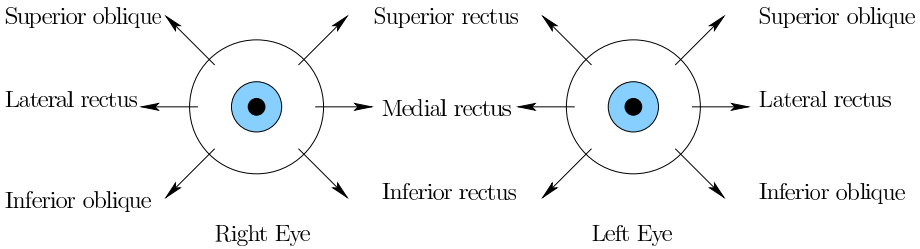


Figure 5.17: There are six muscles per eye, each of which is capable of pulling the pupil toward its location.

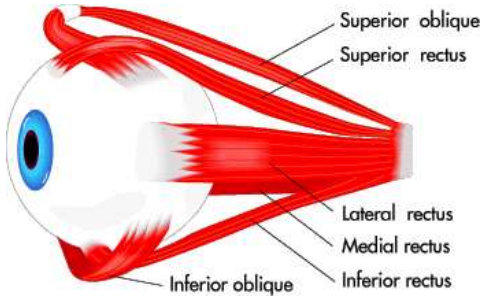


Figure 5.18: The six muscle tendons attach to the eye so that yaw, pitch, and a small amount of roll become possible.

fixating on various features in rapid succession. Each transition between features is accomplished by a saccade. Interestingly, our brains use *saccadic masking* to hide the intervals of time over which saccades occur from our memory. This results in distorted time perception, as in the case when second hands click into position on an analog clock. The result of saccades is that we obtain the illusion of high acuity over a large angular range. Although saccades frequently occur while we have little or no awareness of them, we have the ability to consciously control them as we choose features for fixation.

Smooth pursuit In the case of *smooth pursuit*, the eye slowly rotates to track a moving target feature. Examples are a car, a tennis ball, or a person walking by. The rate of rotation usually less than 30° per second, which is much slower than for saccades. The main function of smooth pursuit is to reduce motion blur on the retina; this is also known as *image stabilization*. The blur is due to the slow response time of photoreceptors, as discussed in Section 5.1. If the target is moving too fast, then saccades may be intermittently inserted into the pursuit motions to catch up to it.

Vestibulo-ocular reflex One of the most important motions to understand for VR is the *vestibulo-ocular reflex* or *VOR*. Hold your finger at a comfortable distance in front of your face. Next, yaw your head back and forth (like you are nodding "no"), turning about 20 or 30 degrees to the left and right sides each time. You may notice that your eyes are effortlessly rotating to counteract the rotation of your head so that your finger remains in view. The eye motion is involuntary. If you do not believe it, then try to avoid rotating your eyes while paying attention to your finger and rotating your head. It is called a reflex because the motion control bypasses higher brain functions. Figure 5.19 shows how this circuitry works. Based on angular accelerations sensed by our vesibular organs, signales are sent to the eye muscles to provide the appropriate counter motion. The main purpose of the VOR is to provide image stabilization, as in the case of smooth pursuit. For more details about the vestibular organ, see Section 8.2.

Optokinetic reflex The next category is called the *optokinetic reflex*, which occurs when a fast object speeds along. This occurs when watching a fast-moving train while standing nearby on fixed ground. The eyes rapidly and involuntarily choose features for tracking on the object, while alternating between smooth pursuit and saccade motions.

Vergence *Stereopsis* refers to the case in which both eyes are fixated on the same object, resulting in a single perceived image. Two kinds of *vergence* motions occur to align the eyes with an object. See Figure 5.20. If the object is closer than a previous fixation, then a *convergence* motion occurs. This means that the eyes are rotating so that the pupils are becoming closer. If the object is further, then *divergence* motion occurs, which causes the pupils to move further apart. The eye

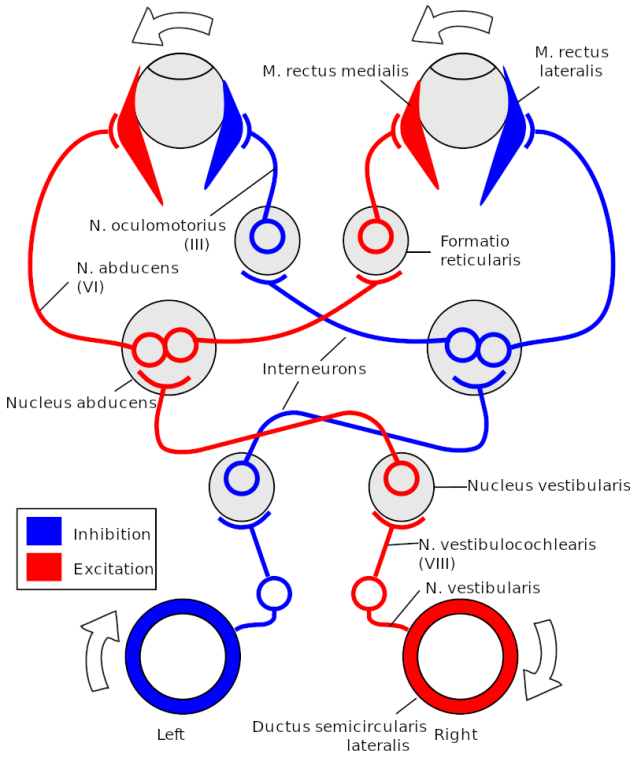


Figure 5.19: The vestibulo-oculus reflex (VOR). The eye muscles are wired to angular accelerometers in the vestibular organ to counter head movement with the opposite eye movement with less than 10ms of latency. The connection between the eyes and vestibular organ is provided by specialized vestibular and extraocular motor nuclei, thereby bypassing higher brain functions.

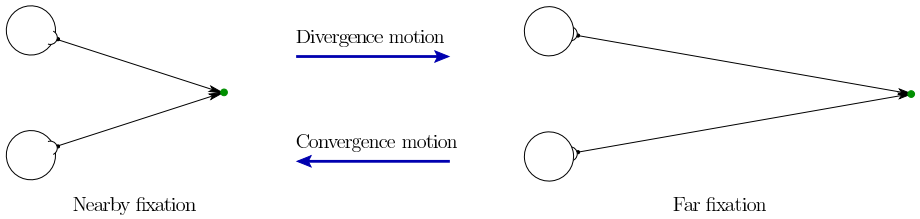


Figure 5.20: In the process of stereopsis, both eyes are fixated on the same feature in the world. To transition from a close to far feature, a divergence motion occurs. A convergence motion happens for the opposite transition.

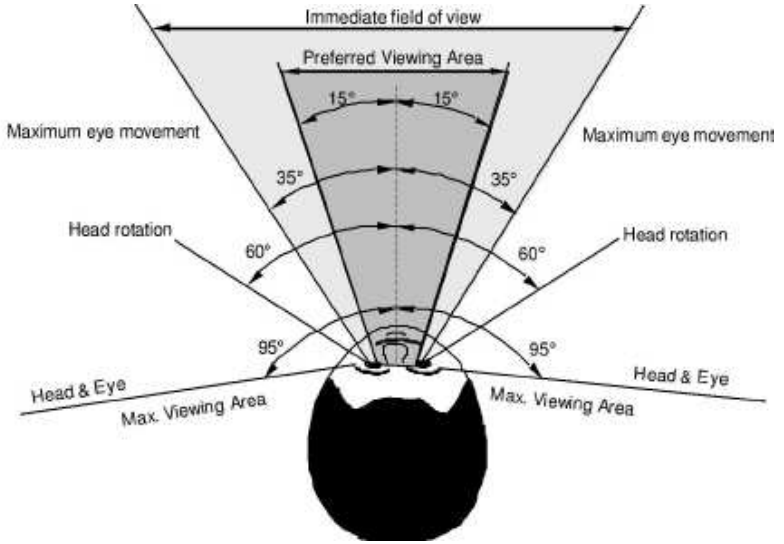


Figure 5.21: The head and eyes rotate together to fixate on moving or new targets.

orientations resulting from vergence motions provide important information about the distance of objects.

Microsaccades The sixth category of movements is called *microsaccades*, which are small, involuntary jerks of less than one degree that trace out an erratic path. They are believed to augment many other processes, including control of fixations, reduction of perceptual fading due to adapatation, improvement of visual acuity, and resolving perceptual ambiguities [121]. Although these motions have been known since the 18th century [32], their behavior is extremely complex and not fully understood. Microsaccades are an active topic of research in perceptual psychology, biology, and neuroscience.

Eye and head movements together Although this section has focused on eye movement, it is important to understand that most of the time the eyes and head are moving together. Figure 5.21 shows the angular range for yaw rotations of the head and eyes. Although eye yaw is symmetric by allowing 35° to the left or right, pitching of the eyes is not. Human eyes can pitch 20° upward and 25° downward, which suggests that it might be optimal to center a VR display slightly below the pupils when the eyes are looking directly forward. In the case of VOR, eye rotation is controlled to counteract head motion. In the case of smooth pursuit, the head and eyes may move together to keep a moving target in the preferred viewing area.

5.4 Implications for VR

This chapter has so far covered the human hardware for vision. Basic physiological properties, such as photoreceptor density or VOR circuitry directly impact the engineering requirements for visual display hardware. The engineered systems must be good enough to adequately fool our senses, but they need not have levels of quality that are well beyond the limits of our receptors. Thus, the VR display should ideally be designed to perfectly match the performance of the sense it is trying to fool.

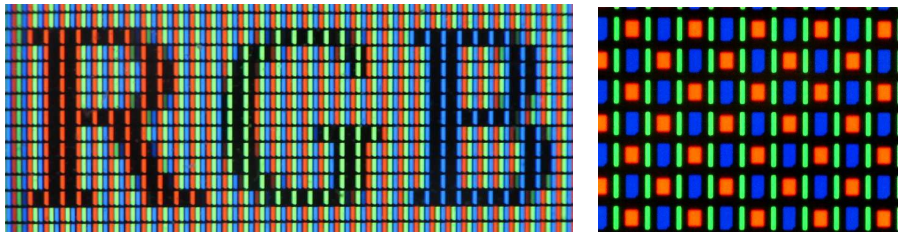
How good does the VR visual display need to be? Three crucial factors for the display are:

1. *Spatial resolution:* How many pixels per square area are needed?
2. *Intensity resolution and range:* How many intensity values can be produced, and what are the minimum and maximum intensity values?
3. *Temporal resolution:* How fast do displays need to change their pixels?

The spatial resolution factor will be addressed in the next paragraph. The second factor could also be called *color resolution and range* because the intensity values of each red, green, or blue subpixel produce points in the space of colors; see Section 6.3. Recall the range of intensities from Figure 5.4 that trigger photoreceptors. Photoreceptors can span seven orders of magnitude of light intensity. However, displays have only 256 intensity levels per color to cover this range. Entering scotopic vision mode does not even seem possible using current display technology because of the high intensity resolution needed at extremely low light levels. Temporal resolution is extremely important, but is deferred until Section 6.2, in the context of motion perception.

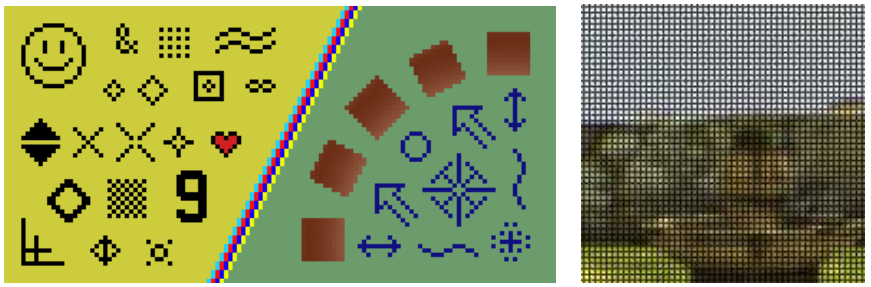
How much pixel density is enough? We now address the spatial resolution. Insights into needed spatial resolution are obtained from the photoreceptor densities. As shown in Figure 5.22, we see individual lights when a display is highly magnified. As it is zoomed out, we may still perceive sharp diagonal lines as being jagged, as shown in Figure 5.23(a); this phenomenon is known as *aliasing*. Another artifact is the *screen-door effect*, shown in Figure 5.23(b); this is commonly noticed in an image produced by a digital LCD projector. What does the display pixel density need to be so that we do not perceive individual pixels? In 2010, Steve Jobs of Apple Inc. claimed that 326 pixels per inch (165 pixels per mm^2) is enough, achieving what they called a *retina display*.

Assume that the fovea is pointed directly at the display to provide the best sensing possible. The first issue is that red, green, and blue cones are arranged in a mosaic, as shown in Figure 5.24. The patterns are more erratic than the engineered versions in Figure 5.22. Vision scientists and neurobiologists have studied the effective input resolution through measures of *visual acuity*. One measure is *cycles*



(a) (b)

Figure 5.22: In displays, the pixels break into subpixels, much in the same way that photoreceptors break into red, blue, and green components. (a) An LCD display. (Photo by Luis Flavio Loureiro dos Santos.) (b) An AMOLED PenTile display from the Nexus One smartphone. (Photo by Matthew Rollings.)



(a) (b)

Figure 5.23: (a) Due to pixels, we obtain a bad case of the *jaggies* (more formally known as *aliasing*) instead of sharp, straight lines. (Figure from Wikipedia user Jmf145.) (b) In the *screen-door effect*, a black grid is visible around the pixels.

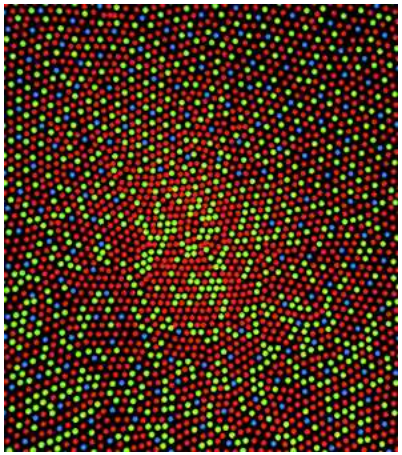
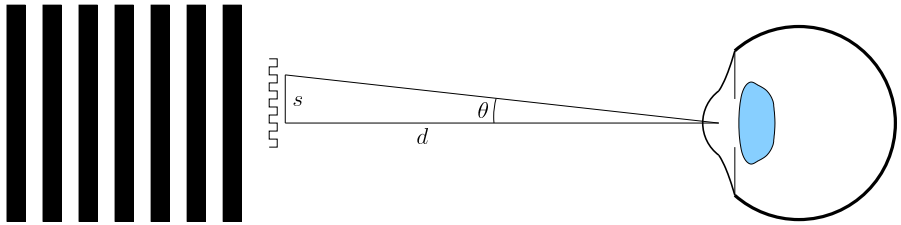


Figure 5.24: Red, green, and blue cone photoreceptors are distributed in a complicated mosaic in the center of the fovea. (Figure by Mark Fairchild.)



(a) (b)

Figure 5.25: (a) A grating that could be used to measure visual acuity (seven cycles are shown). (b) The number of cycles per radian is the number of cycles over a span $s = d \tan \theta$, when viewed from distance d.

per degree, which roughly corresponds to the number of stripes that can be seen as separate along a viewing arc; see Figure 5.25. This has been estimated to be around 77 per degree, based on photoreceptor density and neural processes [31]. Supposing that one cycle is equivalent to the distance between pixel centers, this implies that $(77)^2 = 5929$ pixels should exist over a square area of one degree by one degree.

Using simple trigonometry,

$$s = d \tan \theta, \tag{5.1}$$

we can determine what the span s should be for a viewing angle θ at a distance d from the eye. For very small θ, $\tan \theta \approx \theta$ (in radians). If a smartphone screen is placed 30cm in front of an eye, then $s = 5.236$mm per degree. This means that 5929 pixels should be distributed over $(5.236)^2 = 27.416\text{mm}^2$, which implies that the display should have at least 216.3 pixels per one mm^2 to be a retina display. The density can be significantly lower and nevertheless be a retina display due to optical aberrations in the human eye [64], thereby making Apple's claim of 165 reasonable. Instead of 77 cycles per degree, 60 cycles per degree is commonly used to account for thus, which would require only 131.3 pixels per mm^2.

In the case of VR, we are not looking directly at a screen as in the case of smartphones. If we bring the screen up to 10cm from the eye, then $s = 1.745$mm per degree and the required density increases (using 77 cycles per degree) to 1946 pixels per mm^2; however, it becomes uncomfortable or impossible for people to focus on it at such close range. By inserting a lens for magnification, the display can be brought even closer to the eye. This is commonly done for VR headsets, as was shown in Figure 4.30. Suppose the lens is positioned at its focal distance away from the screen; a typical number is 4cm. In this case, $s = 0.698$mm and 12165 pixels per mm^2 would be needed. For comparison, the most dense smartphone screen available today is the Super AMOLED 1440x2560 5.1 inch screen on the Samsung S6. It has about 516 pixels per mm^2. Using units that are common in industry, the Samsung S6 screen has 577 pixels per inch (along each axis), but the requirement for an effective retina display would be 2801 pixels per inch. Using 60 cycles per degree, it would be only 7386 pixels per mm^2 or 2183 pixels per inch. Thus, to approach the threshold for a VR retina display using a screen and simple lenses, the pixel density needs to be about four times higher in each dimension.

How much FOV is enough? What if the screen is brought even closer to the eye to fill more of the field of view? Based on the photoreceptor density plot in Figure 5.5 and the limits of eye rotations shown in Figure 5.21, the maximum FOV seems to be around 270°, which is larger than what could be provided by a flat screen (less than 180°). Increasing the FOV by bringing the screen closer would require even higher pixel density, but lens aberrations (Section 4.3) at the periphery may limit the effective field of view. Furthermore, if the lens is too thick and too close to the eye, then the eyelashes may scrape it; Fresnel lenses may provide a thin alternative, but present additional artifacts. Thus, the quest for a VR retina display may end with a balance between optical system quality

and limitations of the human eye. Curved screens may help alleviate some of the problems.

Foveated rendering One of the frustrations with this analysis is that we have not been able to exploit that fact that photoreceptor density decreases away from the fovea. We had to keep the pixel density high everywhere because we have no control over which part of the display the user will be look at. If we could track where the eye is looking and have a tiny, movable display that is always positioned in front of the pupil, with zero delay, then much fewer pixels would be needed. This would greatly decrease computational burdens on graphical rendering systems (covered in Chapter 7). Instead of moving a tiny screen, the process can be simulated by keeping the fixed display but focusing the graphical rendering only in the spot where the eye is looking. This is called *foveated rendering*, which has been shown to work [53], but is currently too costly and there is too much delay and other disrepancies between the eye movements and the display updates. In the near future, it may become an effective approach for the mass market.

VOR gain adaptation The *VOR gain* is a ratio that compares the eye rotation rate (numerator) to counter the rotation and translation rate of the head (denominator). Because head motion has six DOFs, it is appropriate to break the gain into six components. In the case of head pitch and yaw, the VOR gain is close to 1.0. For example, if you yaw your head to the left at 10° per second, then the eye yaws at 10° per second in the opposite direction. The VOR roll gain is very small because the eyes have a tiny roll range. The VOR translational gain depends on the distance to the features.

Recall from Section 2.3 that adaptation is a universal feature of our sensory systems. VOR gain is no exception. For those who wear eyeglasses, the VOR gain must adapt due to the optical transformations described in Section 4.2. Lenses affect the field of view and perceived size and distance of objects. The VOR comfortably adapts to this problem by changing the gain. Now suppose that you are wearing a VR headset that may suffer from flaws such as an imperfect optical system, tracking latency, and incorrectly rendered objects on the screen. In this case, adaptation may occur as the brain attempts to adapt its perception of stationarity to componsate for the flaws. In this case, your visual system could convince your brain that the headset is functioning correctly, and then your perception of stationarity in the real world would become distorted until you readapt. For example, after a flawed VR experience, you might yaw your head in the real world and have the sensation that truly stationary objects are sliding back and forth![1]

Display scanout Recall from Section 4.5 that cameras have either a rolling or global shutter based on whether the sensing elements are scanned line-by-line or in parallel. Displays work the same way, but whereas cameras are an *input* device,

[1]This frequently happened to the author while developing and testing the Oculus Rift.

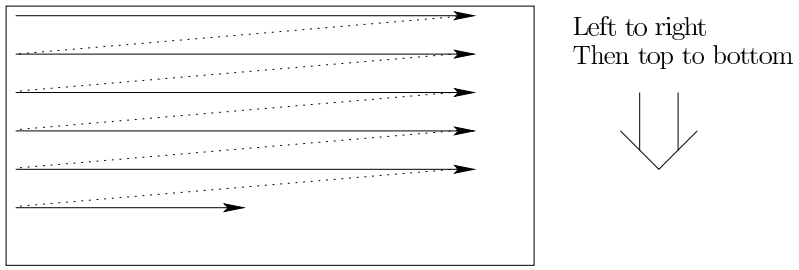
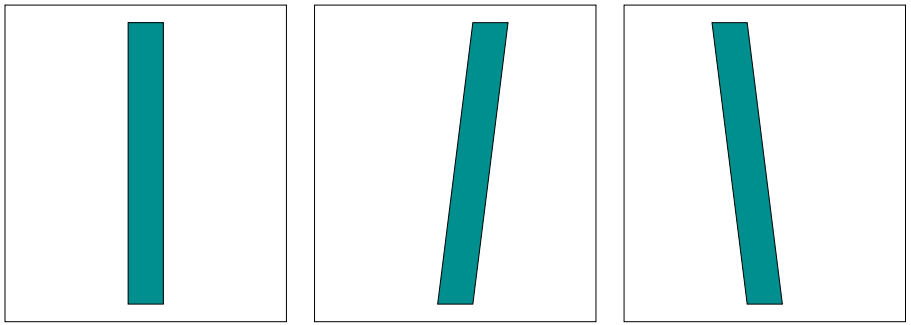


Figure 5.26: Most displays still work in the way as old TV sets and CRT monitors: By updating pixels line-by-line. For a display that has 60 FPS (frames per second), this could take up to 16.67ms.

displays are the *output* analog. The vast majority of displays today have a *rolling scanout* (called *raster scan*), rather than *global scanout*. This implies that the pixels are updated line by line, as shown in Figure 5.26. This procedure is an artifact of old TV sets and monitors, which each had a cathode ray tube (CRT) with phosphor elements on the screen. An electron beam was bent by electromagnets so that it would repeatedly stike and refresh the glowing phosphors.

Due to the slow charge and response time of photoreceptors, we do not preceive the scanout pattern during normal use. However, when our eyes, features in the scene, or both are moving, then side effects of the rolling scanout may become perceptible. Think about the operation of a line-by-line printer, as in the case of a receipt printer on a cash register. If we pull on the tape while it is printing, then the lines would become streched apart. If it is unable to print a single line at once, then the lines themselves would become slanted. If we could pull the tape to the side while it is printing, then the entire page would become slanted. You can also achieve this effect by repeatedly drawing a horizontal line with a pencil while using the other hand to gently pull the paper in a particular direction. The paper in this analogy is the retina and the pencil corresponds to light rays attempting to charge photoreceptors. Figure 5.27 shows how a rectangle would distort under cases of smooth pursuit and VOR. Current displays have an option called *vsync*, which synchronizes output to the dislay so that the displayed image corresponds to a single, undistorted frame. By turning off this option, more frames per second could be rendered to the display then it can handle. The display buffer simple gets updated in the middle of the scan. This reduces the side effects due to scanout, but introduces another artifact called *tearing* (as in tearing a sheet of paper). Further improvements could be made by *beam racing* and *just-in-time pixels*, which means rendering each line by taking into account the precise time at which it will be drawn [1, 15, 97]. Yet another problem with displays is that the pixels could take so long to switch (up to 20ms) that sharp edges appear to be blurred. We will continue discussing these problems in Section 6.2 in the context of motion perception.



(a) (b) (c)

Figure 5.27: Artifacts due to display scanout: (a) A vertical rectangle in the scene. (b) How it may distort during smooth pursuit while the rectangle moves to the right in the virtual world. (c) How a stationary rectangle may distort when rotating the head to the right while using the VOR to compensate. The cases of (b) are (c) are swapped if the direction of motion is reversed in each case.

Retinal image slip Recall that eye movements contribute both to maintaining a target in a fixed location on the retina (smooth pursuit, VOR) and also changing its location slightly to reduce perceptual fading (microsaccades). During ordinary activities (not VR), the eyes move and the image of a feature may move slightly on the retina due to motions and optical distortions. This is called *retinal image slip*. Once a VR headset is used, the motions of image features on the retina might not match what would happen in the real world. This is due to many factors already mentioned, such as optical distortions, tracking latency, and display scanout. Thus, the retinal image slip due to VR artifacts does not match the retinal image slip encountered in the real world? The consequences of this barely been identified, much less characterized scientifically. They are likely to contribute to fatigue, and possibly VR sickness. As an example of the problem, there is evidence that mircosaccades are triggered by the lack of retinal image slip [37]. This implies that differences in retinal image slip due to VR usage could interfere with microsaccade motions, which are already not fully understood.

Vergence-accommodation mismatch Recall from Section 4.4 that accommodation is the process of changing the eye lens' optical power so that close objects can be brought into focus. This normally occurs with both eyes fixated on the same object, resulting in a steroscopic view that is brought into focus. In the real world, the vergence motion of the eyes and the accommodation of the lens are tightly coupled. For example, if you place your finger 10cm in front of your face, then your eyes will try to increase the lens power while the eyes are strongly converging. If a lens is placed at a distance of its focal length from a screen,

then with normal eyes it will always be in focus while the eye is relaxed (recall Figure 4.30). What if an object is rendered to the screen so that it appears to be only 10cm away? In this case, the eyes strongly converge, but they do not need to change the optical power of the eye lens. The eyes may nevertheless try to accommodate, which would have the effect of blurring the perceived image. The result is called *vergence-accommodation mismatch* because the stimulus provided by VR is inconsistent with the real world. Even if the eyes become accustomed to the mismatch, the user may feel extra strain or fatigue after prolonged use. The eyes are essentially being trained to allow a new degree of freedom: Separating vergence from accommodation, rather than coupling them. Engineering solutions may provide some relief from this problem, but they are currently too costly and imprecise. For example, the mismatch can be greatly reduced by employing eye tracking to estimate the amount of vergence and then altering the power of the optical system [2, 86].

Further Reading

For further reading on the photoreceptor mosaic, see Chapter 3 of [152].

Structure of cone photoreceptors, D. Mustafi, A. H. Engel, K. Palczewski, Progress in Retinal and Eye Research, 28, 2009, pp. 289-302.

Photoreceptor density variation over humans: [31].

Retina display analysis (non-VR): [64]

More about eyes and lenses together: [135].

All about eye movement from a neuroscience perspective: [82].

VOR gain adapation: [34, 46, 129]

Survey of microsaccades: [121]

Smooth pursuit and saccade coordination: [38]

Monkey paper for head/eye coordination: [76]

See Oxford Handbook of Eye Movements: [85]

For more neuroscience, see Chapter 7 of [92].

Nice vision summary:
`http://onlinelibrary.wiley.com/doi/10.1002/9780470478509.neubb001012/pdf`

You can also figure out whether it is worthwhile to upgrade your TV by using the retina display analysis:
`http://www.rtings.com/tv/learn/size-to-distance-relationship`

Abrash blog post about scanout: [1]

Comfort of vergence-accommodation mismatch: [131]

Chapter 6

Visual Perception

Chapter Status	Taken from *Virtual Reality*, S. M. LaValle
	This online chapter is not the final version! Check **http://vr.cs.uiuc.edu/** for information on the latest draft version. This draft was compiled on June 29, 2016.

This chapter continues where Chapter 5 left off by transitioning from the *physiology* of human vision to *perception*. If we were computers, then this transition might seem like going from low-level hardware to higher-level software and algorithms. How do our brains interpret the world around us so effectively in spite of our limited biological hardware? To understand how we may be fooled by visual stimuli presented by a display, you must first understand how our we perceive or interpret the real world under normal circumstances. It is not always clear what we will perceive. We have already seen several optical illusions. VR itself can be considered as a grand optical illusion. Under what conditions will it succeed or fail?

Section 6.1 covers perception of the distance of objects from our eyes, which is also related to the perception of object scale. Section 6.2 explains how we perceive motion. An important part of this is the illusion of motion that we perceive from videos, which are merely a sequence of pictures. Section 6.3 covers the perception of color, which may help explain why displays use only three colors (red, green, and blue) to simulate the entire spectral power distribution of light (recall from Section 4.1). Finally, Section 6.4 presents a statistically based model of how information is combined from multiple sources to produce a perceptual experience.

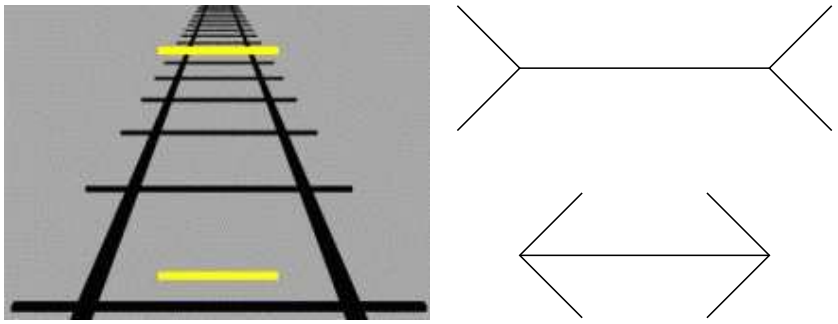
6.1 Perception of Depth

This section explains how humans judge the distance from their eyes to objects in the real world using vision. The perceived distance could be *metric*, which means

Figure 6.1: This painting uses a monocular depth cue called a *texture gradient* to enhance depth perception: The bricks become smaller and thinner as the depth increases. Other cues arise from perspective projection, including height in the visual field and retinal image size. ("Paris Street, Rainy Day," Gustave Caillebotte, 1877. Art Institute of Chicago.)

that an estimate of the absolute distance is obtained. For example, a house may appear to be about 100 meters away. Alternatively, the distance information could be *ordinal*, which means that the relative arrangement of visible objects can be inferred. For example, one house appears to be closer than another one because it is partially blocking the view of the further one.

Monocular vs. stereo cues A piece of information that is derived from sensory stimulation and is relevant for perception is called a *sensory cue* or simply a *cue*. In this section, we consider only *depth cues*, which contribute toward depth perception. If a depth cue is derived from the photoreceptors or movements of a single eye, then it is called a *monocular depth cue*. If both eyes are required, then it is a *stereo depth cue*. There are many more monocular depth cues than stereo, which explains why we are able to infer so much depth information from a single photograph. Figure 6.1 shows an example. The illusions in Figure 6.2 show that even simple line drawings are enough to provide strong cues. Interestingly, the cues used by humans also work in computer vision algorithms to extract depth information from images [145].



(a) (b)

Figure 6.2: Even simple line drawings provide signficant depth cues. (a) The Ponzo illision: The upper yellow bar appears to be longer, but both are the same length. (b) The Müller-Lyer illision: The lower horizontal segment appears to be shorter than the one above, but they are the same length.

6.1.1 Monocular depth cues

Retinal image size Many cues result from the geometric distortions caused by perspective projection; recall the "3D" appearance of Figure 1.19(c). For a familiar object, such as a human, coin, or basketball, we often judge its distance by how "large" is appears to be. Recalling the perspective projection math from Section 3.4, the size of the image on the retina is proportional to $1/z$, in which z is the distance from the eye (or the common convergence point for all projection lines). See Figure 6.3. The same thing happens when taking a picture with a camera: A picture of a basketball would occupy larger part of the image, covering more pixels, as it becomes closer to the camera. Two important factors exist. First, the viewer must be familiar with the object to the point of comfortably knowing its true size. For familiar objects, such as people or cars, our brains performance *size constancy scaling* by assuming that the distance, rather than the size, of the person is changing if they come closer. Size constancy falls of the general heading of *subjective constancy*, which appears through many aspects of perception, including shape, size, and color. The second factor is that, the object must be appear naturally so that it does not conflict with other depth cues.

If there is significant uncertainty about the size of an object, then knowledge of its distance should contribute to estimating its size. This falls under *size perception*, which is closely coupled to depth perception. Cues for each influence the other, in a way discussed in Section 6.4.

One controversial theory is that our *perceived visual angle* differs from the actual visual angle. The visual angle is proportional to the retinal image size. This theory is used to explain the illusion that the moon appears to be larger when it is near the horizon. For another example, see Figure 6.4.

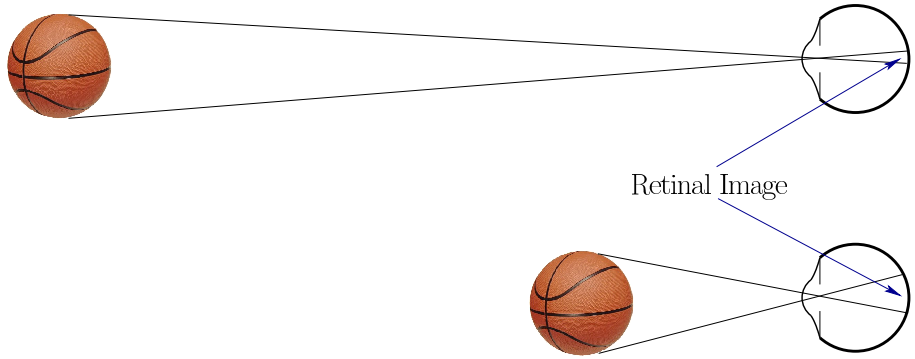


Figure 6.3: The retinal image size of a familiar object is a strong monocular depth cue. The closer object projects onto a larger number of photoreceptors, which cover a larger portion of the retina.

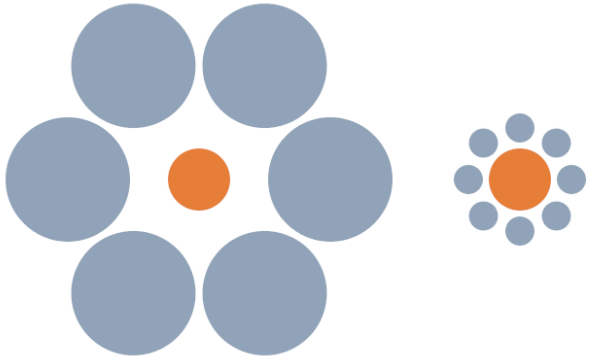
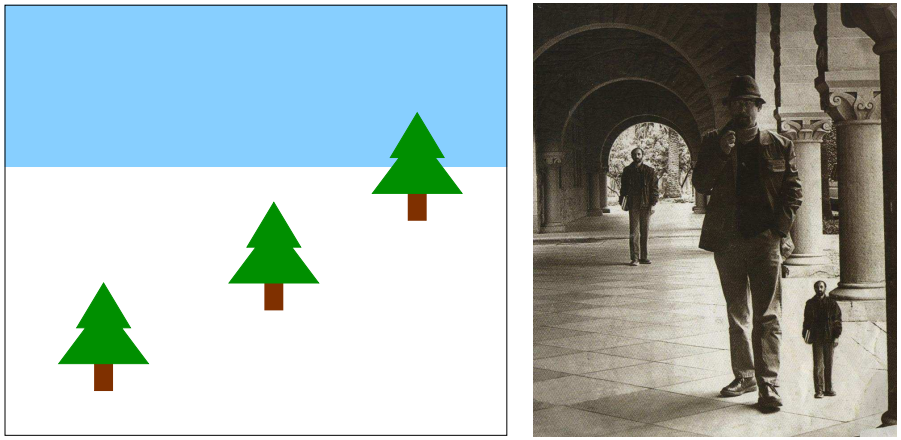


Figure 6.4: For the Ebbinghaus illision, the inner disc appears larger when surrounded by smaller discs. The inner disc is the same size in either case. This may be evidence of disrepancy betwen the true visual angle (or retinal image size) and the *perceived* visual angle.



(a) (b)

Figure 6.5: Height in visual field. (a) Trees closer to the horizon appear to be further away, even though all yield the same retinal image size. (b) Incorrect placement of people in the visual field illustrates *size constancy scaling*, which is closely coupled with depth cues.

Height in the visual field Figure 6.5(a) illustrates another important cue, which is the height of the object in the visual field. The Ponzo illision in Figure 6.2(a) exploits this cue. Suppose that we can see over a long distance without obstructions. Due to perspective projection, the horizon is a line that divides the view in half. The upper half is perceived as the sky, and the lower half is the ground. The distance of objects from the horizon line corresponds directly to their distance due to perspective projection: The closer to the horizon, the further the perceived distance. Size constancy scaling, if available, combines with the height in the visual field, as shown in Figure 6.5(b).

Accommodation Recall from Section 4.4 that the human eye lens can change its optical power through the process of accommodation. For young adults, the amount of change is around 10D (diopters), but it decreases to less than 1D for adults over 50 years old. The ciliary muscles control the lens and their tension level is reported to the brain through efference copies of the motor control signal. This is the first depth cue that does not depend on signals generated by the photoreceptors.

Motion parallax Up until now, the depth cues have not exploited motions. If you have ever looked out the side window of a fast-moving vehicle, you might have noticed that the nearby objects race by much faster than further objects. The relative difference in speeds is called *parallax* and is an important depth cue; see

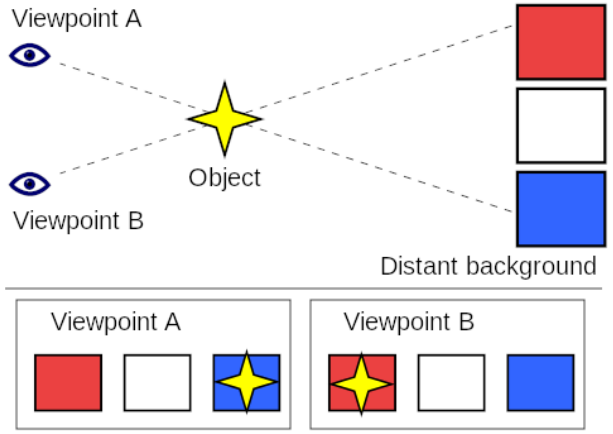


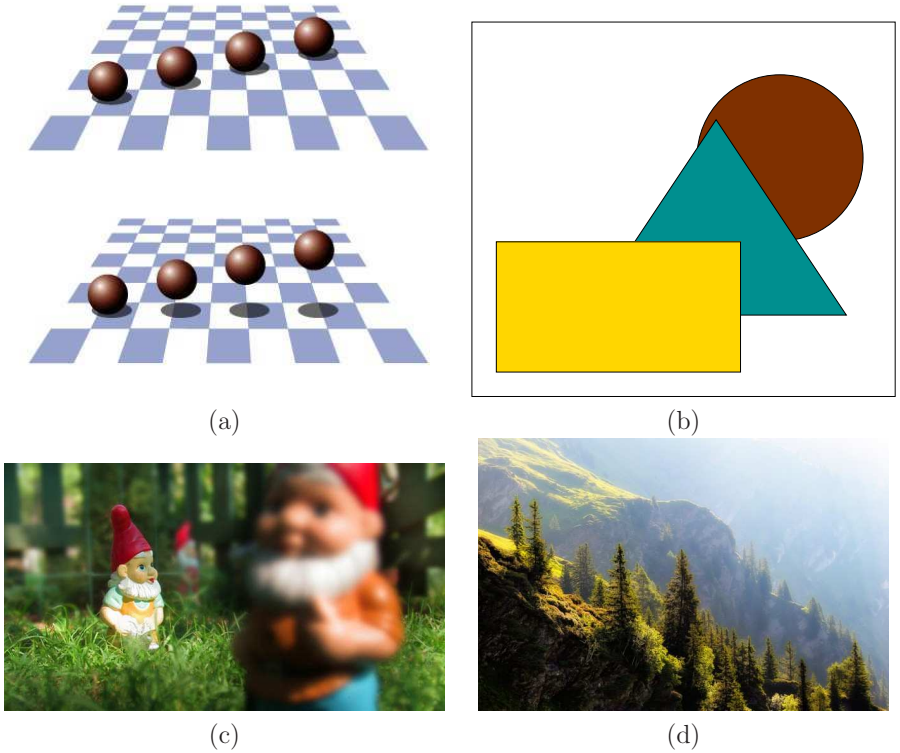
Figure 6.6: Motion parallax: As the perspective changes laterally, closer objects have larger image displacements than further objects. (Figure from Wikipedia.)

Figure 6.6. Even two images, from varying viewpoints within a short amount of time, provide strong depth information. Imagine trying to simulate a *stereo rig* of cameras my snapping one photo and quickly moving the camera sideways to snap another. If the rest of the world is stationary, then the result is roughly equivalent to having two side-by-side cameras. Pigeons frequently bob their heads back and forth to obtain stronger depth information than is provided by their pair of eyes. Finally, closely related to motion parallax is *optical flow*, which is a characterization of the rates at which features move across the retina. This will be revisited in Sections 6.2 and 8.4.

Other monocular cues Figure 6.7 shows several other monocular cues. As shown in Figure 6.7(a), shadows that are cast by a light source encountering an object provide an important cue. Figure 6.7(b) shows a simple drawing that provides an ordinal depth cue called *interposition* by indicating which objects are in front of others. Figure 6.7(c) illustrates the *image blur* cue, where levels are depth are inferred from the varying sharpness of focus. Figure 6.7(d) shows an *atmospheric cue* in which air humidity causes far away scenery to have lower contrast, thereby appearing to be further away.

6.1.2 Stereo depth cues

As you may expect, focusing both eyes on the same object enhances depth perception. Humans perceive a single focused image over a surface in space called the *horopter*; see Figure 6.8. Recall the vergence motions from Section 5.3. Similar to the accommodation cue case, motor control of the eye muscles for vergence mo-



(a) (b) (c) (d)

Figure 6.7: Several more monocular depth cues: (a) *Shadows* resolve ambiguous depth in the *ball and shadow illusion*. (b) The *interposition* of objects provides an ordinal depth cue. (c) Due to *image blur*, one gnome appears to be much closer than the others. (d) This scene provides an *atmospheric cue*: Some scenery is perceived to be further away because it has lower contrast.

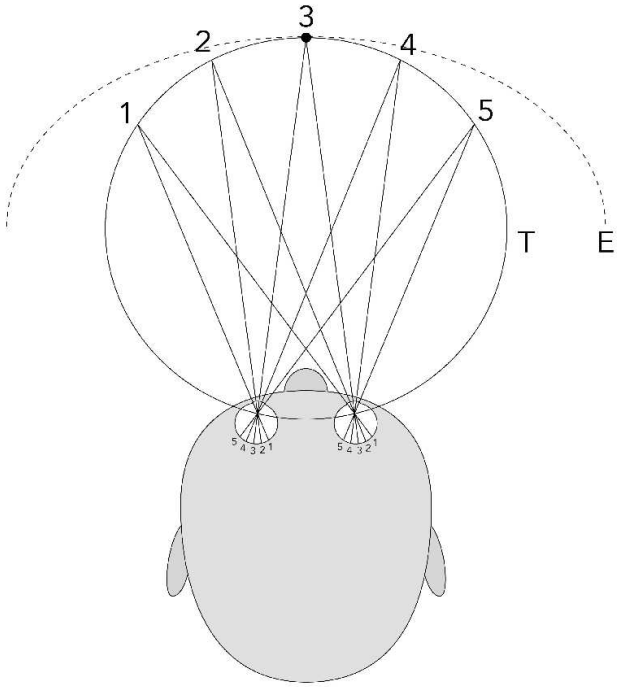


Figure 6.8: The *horopter* is the loci of points over which the eyes can converge and focus on a single depth. The T curve shows the theoretical horopter based on simple geometry. The E curve shows the empirical horopter, which is much larger and correspond to the region over which a single focused image is perceived. (Figure by Rainer Zenz.)

tions provides information to the brain about the amount of convergence, thereby providing a direct estimate of distance. Each eye provides a different viewpoint, which results in different images on the retina. This phenomenon is called *binocular disparity*. Recall from (3.49) in Section 3.5 that the viewpoint is shifted to the right or left to provide a lateral offset for each of the eyes. The transform essentially shifts the virtual world to either side. The same shift would happen for a stereo rig of side-by-side cameras in the real world. However, the binocular disparity for humans is different because the eyes can rotate to converge, in addition to having a lateral offset. Thus, when fixating on an object, the retinal images between the left and right eyes may vary only slightly, but this nevertheless provides a powerful cue used by the brain.

Furthermore, when converging on an object at one depth, we perceive double images of objects at other depths (although we usually pay no attention to it). This double-image effect is called *diplopia*. You can perceive it by placing your finger about 20cm in front of your face and converging on it. While fixating on your finger, you should perceive double images on other objects around the periphery. You can also stare into the distance while keeping your finger in the same place. You will then see a double image of your finger. If you additionally roll your head back and forth, you should appears as if the left and right versions of your finger are moving up and down with respect to each other. These correspond to dramatic differences in the retinal image, but we are usually not aware of them because we perceive a both retinal images as a single image.

6.1.3 Implications for VR

Incorrect scale perception A virtual world may be filled with objects that are not familiar to us in the real world. In many cases, they might resemble familiar objects, but their precise scale might be difficult to determine. Consider the Tuscany demo world from Oculus VR, shown in Figure 6.9. The virtual villa is designed to be inhabited with humans, but it is difficult to judge the relative sizes and distances of objects because there are not enough familiar objects. Futher complicating the problem is that the user's height in VR might not match his height in the virtual world. Is the user too short, or is the world too big? A common and confusing occurrence is that the user might be sitting down in the real world, but standing in the virtual world. An additional complication occurs if the interpupillary distance (recall from Section 4.4) is not matched with the real world. For example, if the user's pupils are 64mm apart in the real world but only 50mm apart in the virtual world, then the virtual world will seem much larger, which dramatically affects depth perception. Likewise, if the pupils are very far apart, the user could either feel enormous or the virtual world might seem small. Imagine simulating a Godzilla experience, where the user is 200 meters tall and the entire city appears to be a model. It is fine to experiment with such scale and depth distortions in VR, but it is important to understand their implications on the user's perception.

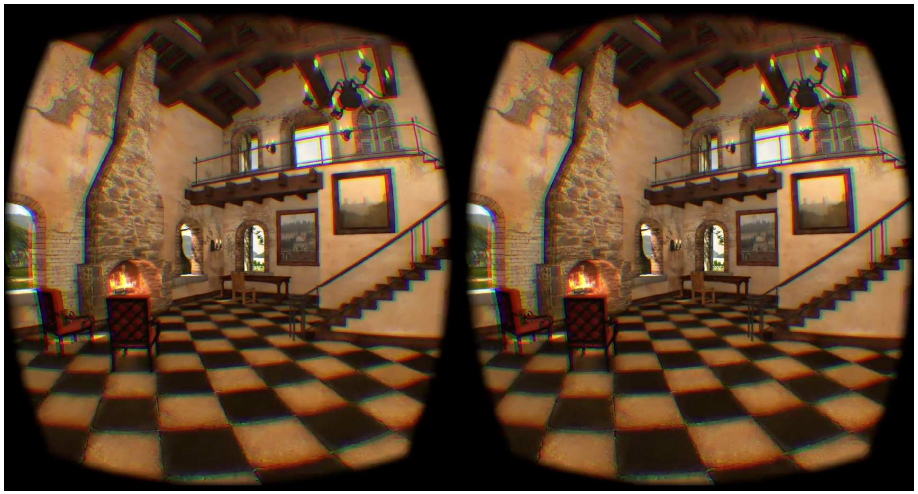
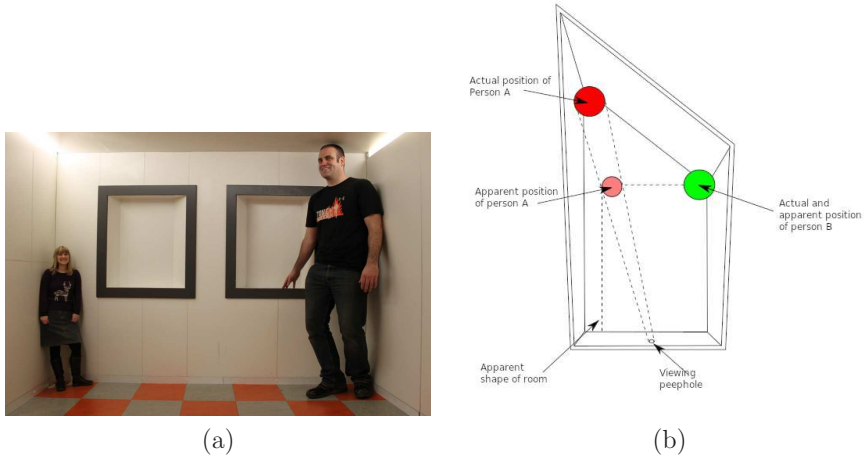


Figure 6.9: In the Tuscany demo from Oculus VR, there are not enough familiar objects to precisely resolve depth and size. Have you ever been to a villa like this? Are the floor tiles a familiar size? Is the desk too low?

Mismatches In the real world, all of the depth cues work together in harmony. We are sometimes fooled by optical illusions that are designed to intentionally cause inconsistencies among cues. Sometimes a simple drawing is sufficient. Figure 6.10 shows an elaborate illusion that requires building a distorted room in the real world. It is perfectly designed so that when viewed under perspective projection from one location, it appears to be a rectangular box. Once our brains accept this, we unexpectedly perceive the size of people changing as they walk across the room! This is because all of the cues based on perspective appear to be functioning correctly. Section 6.4 may help you to understand how multiple cues are resolved, even in the case of inconsistencies.

In a VR system, it is easy to cause mismatches and in many cases they are unavoidable. Recall from Section 5.4 that vergence-accommodation mismatch occurs in VR headsets. Another source of mismatch may occur from imperfect head tracking. If there is significant latency, then the visual stimuli will not appear in the expected place at the expected time. Furthermore, many tracking systems track the head orientation only. This makes it impossible to use motion parallax as a depth cue if the user moves from side to side without any rotation. To preserve most depth cues based on motion, it is important to track head *position*, in addition to orientation; see Section 9.3. Optical distortions may cause even more mistmatch.

Monocular cues are powerful! A common misunderstanding among the general public is that depth perception enabled by stereo cues alone. We are bom-



(a) (b)

Figure 6.10: The Ames room: (a) Due to incorrect depth cues, incorrect size perception results. (b) The room is designed so that it only appears to be rectangular after perspective projection is applied. One person is actually much further away than the other. (Figure by Alex Valavanis.)

barded with marketing of "3D" movies and *stereo displays*. The most common instance today is the use of circularly polarized *3D glasses* in movie theaters so that each eye receives a different image when looking at the screen. VR is no exception to this common misunderstanding. CAVE systems provided 3D glasses with an active shutter inside so that alternating left and right frames can be presented to the eyes. Note that this cuts the frame rate in half. Now that we have comfortable headsets, presenting separate visual stimuli to each eye is much simpler. One drawback is that the rendering effort (the subject of Chapter 7) is doubled, although this can be improved through some context-specific tricks.

As you have seen in this section, there are many more monocular depth cues than stereo cues. Therefore, it is wrong to assume that the world is perceived as "3D" *only if* there are stereo images. This is particularly valuable for leveraging captured data from the real world. Recall from Section 1.1 that the virtual world may be synthetic or captured. It is generally more costly to create synthetic worlds, but it is then simple to generate stereo viewpoints (at a higher rendering cost). On the other hand, capturing panoramic, monoscopic images and movies is fast and inexpensive (examples were shown in Figure 1.7). There are already smartphone apps that stitch pictures together to make a panoramic photo, and direct capture of panoramic video is likely to be a standard feature on smartphones within a few years. By recognizing that this content is sufficiently "3D" due to the wide field of view and monocular depth cues, it becomes a powerful way to create VR experiences. There are already hundreds of millions of images in Google Street

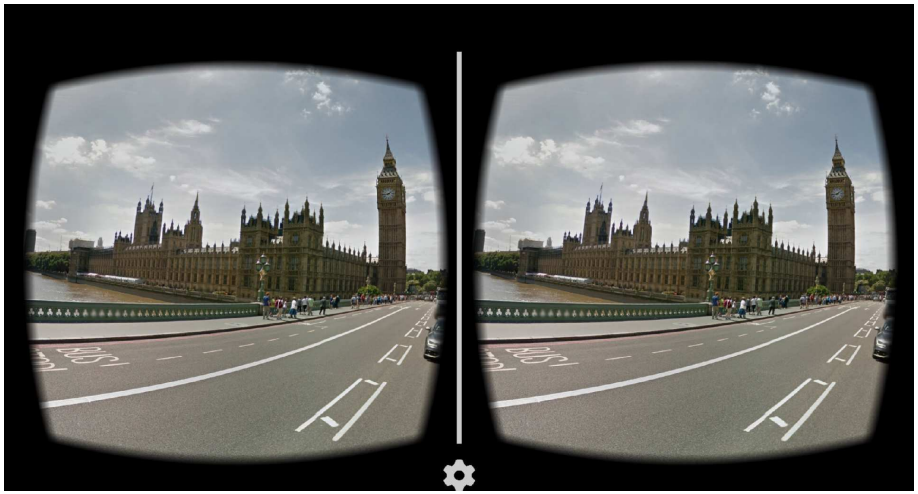


Figure 6.11: In Google Cardboard and other VR headsets, hundreds of millions of panoramic Street View images can be viewed. There is significant depth perception, even when the same image is presented to both eyes, because of monoscopic depth cues.

View, shown in Figure 6.11, which can be easily viewed using Google Cardboard or other headsets. They provide a highly immersive experience with substantial depth perception, even though there is no stereo. There is even strong evidence that stereo displays cause significant fatigue and discomfort, especially for objects as a close depth [**?**]. Therefore, one should think very carefully about the use of stereo. In many cases, it might be more time, cost, and trouble that it is worth to obtain the stereo cues when there may already be sufficient monocular cues for the VR task or experience.

6.2 Perception of Motion

We rely on our vision to perceive motion for many crucial activities. One use to separate a moving figure from a stationary background. For example, a camouflaged animal in the forest might only become noticeable when moving. This is clearly useful whether humans are the hunter or the hunted. Motion also helps people to assess the 3D structure of an object. Imagine assessing the value of a piece of fruit in the market by rotating it around. Another use is to visually guide actions, such as walking down the street or hammering a nail. VR systems have the tall order of replicating these uses in a virtual world in spite of limited technology. Just as important as the perception of motion is the perception of non-motion, which we called *perception of stationarity* in Section 2.3. For exam-

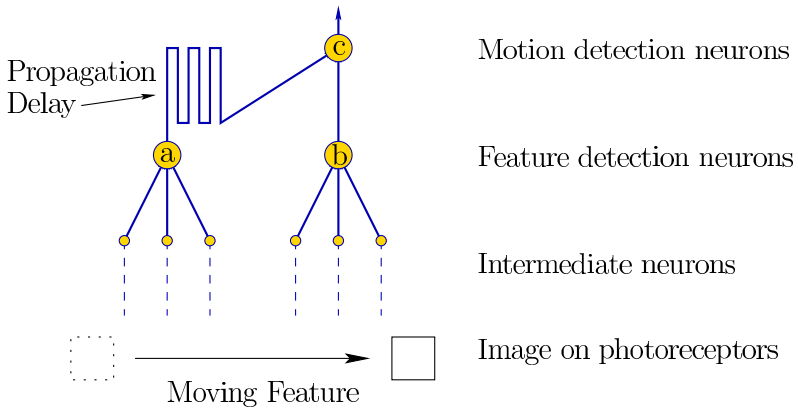


Figure 6.12: The neural circuitry directly supports motion detection. As the image feature moves across the retina, nearby feature detection neurons (labeled a and b) activate in succession. Their outputs connect to motion detection neurons (labeled c). Due to different path lengths from a and b to c, the activation signal arrives at different times. Thus, c activates when the feature was detected by a slightly before being detected by b.

ple, if we apply the VOR by turning our heads, do the virtual world objects move correctly on the display so that they appear to be stationary? Slight errors in time or image position might inadvertently trigger the perception of motion.

6.2.1 Detection mechanisms

Reichardt detector Figure 6.12 shows a neural circuitry model, called a *Reichardt detector*, which responds to directional motion in the human vision system. Neurons in the ganglion layer and LGN detect simple features in different spots in the retinal image. At higher levels, motion detection neurons exist that respond when the feature moves from one spot on the retina to another nearby. The motion detection neuron activates for a feature speed that depends on the difference in path lengths from its input neurons. It is also sensitive to a particular direction of motion based on the relative locations of the receptive fields of the input neurons. Due to the simplicity of the motion detector, it can be easily fooled. Figure 6.12 shows a feature moving from left to right. Suppose that a train of features moves from right to left. Based on the speed of the features and the spacing between them, the detector may inadvertently fire, causing motion to be perceived in the opposite direction. This is the basis of the *wagon-wheel effect*, for which a wheel with spokes or a propeller may appear to be rotating in the opposite direction, depending on the speed. The process can be further disrupted by causing eye vibrations from humming [125]. This simulates stroboscopic conditions, which discussed in Section 6.2.2. Another point is that the motion detectors are subject

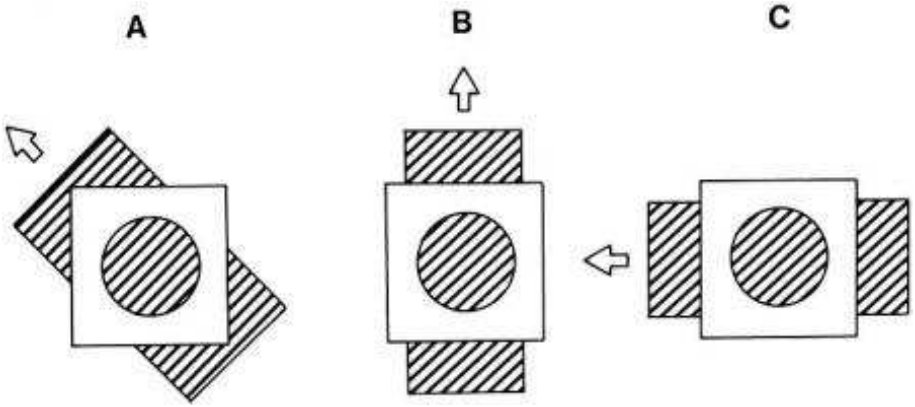


Figure 6.13: Due to local nature of motion detectors, the *aperture problem* results. The motion of the larger body is ambiguous when perceived through a small hole because a wide range of possible body motions could produce the same effect inside of the hole. An incorrect motion inference usually results.

to adaptation. Therefore, several illusions exist, such as the *waterfall illusion* and the *spiral aftereffect*, in which incorrect motions are perceived due to aftereffects.

From local data to global conclusions Motion detectors are *local* in the sense that a tiny portion of the visual field causes each to activate. In most cases, data from detectors across large patches of the visual field are *integrated* to indicate coherent motions of rigid bodies. (An exception would be staring a pure analog TV static.) All pieces of a rigid body move through space according to the equations from Section 3.2. This coordinated motion is anticipated by our visual experience to match common expectations. If too much of the moving body is blocked, then the *aperture problem* results, which is shown in Figure 6.13. A clean mathematical way to describe the global motions across the retina is by a *vector field*, which assigns a velocity vector at every position. The global result is called the *optical flow*, which provides powerful cues for both object motion and self motion. The latter case results in *vection*, which is a leading cause of VR sickness; see Sections 8.4 and 10.2 for details.

Distinguishing object motion from observer motion Figure 6.14 shows two cases that produce the same images across the retina over time. In Figure 6.14(a), the eye is fixed while the object moves by. In Figure 6.14(b), the situation is reversed: The object is fixed, but the eye moves. The brain uses several cues to differentiate between these cases. Saccadic suppression, which was mentioned in Section 5.3, suppresses vision during movements, which may suppress motion detectors in the second case. Another cue is provided by proprioception, which is the body's ability to estimate its own motions due to motor commands. This

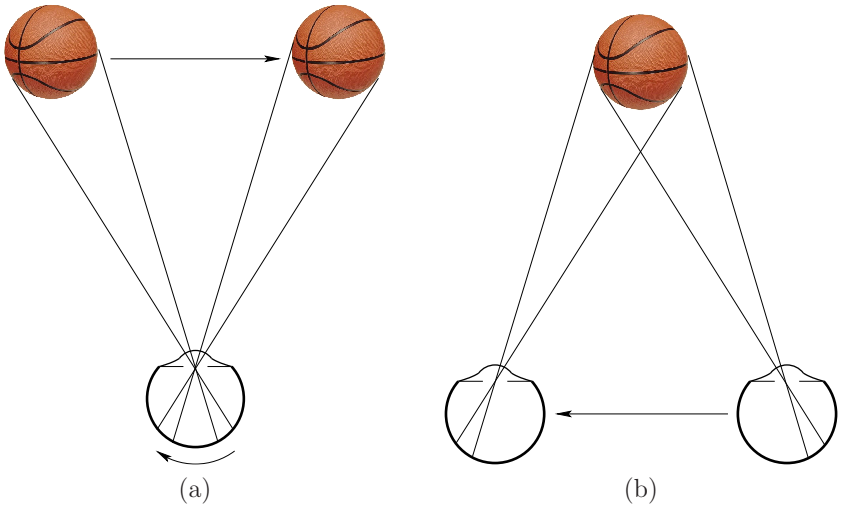


Figure 6.14: Two motions that cause equivalent movement of the image on the retina: (a) The eye is fixed and the object moves; (b) the eye moves while the object is fixed. Both of these are hard to achieve in practice due to eye rotations (smooth pursuit and VOR).

includes the use of eye muscles in the second case. Finally, information is provided by large-scale motion. If it appears that the entire scene is moving, then the brain assumes the most likely interpretation, which is that the user must be moving. This is why the haunted swing illusion, shown in Figure 2.20, is so effective.

6.2.2 Stroboscopic apparent motion

Nearly everyone on Earth has seen a motion picture, whether through a TV, smartphone, or movie screen. The motions we see are an illusion because a sequence of still pictures is being flashed onto the screen. This phenomenon is called *stroboscopic apparent motion*; it was discovered and refined across the 19th century. The *zoetrope*, shown in Figure 6.15 was developed around 1834. It consists of a rotating drum with slits that allow each frame to be visible for an instant while the drum rotates. In Section 1.3, Figure 1.20 showed the Horse in Motion film from 1878.

Why does this illusion of motion work? An early theory, which has largely been refuted in recent years, is called *persistence of vision*. The theory states that images persist in the vision system during the intervals in between frames, thereby causing them to be perceived as continuous. One piece of evidence against this theory is that images persist for up to 100ms [], which implies that the 10 FPS (Frames Per Second) is the slowest speed that stroboscopic apparent motion would

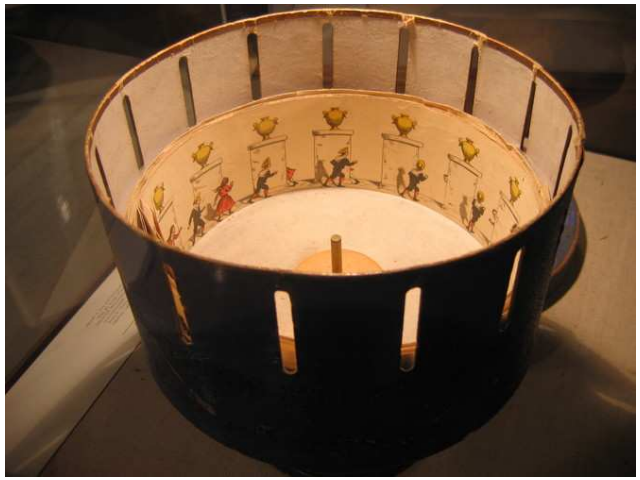


Figure 6.15: The *zoetrope* was developed in the 1830s and provided stroboscopic apparent motion as images became visible through slits in a rotating disc.

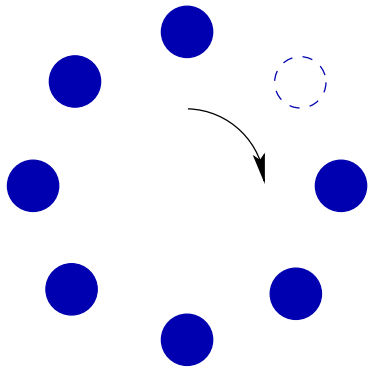


Figure 6.16: The *phi phenomenon* and *beta movement* are physiologically distinct effects in which motion is perceived. In the sequence of dots, one is turned *off* at any give time. A different dot is turned *off* in each frame, following a clockwise pattern. At a very low speed (2 FPS), beta movement triggers a motion perception of each dot on the border of the *off* dot moving positions. At a higher rate, such as 15 FPS, there appears to be a moving hole; this corresponds to the phi phenomenon.

FPS	Occurrence
2	Stroboscopic apparent motion starts
10	Ability to distinguish individual frames is lost
16	Old home movies; early silent films
24	Hollywood classic standard
25	PAL television before interlacing
30	NTSC television before interlacing
48	Two-blade shutter; proposed new Hollywood standard
50	Interlaced PAL television
60	Interlaced NTSC television; perceived flicker in some displays
72	Three-blade shutter; minimum CRT refresh rate for comfort
90	Modern VR headsets; no more discomfort from flicker
1000	Ability to see zipper effect for fast, blinking LED
5000	Cannot perceive zipper effect

Figure 6.17: Various frame rates and comments on the corresponding stroboscopic apparent motion. Units are in Frames Per Second (FPS).

work; however, it is also perceived down to 2 FPS [140]. Another piece of evidence against the persistence of vision is the existence of stroboscopic apparent motions that cannot be accounted for by it. The *phi phenomenon* and *beta movement* are examples of motion perceived in a sequence of blinking lights, rather than flashing frames (see Figure 6.16). The most likely reason that stroboscopic apparent motion works is that it triggers the neural motion detection circuitry illustrated in Figure 6.12 [92, 96].

Frame rates How many frames per second are appropriate for a motion picture? The answer depends on the intended use. Figure 6.17 shows a table of significant frame rates from 2 to 5000. Stroboscopic apparent motion begins at 2 FPS. Imagine watching a security video at this rate. It is easy to distinguish individual frames, but the motion of a person would also be perceived. Once 10 FPS is reached, the motion is obviously more smooth and we start to lose the ability to distinguish individual frames. Early silent films ranged from 16 to 24 FPS. The frame rates were often fluctuatiing and at a faster speed than they were filmed. Once sound was added to film, incorrect speeds and fluctuations in the speed were no longer tolerated because both sound and video needed to be synchronized. This motivated a fixed rate of 24 FPS that is still used today by the movie industry. Personal video cameras remained at 16 or 18 FPS into the 1970s. The famous Zapruder film of the Kennedy assassination in 1963 was taken at 18.3 FPS. Although 24 FPS may be enough to perceive motions smoothly, a large part of cinematography is devoted to ensuring that motions are not so fast that jumps are visible due to the slow frame rate.

Such low frame rates unfortunately lead to perceptible flicker as the images rapidly flash on the screen with black in between. This motivated several workarounds.

In the case of movie projectors, two-blade and three-blade shutters were invented so that they would show each frame two or three times, respectively. This enabled movies to be shown at 48 FPS and 72 FPS, thereby reducing discomfort from flickering. Analog television broadcasts in the 20th century were at 25 (*PAL standard*) or 30 FPS (*NTSC standard*), depending on the country. To double the frame rate and reduce perceived flicker, they used *interlacing* to draw half the image in one frame time, and then half in the other. Every other horizontal line is drawn in the first half, and the remaining lines are drawn in the second. This increased the frames rates on television screens to 50 and 60 FPS. The game industry has used 60 FPS standard target for smooth game play.

As people started sitting close to giant CRT monitors in the early 1990s, the flicker problem became problematic again. Our perception of flicker is stronger at the periphery, particularly at about 30° from center []. Furthermore, even when flicker cannot be directly perceived, it may still contribute to fatigue or headaches. Therefore, frame rates were increased to even higher levels. A minimum acceptable ergonomic standard for large CRT monitors was 72 FPS, with 85 to 90 FPS being widely considered as sufficiently high to eliminate flicker problems. The problem has been carefully studied by psychologists under the heading of *flicker fusion threshold*; the precise rates at which flicker is perceptible or causes fatigue depends on many factors in addition to FPS, such as position on retina, age, color, and light intensity. Thus, the actual limit depends on the kind of display size, specifications, how it is used, and who is using it. Modern LCD and LED displays, used as televisions, computer screens, and smartphone screens, have 60, 120, and even 240 FPS.

The story does not end there. If you connect an LED to a pulse generator (put a resistor in series), then flicker can be perceived at much higher rates. Go to a dark room and hold the LED in your hand. If you wave it around so fast that your eyes cannot track it, then the flicker becomes perceptible as a zipper pattern. This happens because each time the LED pulses on, it is imaged in a different place on the retina. Without image stabilization, it appears as an array of lights. The faster the motion, the further apart the images will appear. The higher the pulse rate (or FPS), the closer together the images will appear. Therefore, to see the zipper effect at very high speeds, you need to move the LED very quickly. It is possible to see the effect for a few thousand FPS.

6.2.3 Implications for VR

Unfortunately, VR systems require much higher display performance than usual. We have already seen in Section 5.4 that much higher resolution is needed so that pixels and aliasing artifacts are not visible. The next problem is that higher frame rates are needed in comparison to ordinary television or movie standards of 24 FPS or even 60 FPS. To understand why, see Figure 6.18. The problem is easiest to understand for the *perception of stationarity*, which was mentioned in Section 2.3. Fixate on a nearby object and yaw your head to the left. Your eyes should

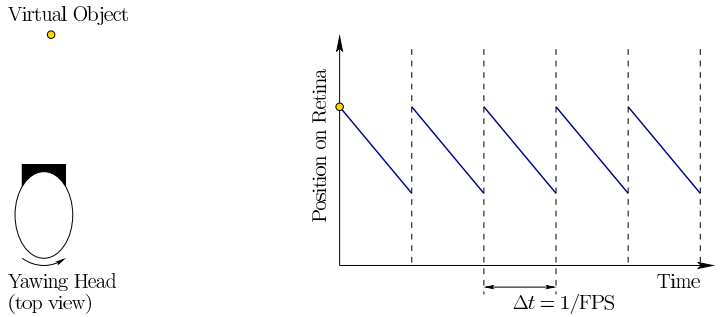


Figure 6.18: A problem with *perception of stationarity* under stroboscopic apparent motion: The image of a feature slips across the retina in a repeating pattern as the VOR is performed.

then rotate to the right to maintain the object in a fixed location on the retina, due to the VOR (Section 5.3). If you do the same while wearing a VR headset and fixating on an object in the virtual world, then the image of the object needs to shift across the screen while you turn your head. Assuming that the pixels instantaneously change at each new frame time, the image of the virtual object will slip across the retina as shown in Figure 6.18. The result is a kind of *judder* in which the object appears to be wobbling from side to side with high frequency but small amplitude.

The problem is that the image is fixed on the screen for too long while it is supposed to be moving continuously across the screen. At 60 FPS, it is fixed for 16.67ms during each frame (in an idealized setting). If the screen is instead turned on for only one or two milliseconds for each frame, and then made black during the remaining times, then the amount of retinal image slip is greatly reduced. This display mode is called *low persistence*, and is shown in Figure 6.19(a). The short amount of time that the display is illuminated is sufficient for the photoreceptors to collect enough photons to cause the image to be perceived. The problem is that at 60 FPS in low-persistence mode, flicker is perceived, which can lead to fatigue or headaches. This can be easily perceived at the periphery in a bright scene in the Samsung Gear VR headset. If the frame rate is increased to 90 FPS or above, then the adverse side effects of flicker subside for nearly everyone. If the frame rate is increased to 500 FPS or beyond, then it would not even need to flicker, as depicted in Figure 6.19(b).

One final point is that fast pixel switching speed is implied in the Figure 6.19. In a modern OLED display panel, the pixels can reach their target intensity values in less than 0.1ms. However, many LCD displays change pixel values much more slowly. The delay to reach the target intensity may be as long as 20ms, depending on the amount and direction of intensity change. In this case, a fixed virtual object appears to smear or blur in the direction of motion. This was easily observable in the Oculus Rift DK1, which used an LCD display panel.

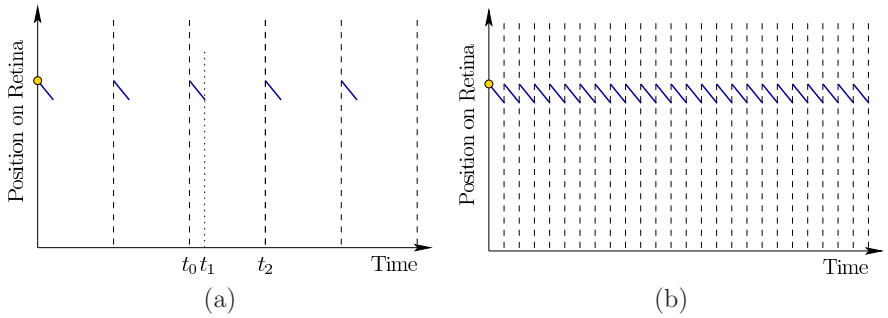


Figure 6.19: An engineering solution to reduce retinal image slip: (a) Using *low persistence*, the display is lit for a short enough time to trigger photoreceptors ($t_1 - t_0$) and then blanked for the remaining time ($t_2 - t_1$). Typically, $t_1 - t_0$ is around one to two milliseconds. (b) If the frame rate were extremely fast (at least 500 FPS), then the blank interval would not be needed.

6.3 Perception of Color

What makes an object "purple", "pink", or "gray"? Color perception is unusual because it is purely the result of our visual physiology and neural structures, rather than something that can be measured in the physical world. In other words, "It's all in your head." If two people have comparable color perception systems, then they can discuss colors using commonly agreed upon names as they perceive an object as having the same color. This contrasts other perception topics such as motion, depth, and scale, all of which correspond to measurable activity in the surrounding world. The size of an object or the speed of its motion relative to some frame could be determined by instrumentation. Humans would be forced to agree on the numerical outcomes regardless of how their individual perceptual systems are functioning.

The dress Figure 6.20 illustrates this point with the *dress color illusion*. It was worn by Cecilia Bleasdale and became an Internet meme when millions of people quickly began to argue about the color of the dress. Based on the precise combination of colors and lighting conditions, its apearance fell on the boundary of what human color perceptual systems can handle. About 57% perceive it as blue and black (correct), 30% percent perceive it as white and gold, 10% perceive blue and brown, and 10% could switch between perceiving any of the color combinations [73].

Dimensionality reduction Recall from Section 4.1 that light energy is a jumble of wavelengths and magnitudes that form the spectral power distribution. Figure 4.6 provided an illustration. As we see objects, the light in the environment is

Figure 6.20: In 2014, this dress photo became an Internet sensation as people were unable to agree upon whether it was "blue and black" or "white and gold", which are strikingly different perceptions of color.

reflected off of surfaces in a wavelength-dependent way according to the spectral distribution function (Figure 4.7). As the light passes through our eyes and is focused onto the retina, each photoreceptor receives a jumble of light energy that contains many wavelengths. Since the power distribution is a function of wavelength, the set of all possible distrubutions is a *function space*, which is generally infinite-dimensional. Our limited hardware cannot possibly sense the entire function. Instead, the rod and cone photoreceptors sample it with a bias toward certain target wavelengths, as was shown in Figure 5.3 of Section 5.1. The result is a well-studied principle in engineering called *dimensionality reduction*. Here, the infinite-dimensional space of power distributions collapses down to a 3D *color space*. It is no coincidence that we have precisely three types of cones, and that our RGB displays target the same colors as the photoreceptors.

Yellow = Green + Red To help understand this reduction, consider the perception of "yellow". According to the visible light spectrum (Figure 4.5), a wavelength of about 580nm. Suppose we had a pure light source that shines light of exactly 580nm wavelength onto our retinas with no other wavelengths. The spectral distribution function would have a spike at 580nm and be zero everywhere else. If we had a cone with peak detection at 580nm and no sensitivity to other wavelengths, then it would perfectly detect yellow. Instead, we perceive yellow by activation of both green and red cones because their sensitivity regions (Figure 5.3) include 580nm. It should then be possible to generate the same photoreceptor response by sending a jumble of light that contains precisely two wavelengths: 1) Some "green" at 533nm, and 2) some "red" at 564nm. If the magnitudes of green and red are tuned so that the green and red cones activate in the same way as

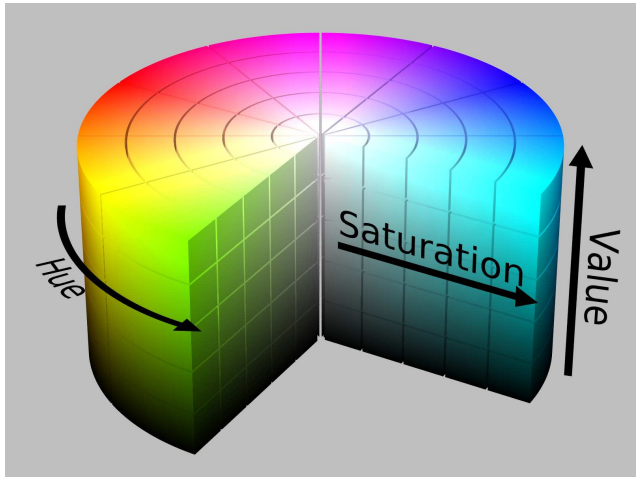


Figure 6.21: One representation of the HSV color space, which involves three paramters: hue, saturation, and value (brightness). (Figure by Wikipedia user SharkD.)

they did for pure yellow, then it becomes impossible for our visual system to distinguish the green/red mixture from pure yellow. Both are perceived as "yellow". This matching of colors from red, green and blue components is called *metamerism*. Such a blending is precisely what is done on a RGB display to produce yellow. Suppose the intensity of each color ranges from 0 (dark) to 255 (bright). Red is produced by RGB= $(255, 0, 0)$, and green is RGB= $(0, 255, 0)$. These each activate one LED (or LCD) color, thereby producing a pure red or green. If both are turned on, then yellow is perceived. Thus, yellow is RGB= $(255, 255, 0)$.

Color spaces For convenience, a parameterized *color space* is often defined. One of the most common in computer graphics is called *HSV*, which has the following three components (Figure 6.21):

- The *hue*, which corresponds directly to the perceived color, such as "red" or "green".
- The *saturation*, which is the purity of the color. In other words, how much energy is coming from wavelengths other than the wavelength of the hue?
- The *value*, which corresponds to the brightness.

There are many methods to scale the HSV coordinates, which distort the color space in various ways. The RGB values could alternatively be used, but are sometimes more difficult for people to interpret.

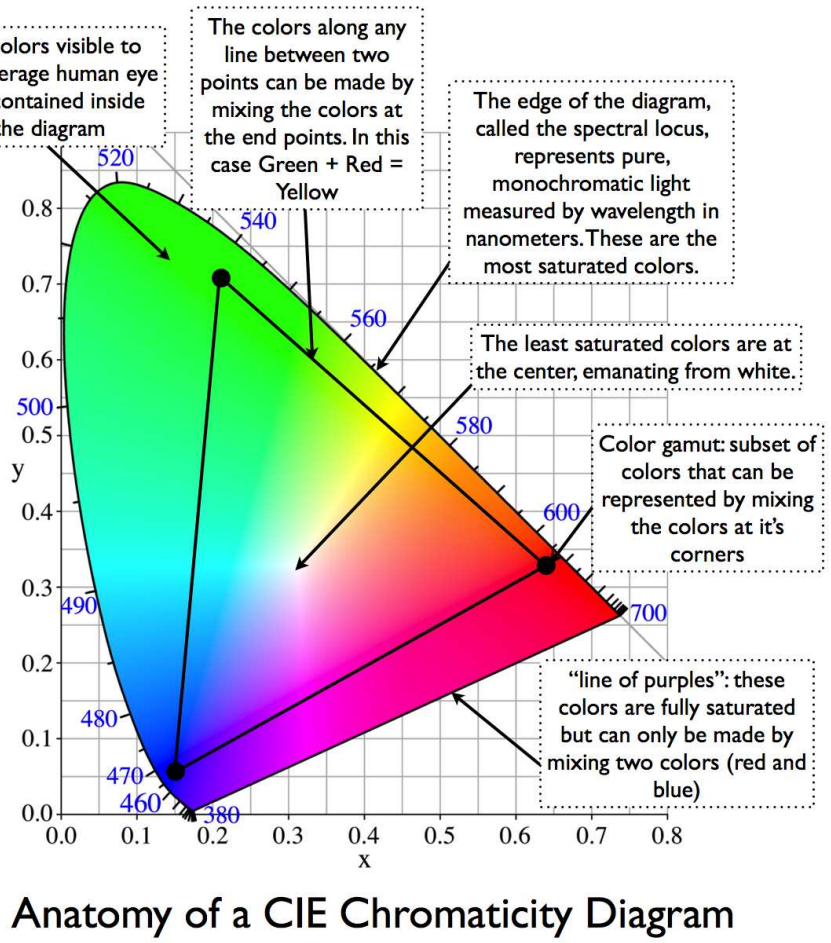


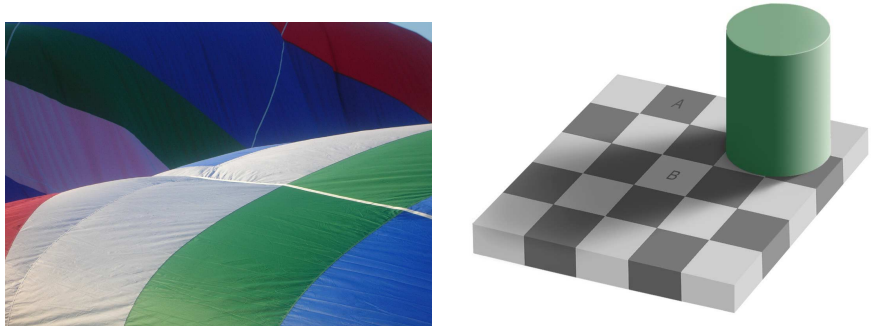
Figure 6.22: 1931 CIE color standard with RGB triangle. This representation is correct in terms of distances between perceived colors. (Figure by Jeff Yurek.)

It would be ideal to have a representation in which the distance between two points corresponds to the amount of perceptual difference. In other words, as two points are further apart, our ability to distinguish them is increased. The distance should correspond directly to the amount of distinguishability. Vision scientists designed a representation to achieve this, resulting in the 1931 CIE color standard shown in Figure 6.22. Thus, the CIE is considered to be undistorted from a perceptual perspective. It is only two-dimensional because it disregards the brightness component, which is independent of color perception according to color matching experiments [92].

Mixing colors Suppose that we have three pure source of light, as in that produced by an LED, in red, blue, and green colors. We have already discussed how to produce yellow by blending red and green. In general, most perceptible colors can be matched by a mixture of three. This is called *trichromatic theory* (or *Young-Helmholtz theory*). A set of colors that achieves this is called *primary colors*. Mixing all three evenly produces perceived *white light*, which on a display is achieved as RGB= $(255, 255, 255)$. Black is the opposite: RGB= $(0, 0, 0)$. Such light mixtures follow a linearity property. Suppose primary colors are used to perceptually match power distributions of two different light sources. If the light sources are combined, then their intensities of the primary colors need only to be added to obtain the perceptual match for the combination. Furthermore, the overall intensity can be scaled by multiplying the red, green, and blue components without affecting the perceived color. Only the perceived brightness may be changed.

The discussion so far has focused on *additive mixtures*. When mixing paints or printing books, colors mix subtractively because the spectral reflectance function is being altered. When starting with a white canvass or sheet of paper, virtually all wavelengths are reflected. Painting a green line on the page prevents all wavelengths other than green from being reflected at that spot. Removing all wavelengths results in black. Rather than using RGB components, printing presses are based on CMYK, which correspond to cyan, magenta, yellow, and black. The first three are pairwise mixes of the primary colors. A black component is included to reduce the amount of ink wasted by using the other three colors to subtractively produce black. Note that the targeted colors are observed only if the incoming light contains the targeted wavelengths. The green line would appear green under pure, matching green light, but might appear black under pure blue light.

Constancy The dress in Figure 6.20 showed an extreme case that results in color confusion across people due to the strange lighting conditions. Ordinarily, human color perception is surprisingly robust to the source of color. A red shirt appears to be red whether illuminated under indoor lights at night or in direct sunlight. These correspond to vastly different cases in terms of the spectral power distribution that reaches the retina. Our ability to perceive an object as having the same color over a wide variety of lighting conditions is called *color constancy*.



(a) (b)

Figure 6.23: (a) The perceived hot air balloon colors are perceived the same regardless of the portions that are in direct sunlight or in a shadow. (Figure by Wikipedia user Shanta.) (b) The *checker shadow illusion* from Section 2.3 is explained by the *lightness constancy* principle as the shadows prompt compensation of the perceived lightness. (Figure by Adrian Pingstone.)

Several perceptual mechanisms allow this to happen. One of them is *chromatic adaptation*, which results in a shift in perceived colors due to prolonged exposure to specific colors. Another factor in the perceived color is the expectation from the colors of surrounding objects. Furthermore, memory about objects are usually colored in the environment biases our interpretation.

The constancy principle also appears without regard to particular colors. Our perceptual system also maintains *lightness constancy* so that the overall brightness levels appear to be unchanged, even after lighting conditions are dramatically altered; see Figure 6.23(a). Under the *ratio principle* theory, only the ratio of reflectances between objects in a scene are perceptually maintained, whereas the overall amount of reflected intensity is not perceived. Further complicating matters, our perception of object lightness and color are maintained as the scene contains uneven illumination. A clear example is provided by by shadows cast by one object onto another. Our perceptual system accounts for the shadow and adjusts our perception of the object shade or color. The checker shadow illusion shown in Figure 6.23 is caused by this compensation due to shadows.

Display issues Displays generally use RGB lights to generate the palette of colors and brightess. Recall Figure 5.22, which showed the subpixel mosaic of individual component colors for some common displays. Usually, the intensity of each R, G, and B value is set by selecting an integer from 0 to 255. This is a severe limitation on the number of brightness levels, as stated in Section 5.4. One cannot hope to densely cover all seven orders of magnitude of perceptible light intensity. One way to enhance the amount of contrast over the entire range is to perform

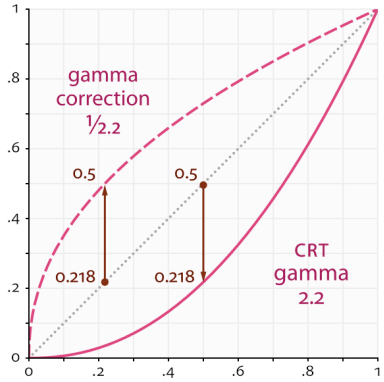


Figure 6.24: Gamma correction is used to span more orders of magnitude in spite of a limited number of bits. The transformation is $v' = cv^{\gamma}$, in which c is constant (usually $c = 1$) and γ controls the nonlinearity of the correction or distortion.

gamma correction. In most displays, images are encoded with a gamma of about 0.45 and decoded with a gamma of 2.2.

Another issue is that the set of all available colors lies inside of the triangle formed by R, G, and B vertices. This limitation is shown for the case of the *sRGB* standard in Figure 6.22. Most the CIE is covered, but many colors that humans are capable of perceiving cannot be generated on the display.

6.4 Combining Sources of Information

Throughout this chapter, we have seen perceptual processes that combine information from multiple sources. These could be cues from the same sense, as in the numerous monocular cues used to judge depth. Perception may also combine information from two or more senses. For example, people typically combine both visual and auditory cues when speaking face to face. Information from both sources makes it easier to understand someone, especially if there is significant background noise. We have also seen that information is integrated over time, as in the case of saccades being employed to fixate on several object features. Finally, our memories and general expectations about the behavior of the surrounding world bias our conclusions. Thus, information is integrated from prior expectations and the reception of many cues, which may come from different senses at different times.

Statistical decision theory provides a useful and straightforward mathematical model for making choices that incorporate prior biases and sources of relevant, observed data. It has been applied in many fields, including economics, psychology, signal processing, and computer science. One key component is *Bayes' rule*, which specifies how the *prior* beliefs should be updated in light of new observations, to

obtain *posterior* beliefs. More formally, the "beliefs" are referred as *probabilities*. If the probability takes into account information from previous information, it is called a *conditional probability*. There is no room to properly introduce probability theory here; only the basic ideas are given to provide some intuition without the rigor. For further study, find an online course or classic textbook (for example, [123]).

Let

$$H = \{h_1, h_2, \ldots, h_n\} \tag{6.1}$$

be a set of *hypotheses* (or interpretations). Similarly, let

$$C = \{c_1, c_2, \ldots, c_m\} \tag{6.2}$$

C be a set of possible outputs of a *cue detector*. For example, the cue detector might output the eye color of a face that is currently visible. In this case C is the set of possible colors:

$$C = \{\text{BROWN}, \text{BLUE}, \text{GREEN}, \text{HAZEL}\}. \tag{6.3}$$

Modeling a face recognizer, H would correspond to the set of people familiar to the person.

We want to calculate probability values for each of the hypotheses in H. Each probability value must lie between 0 to 1, and the sum of the probability values for every hypothesis in H must sum to one. Before any cues, we start with an assignment of values called the *prior distribution*, which is written as $P(h)$. The "P" denotes that it is a probability function or assignment; $P(h)$ means that an assignment has been applied to every h in H. The assignment must be made so that

$$P(h_1) + P(h_2) + \cdots + P(h_n) = 1, \tag{6.4}$$

and $0 \leq P(h_i) \leq$ for each i from 1 to n.

The prior probabilities are generally distributed across the hypotheses in a diffuse way; an example is shown in Figure 6.25(a). The likelihood of any hypothesis being true before any cues is proportional to its frequency of occurring naturally, based on evolution and the lifetime of experiences of the person. For example, if you open your eyes at a random time in your life, what is the likelihood of seeing a human being versus a wild boar?

Under normal circumstances (not VR!), we expect that the probability for the correct interpretation will rise as cues arrive. The probability of the correct hypothesis should pull upward toward 1, effectively stealing probability mass from the other hypotheses, which pushes their values toward 0; see Figure 6.25(b). A "strong" cue should lift the correct hypothesis upward more quickly than a "weak" cue. If a single hypothesis has a probability value close to 1, then the distribution is considered *peaked*, which implies high confidence; see Figure 6.25(c). In the other direction, inconsistent or incorrect cues have the effect of diffusing the probability across two or more hypotheses. Thus, the probability of the correct hypothesis

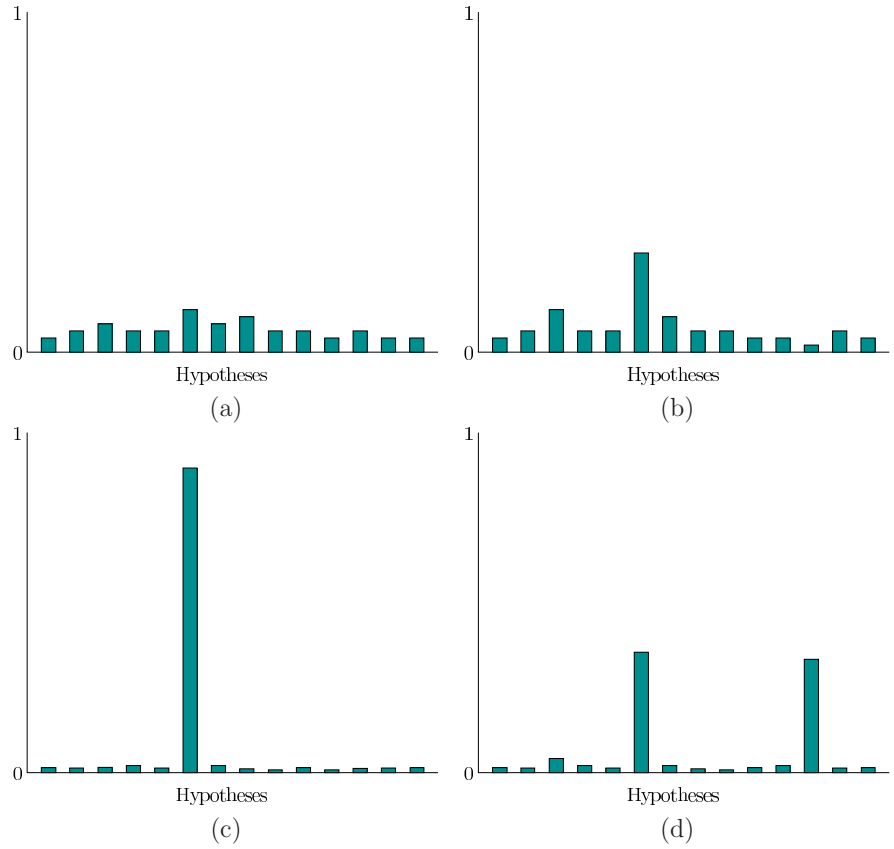


Figure 6.25: Example probability distributions: (a) A possible prior distribution. (b) Preference for one hypothesis starts to emerge after a cue. (c) A peaked distribution, which results from strong, consistent cues. (d) Ambiguity may result in two (or more) hypotheses that are strongly favored over others; this is the basis of multistable perception.

may be lowered as other hypotheses are considered plausible and receive higher values. It may also be possible that two alternative hypotheses remain strong due to ambiguity that cannot be solved from the given cues; see Figure 6.25(d).

To take into account information from a cue, a *conditional distribution* is defined, which is written as $P(h \mid c)$. This is spoken as "the probability of h given c." This corresponds to a probability assignment for all possible combinations of hypotheses and cues. For example, it would include $P(h_2 \mid c_5)$, if there are at least two hypotheses and five cues. Continuing our face recognizer, this would look like $P(\text{BARACK OBAMA} \mid \text{BROWN})$, which should be larger than $P(\text{BARACK OBAMA} \mid \text{BLUE})$ (he has brown eyes).

We now arrive at the fundamental problem, which is to calculate $P(h \mid c)$ after the cue arrives. This is accomplished by *Bayes' rule*:

$$P(h \mid c) = \frac{P(c \mid h)P(h)}{P(c)}. \tag{6.5}$$

The denominator can be expressed as

$$P(c) = P(c \mid h_1)P(h_1) + P(c \mid h_2)P(h_2) + \cdots + P(c \mid h_n)P(h_n), \tag{6.6}$$

or it can be ignored it as a normalization constant, at which point only relative likelihoods are calculated instead of proper probabilities.

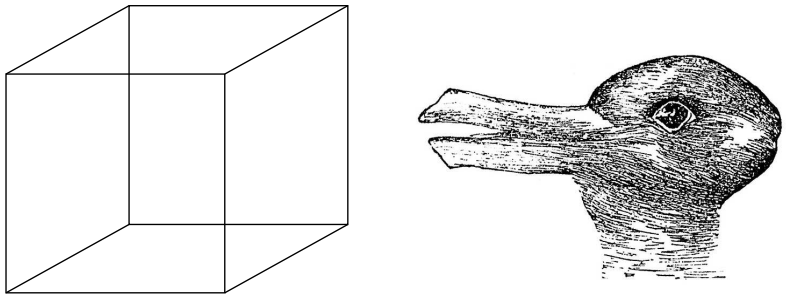
The only thing accomplished by Bayes' rule was to express $P(h \mid c)$ in terms of the prior distribution $P(h)$ and a new conditional distribution $P(c \mid h)$. The new conditional distribution is easy to work with in terms of modeling. It characterizes the likelihood that each specific cue will appear given that the hypothesis is true.

What if information arrives from a second cue detector? In this case, (6.5) is applied again, but $P(h \mid c)$ is now considered the prior distribution with respect to the new information. Let $D = \{d_1, d_2, \ldots, d_k\}$ represent the possible outputs of the new cue detector. Bayes' rule becomes

$$P(h \mid c, d) = \frac{P(d \mid h)P(h \mid c)}{P(d|c)}. \tag{6.7}$$

Above, $P(d \mid h)$ makes what is called a *conditional independence* assumption: $P(d \mid h) = P(d \mid h, c)$. This is simpler from a modeling perspective. More generally, all four conditional parts of (6.7) should contain c because it is given before d arrives. As information from even more cues becomes available, Bayes' rule is applied again as many times as needed. One difficulty that occurs in practice and modeled here is *cognitive bias*, which corresponds to numerous ways in which humans make irrational judgments in spite of the probabilistic implications of the data.

Multistable perception In some cases, our perceptual system may alternate between two or more conclusions. This is called *multistable perception*, for which the special case of two conclusions is called *bistable perception*. Figure 6.26(a)



(a) (b)

Figure 6.26: (a) The Necker cube, studied in 1832 by Swiss crystallographer Louis Albert Necker. (b) The *rabbit duck illusion*, from the 23 October 1892 issue of Fliegende Blätter.

shows two well-known examples. For the *Necker cube*, it is ambiguous which cube face that is parallel to the viewing plane is in the foreground. It is possible to switch between both interpretations, resulting in bistable perception. Figure 6.26(b) shows another example, in which may see a rabbit or a duck at various times. Another well-known example is called the *spinning dancer illusion* by Nobuyuki Kayahara. In that case, the silhouette of a rotating dancer is shown and it is possible to interpret the motion as clockwise or counterclockwise.

McGurk effect The *McGurk effect* is an experiment that clearly indicates the power of integration by mixing visual and auditory cues. A video of a person speaking is shown with the audio track dubbed so that the spoken sounds do not match the video. Two types of illusions were then observed. If "ba" is heard and "ga" is shown, then most subjects perceive "da" being said. This corresponds to a plausible fusion of sounds that explains the mismatch, but does not correspond to either original cue. Alternatively, the sounds may combine to produce a perceived "bga" in the case of "ga" on the sound track and "ba" on the visual track.

Implications for VR Not all senses are taken over by VR. Thus, conflict will arise because of mismatch between the real and virtual worlds. As stated several times, the most problematic case of this is vection, which is a sickness-causing conflict between visual and vestibular cues arising from apparent self motion in VR while remaining stationary in the real world; see Section 8.4. As another example of mismatch, the user's body may sense that it is sitting in a chair, but the VR experience may involve walking. There would then be a height mismatch between the real and virtual worlds, in addition to mismatches based on proprioception and tactile sensing. In addition to mismatches among the senses, imperfections in the VR hardware, software, content, and interfaces cause inconsistencies in

comparison with real-world experiences. The result is that incorrect or untended interpretations may arise. Even worse, such inconsistencies may increase fatigue as our neural structures use more energy to interpret the confusing combination. In light of the McGurk effect, it is easy to believe that many unintended interpretations or perceptions may arise from a VR system that does not provide perfectly consistent cues.

VR is also quite capable of generating new multistable perceptions. One example, which actually occurred in the VR industry, involved designing a popup menu. Suppose the user is in dark environment and a large menu comes rushing up to them. The user may perceive one of two cases: 1) the menu approaches the user, or 2) the user is rushing up to the menu. The vestibular sense should be enough to resolve whether the user is moving, but the visual sense is overpowering. Prior knowledge about which is happening helps yield the correct perception. Unfortunately, if the wrong interpretation is made, then VR sickness in increased due to the sensory conflict. This, our perceptual system could by tricked into an interpretation that is worse for our health!

Further Reading

Retinal image size: [49]

Muller-Lyer Illusion:

Motion detection circuitry: Barlow, Hill, 1963; [96] [117]

Wagon wheel continuous illumination: [127]

History of film: [17]

Many optical illusions shown and explained: [108]

Silent film speed: [21]

LCD backlight scanning

The dress analysis: [73]

For more details on decision theory: Chapter 9 of [77].

McGurk effect: [93]

Bayesian analysis and decision theory: [119].

Phi phenomenon and beta movement: [158, 140].

Adaptation to conflicts in VR: [157]

Chapter 7

Visual Rendering

Chapter Status	Taken from *Virtual Reality*, S. M. LaValle
	This online chapter is not the final version! Check **http://vr.cs.uiuc.edu/** for information on the latest draft version. This draft was compiled on June 29, 2016.

This chapter explains visual rendering, which specifies what the visual display will show through an interface to the virtual world generator (VWG). Chapter 3 already provided the mathematical parts, which express *where* the objects in the virtual world should appear on the screen. This was based on geometric models, rigid body transformations, and viewpoint transformations. We next need to determine *how* these objects should appear, based on knowledge about light propagation, visual physiology, and visual perception. These were the topics of 4, 5, and 6, respectively. Thus, visual rendering is a culmination of everything covered so far.

Sections 7.1 and 7.2 cover the basic concepts; these are considered the core of computer graphics, but VR-specific issues also arise. They mainly address the case of rendering for virtual worlds that are formed synthetically. Section 7.1 explains how to determine the light that should appear at a pixel based on light sources and the reflectance properties of materials that exist purely in the virtual world. Section 7.2 explains rasterization methods, which efficiently solve the rendering problem and are widely used in specialized graphics hardware, called GPUs. Section 7.3 addresses VR-specific problems that arise from imperfections in the optical system. Section 7.4 focuses on latency reduction, which is critical to VR so that objects appear in the right place at the right time. Otherwise, many side effects could arise, such as VR sickness, fatigue, adaptation to the flaws, or simply having an unconvincing experience. Finally, Section 7.5 explains rendering for captured, rather than synthetic, virtual worlds. This covers VR experiences that are formed from panoramic photos and videos.

7.1 Ray Tracing and Shading Models

Suppose that a virtual world has been defined in terms of triangular primitives. Furthermore, a virtual eye has been placed in the world to view it from some particular position and orientation. Using the full chain of transformations from Chapter 3, the location of every triangle is correctly positioned onto a virtual screen (this was depicted in Figure 3.13). The next steps are to determine which screen pixels are covered by the transformed triangle and then illuminate them according to the physics of the virtual world.

An important condition must also be checked: For each pixel, is the triangle even *visible* to the eye, or will it be blocked by part of another triangle? This classic *visibility computation* problem dramatically complicates the rendering process. The general problem is to determine for any pair of points in the virtual world, whether the line segment that connects them intersects with any objects (triangles). If an intersection occurs, then the line-of-sight visibility between the two points is blocked. The main difference between the two major families of rendering methods is due to the how visibility is handled.

Object-order versus image-order rendering For rendering, we need to consider all combinations of objects and pixels. This suggests a nested loop. One way to resolve the visibility is to iterate over the list of all triangles and attempt to render each one to the screen. This is called *object-order rendering*, and is the main topic of Section 7.2. For each triangle that falls into the field of view of the screen, the pixels are updated *only if* the corresponding part of the triangle is closer to the eye than any triangles that have been rendered so far. In this case, the outer loop iterates over triangles whereas the inner loop iterates over pixels. The other family of methods is called *image-order rendering*, and it reverses the order of the loops: Iterate over the image pixels and for each one, determine which triangle should influence its RGB values. To accomplish this, the path of light waves that would enter each pixel is traced out through the virtual environment. This method will be covered first, and many of its components apply to object-order rendering as well.

Ray tracing To calculate the RGB values at a pixel, a *viewing ray* is drawn from the focal point through the center of the pixel on a virtual screen that is placed in the virtual world; see Figure 7.1. The process is divided into two phases:

1. *Ray casting*, in which the viewing ray is defined and its nearest point of intersection among all triangles in the virtual world is calculated.

2. *Shading*, in which the pixel RGB values are calculated based on lighting conditions and material properties at the intersection point.

The first step is based entirely on the virtual world geometry. The second step uses simulated physics of the virtual world. Both the material properties of objects

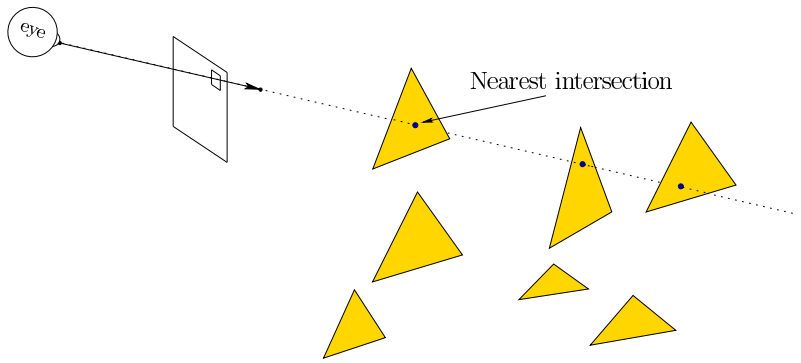


Figure 7.1: The first step in a ray tracing approach is called *ray casting*, which extends a viewing ray that corresponds to a particular pixel on the image. The ray starts at the focal point, which is the origin after the eye transform T_{eye} has been applied. The task is to determine what part of the virtual world model is visible. This is the closest intersection point of the viewing ray and the set of all triangles.

and the lighting conditions are artificial, and are chosen to produce the desired effect, whether realism or fantasy. Remember that the ultimate judge is the user, who interprets the image through perceptual processes.

Ray casting Calculating the first triangle hit by the viewing ray after it leaves the image pixel (Figure 7.1) is straightforward if we neglect the computational performance. Starting with the triangle coordinates, focal point, and the ray direction (vector), the closed-form solution involves basic operations from analytic geometry, including dot products, cross products, and the plane equation []. For each triangle, it must be determined whether the ray intersects it. If not, then the next triangle is considered. If it does, then the intersection is recorded as the candidate solution only if it is closer than the closest intersection encountered so far. After all triangles have been considered, the closest intersection point will be found. Although this is simple, it is far more efficient to arrange the triangles into a 3D data structure. Such structures are usually hierarchical so that many triangles can be eliminated from consideration by quick coordinate tests. Popular examples include BSP-trees and Bounding Volume Hierarchies [25]. Algorithms that sort geometric information to obtain greater efficiently generally fall under computational geometry [33]. In addition to eliminating many triangles from quick tests, many methods of calculating the ray-triangle intersection has been developed to reduce the number of operations. One of the most popular is the *Möller-Trumbore intersection algorithm* [101].

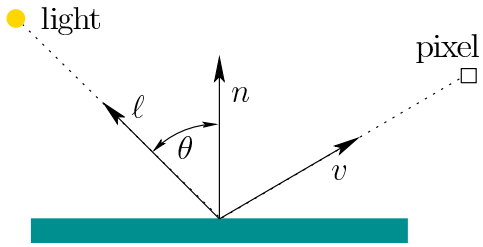


Figure 7.2: In the *Lambertian shading* model, the light reaching the pixel depends on the angle θ between the incoming light and the surface normal, but is independent of the viewing angle.

Lambertian shading Now consider lighting each pixel and recall the basic behavior of light from Section 4.1. The virtual world simulates the real-world physics, which includes the spectral power distribution and spectral reflection function. Suppose that a point-sized light source is placed in the virtual world. Using the trichromatic theory from Section 6.3, its spectral power distribution is sufficiently represented by R, G, and B values. If the viewing ray hits the surface as shown in Figure 7.2, then how should the object appear? Assumptions about the spectral reflection function are taken into account by a *shading model*. The simplest case is *Lambertian shading*, for which the angle that the viewing ray strikes the surface is independent of the resulting pixel R, G, B values. This corresponds to the case of diffuse reflection, which is suitable for a "rough" surface (recall Figure 4.4). All that matters is the angle θ that the surface makes with respect to the light source.

Let n be the outward surface normal and let ℓ be a vector from the surface intersection point to the light source. Assume both n and ℓ are unit vectors. The dot product $n \cdot \ell = \cos\theta$ yields the amount of attenuation (between 0 and 1) due to the tilting of the surface relative to the light source. Think about how the effective area of the triangle is reduced due to its tilt. A pixel under the *Lambertian shading* model is illuminated as

$$\begin{aligned} R &= d_R I_R \max(0, n \cdot \ell) \\ G &= d_G I_G \max(0, n \cdot \ell) \\ B &= d_B I_B \max(0, n \cdot \ell), \end{aligned} \tag{7.1}$$

in which (d_R, d_G, d_B) represents the spectral reflectance property of the material (triangle) and (I_r, I_G, I_R) is represents the spectral power distribution of the light source. Under the typical case of white light, $I_R = I_G = I_B$. For a white or gray material, we would also have $d_R = d_G = d_B$.

Using vector notation, (7.1) can be compressed into

$$L = dI \max(0, n \cdot \ell) \tag{7.2}$$

in which $L = (R, G, B)$, $d = (d_R, d_G, d_B)$, and $I = (I_R, I_G, I_B)$. Each triangle is assumed to be on the surface of an object, rather than the object itself. Therefore, if the light source is behind the triangle, then the triangle should not be illuminated

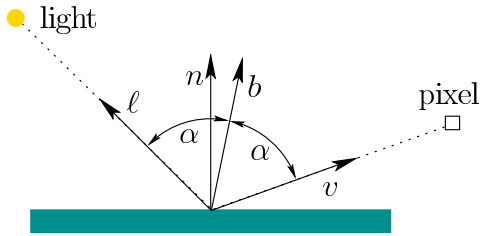


Figure 7.3: In the *Blinn-Phong shading* model, the light reaching the pixel depends on the angle between the normal n and the bisector b of the ℓ and v. If $n = b$, then ideal reflection is obtained, as in the case of a mirror.

because it is facing away from the light (it cannot be lit from behind). To handle this case, the max function appears in (7.2) to avoid $n \cdot \ell < 0$.

Blinn-Phong shading Now suppose that the object is "shiny". If it were a perfect mirror, then all of the light from the source would be reflected to the pixel only if they are perfectly aligned; otherwise, no light would reflect at all. Such full reflection would occur if v and ℓ are the same angle with respect to n. What if the two angles are close, but do not quite match? The *Blinn-Phong shading* model proposes that some amount of light is reflected, depending on the amount of surface shininess and the difference between v and ℓ [14]. See Figure 7.3. The *bisector* b is the vector obtained by averaging ℓ and v:

$$b = \frac{\ell + v}{\|\ell + v\|}. \tag{7.3}$$

Using the compressed vector notation, the *Blinn-Phong shading* model sets the RGB pixel values as

$$L = dI \max(0, n \cdot \ell) + sI \max(0, n \cdot b)^x. \tag{7.4}$$

This additively takes into account shading due to both diffuse and specular components. The first term is just the Lambertian shading model, (7.2). The second component causes increasing amounts of light to be reflected as b becomes closer to n. The exponent x is a material property that expresses the amount of surface shininess. A lower value, such as $x = 100$, results in a mild amount of shininess, whereas $x = 10000$ would make the surface almost like a mirror. This shading model does not correspond directly to the physics of the interaction between light and surfaces. It is merely a convenient and efficient heuristic, but widely used in computer graphics.

Ambient shading Another heuristic is *ambient shading*, which causes an object to glow without being illuminated by a light source. This lights surfaces that fall

Figure 7.4: A *Bidirectional Reflectance Distribution Function*, or *BRDF*, meticulously specifies the ratio of incoming and outgoing light energy for all possible perspectives.

into the shadows of all lights; otherwise, they would be completely black. In the real world this does not happen light interreflects between objects to illuminate an entire environment. Such propagation has not been taken into account in the shading model so far, thereby requiring a hack to fix it. Adding ambient shading yields

$$L = dI \max(0, n \cdot \ell) + sI \max(0, n \cdot b)^x + L_a, \tag{7.5}$$

in which L_a is the ambient light component.

Multiple light sources Typically, the virtual world contains multiple light sources. In this case, the light from each is combined additively at the pixel. The result for N light sources is

$$L = L_a + \sum_{i=1}^{N} dI_i \max(0, n \cdot \ell_i) + sI_i \max(0, n \cdot b_i)^x, \tag{7.6}$$

in which I_i, ℓ_i, and b_i correspond to each source.

BRDFs The shading models presented so far are in widespread use due to their simplicity and efficiency, even though they neglect most of the physics. To account for shading in a more precise and general way, a *bidirectional reflectance distribution function* (*BRDF*) is constructed; see Figure 7.4. The θ_i and θ_r parameters represent the angles of light source and viewing ray, respectively, with respect to the surface. The ϕ_i and ϕ_r parameters range from 0 to 2π and represent the angles made by the light and viewing vectors when looking straight down on the surface (the vector n would point at your eye).

The BRDF is a function of the form

$$f(\theta_i, \phi_i, \theta_r, \theta_i) = \frac{\text{radiance}}{\text{irradiance}}, \tag{7.7}$$

in which *radiance* is the light energy reflected from the surface in directions θ_r and ϕ_r and *irradiance* is the light energy arriving at the surface from directions

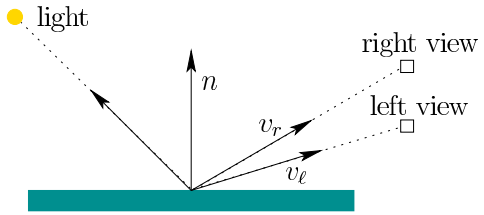


Figure 7.5: Complications emerge with shiny surfaces because the viewpoints are different for the right and left eyes. Using the Blinn-Phong shading model, a specular reflection should have different brightness levels for each eye. It may be difficult to match the effect so that it is consistent with real-world behavior.

θ_i and ϕ_i. These are expressed at a differential level, roughly corresponding to an infinitesimal surface patch. Informally, it is the ratio of the amount of outgoing light to the amount of incoming light at one point on the surface. The previous shading models can be expressed in terms of a simple BRDF. For Lambertian shading, the BRDF is constant because the surface reflects equally in all directions. The BRDF and its extensions can account for much more complex and physically correct lighting effects for a wide variety of surface textures. See Chapter 7 of [3] for extensive coverage.

Global illumination Recall that the ambient shading term (7.5) was introduced to prevent surfaces in the shadows of the light source from appearing black. The computationally intensive but proper way to fix this problem is to calculate how light reflects from object to object in the virtual world. In this way, objects are illuminated *indirectly* from the light that reflects from others, as in the real world. Unfortunately, this effectively turns all object surfaces into potential sources of light. This means that ray tracing must account for multiple reflections. This requires considering piecewise linear paths from the light source to the viewpoint, in which each bend corresponds to a reflection. An upper limit is set on the number of bounces to consider. The simple Lambertian and Blinn-Phong models are often used, but more general BDRFs are also common. Increasing levels of realism can be calculated, but with corresponding increases in computation time.

VR inherits all of the common issues from computer graphics, but also contains unique challenges. Chapters 5 and 6 mentioned the increased resolution and frame rate requirements. This provides strong pressure to reduce rendering complexity. Furthermore, many heuristics that worked well for graphics on a screen may be perceptibly wrong in VR. The combination of high field-of-view, resolution, and stereo images may bring out problems. For example, Figure 7.5 illustrates how differing viewpoints from stereopsis could affect the appearance of shiny surfaces. In general, some rendering artifacts could even contribute to VR sickness. Throughout the remainder of this chapter, complications that are unique to VR will be increasingly discussed.

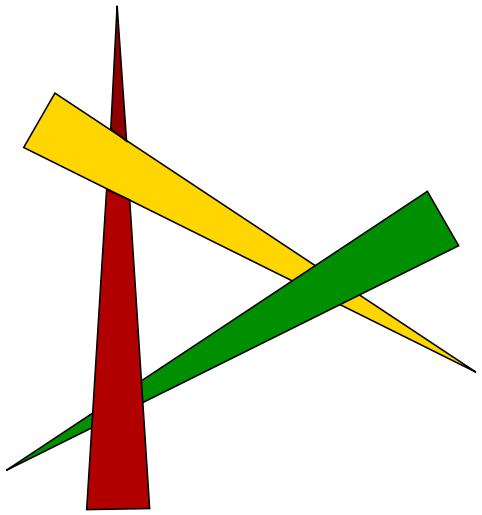


Figure 7.6: Due to the possibility of *depth cycles*, objects cannot be sorted in three dimensions with respect to distance from the observer. Each object is partially in front of one and partially behind another.

7.2 Rasterization

The ray casting operation quickly becomes a bottleneck. For a 1080p image at 90Hz, it would need to be performed over 180 million times per second, and the ray-triangle intersection test would be performed for every triangle (although data structures such as a BSP would quickly eliminate many from consideration). In most common cases, it is much more efficient to switch from such image-order rendering to object-order rendering. The objects in our case are triangles and the resulting process is called *rasterization*, which is the main function of modern graphical processing units (GPUs). In this case, an image is rendered by iterating over every triangle and attempting to color the pixels where the triangle lands on the image. The main problem is that the method must solve the unavoidable problem of determining which part, if any, of the triangle is the closest to the focal point (roughly, the location of the virtual eye).

One way to solve it is to sort the triangles in *depth order* so that the closest triangle is last. This enables the triangles to be drawn on the screen in back-to-front order. If they are properly sorted, then any later triangle to be rendered will rightfully clobber the image of previously rendered triangles at the same pixels. They can be drawn one-by-one while totally neglecting the problem of determining which is nearest. This is known as the *Painter's algorithm*. The main flaw, however, is the potential existence of *depth cycles*, shown in Figure 7.6, in which three or more triangles cannot be rendered correctly in any order by the Painter's algorithm. One possible fix is to detect such cases and split the triangles.

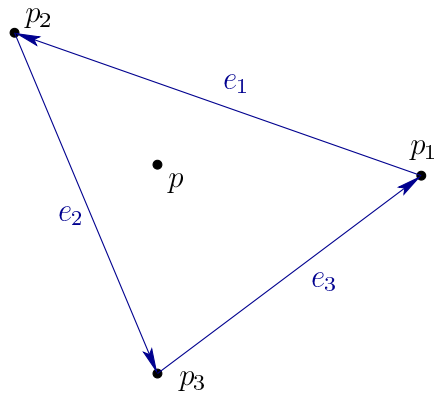


Figure 7.7: Barycentric coordinates specify the location of every point p in a triangle as a weighted average of its vertices p_1, p_2, and p_3.

Depth buffer A simple and efficient method to resolve this problem is to manage the depth problem on a pixel-by-pixel basis by maintaining a *depth buffer* (also called *z-buffer*), which for every pixel records the distance of the triangle from the focal point to the intersection point of the ray that intersects the triangle at that pixel. In other words, if this were the ray casting approach, it would be distance along the ray from the focal point to the intersection point. Using this method, the triangles can be rendered in arbitrary order. The method is also commonly applied to compute the effect of shadows by determining depth order from a light source, rather than the viewpoint. Objects that are closer to the light cast a shadow on further objects.

The depth buffer stores a positive real number (floating point number in practice) at every pixel location. Before any triangles have been rendered, a maximum value (floating-point infinity) is stored at every location to reflect that no surface has yet been encountered at each pixel. At any time in the rendering process, each value in the depth buffer records the distance of the point on the most recently rendered triangle to the focal point, for the corresponding pixel in the image. Initially, all depths are at maximum to reflect that no triangles were rendered yet.

Each triangle is rendered by calculating a rectangular part of the image that fully contains it. This is called a *bounding box*. The box is quickly determined by transforming all three of the triangle vertices to determine the minimum and maximum values for i and j (the row and column indices). An iteration is then performed over all pixels inside of the bounding box to determine which ones lie in inside the triangle and should therefore be rendered. This can be quickly determined by forming the three edge vectors shown in Figure 7.7 as

$$\begin{aligned} e_1 &= p_2 - p_1 \\ e_2 &= p_3 - p_2 \\ e_3 &= p_1 - p_3. \end{aligned} \tag{7.8}$$

The point p lies inside of the triangle if and only if

$$(p - p_1) \times e_1 < 0 \ , \ (p - p_2) \times e_2 < 0 \ , \ (p - p_3) \times e_3 < 0, \tag{7.9}$$

in which $\times$ denotes the standard vector cross product. These three conditions ensure that p is "to the left" of each edge vector.

Barycentric coordinates As each triangle is rendered, information from it is mapped from the virtual world onto the screen. This is usually accomplished using *barycentric coordinates* (see Figure 7.7), which expresses each point in the triangle interior as a weighted average of the three vertices:

$$p = \alpha_1 p_1 + \alpha_2 p_2 + \alpha_3 p_3 \tag{7.10}$$

for which $0 \leq \alpha_1, \alpha_2, \alpha_3 \leq 1$ and $\alpha_1 + \alpha_2 + \alpha_3 = 1$. The closer p is to a vertex p_i, the larger the weight α_1. If p is at the centroid of the triangle, then $\alpha_1 = \alpha_2 = \alpha_3 = 1/3$. If p lies on an edge, then the opposing vertex weight is zero. For example, if p lies on the edge between p_1 and p_2, then $\alpha_3 = 0$. If p lies on a vertex, p_i, then $\alpha_i = 1$, and the other two barycentric coordinates are zero.

The coordinates are calculated using Cramer's rule to solve a resulting linear system of equations. More particularly, let $d_{ij} = e_i \cdot e_j$ for all combinations of i and j. Furthermore, let

$$s = 1/(d_{11}d_{22} - d_{12}d_{12}). \tag{7.11}$$

The coordinates are then given by

$$\begin{aligned} \alpha_1 &= s(d_{22}d_{31} - d_{12}d_{32}) \\ \alpha_2 &= s(d_{11}d_{32} - d_{12}d_{31}) \\ \alpha_3 &= 1 - \alpha_1 - \alpha_2. \end{aligned} \tag{7.12}$$

The same barycentric coordinates may be applied to the points on the model in $\mathbb{R}^3$, or on the resulting 2D projected points (with i and j coordinates) in the image plane. In other words, α_1, α_2, and α_3 refer to the same point on the model both before, during, and after the entire chain of transformations from Section 3.5.

Furthermore, given the barycentric coordinates, the test in (7.9) can be replaced by simply determining whether $\alpha_1 \geq 0$, $\alpha_2 \geq 0$, and $\alpha_3 \geq 0$. If any barycentric coordinate is less than zero, then p must lie outside of the triangle.

Mapping the surface Barycentric coordinates provide a simple and efficient method for linearly interpolating values across a triangle. The simplest case is the propagation of RGB values. Suppose RGB values are calculated at the three triangle vertices using the shading methods of Section 7.1. This results in values (R_i, G_i, B_i) for each i from 1 to 3. For a point p in the triangle with barycentric coordinates $(\alpha_1, \alpha_2, \alpha_3)$, the RGB values for the interior points are calculated as

$$\begin{aligned} R &= \alpha_1 R_1 + \alpha_2 R_2 + \alpha_3 R_3 \\ G &= \alpha_1 G_1 + \alpha_2 G_2 + \alpha_3 G_3 \\ B &= \alpha_1 B_1 + \alpha_2 B_2 + \alpha_3 B_3. \end{aligned} \tag{7.13}$$

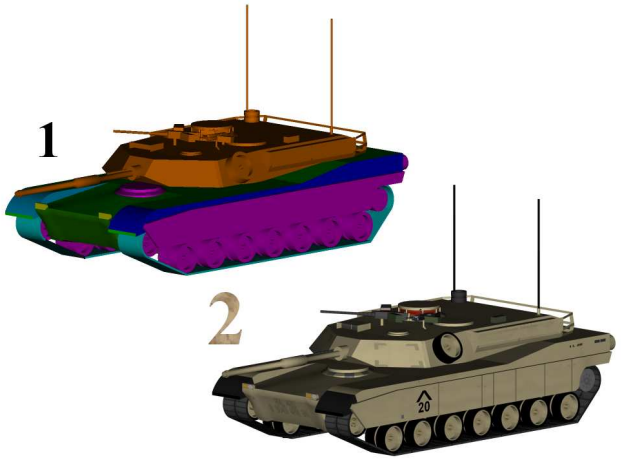


Figure 7.8: Texture mapping: A simple pattern or an entire image can be mapped across the triangles and then rendered in the image to provide much more detail than provided by the triangles in the model. (Figure from Wikipedia.)

The object need not maintain the same properties over an entire triangular patch. With *texture mapping*, a repeating pattern, such as tiles or stripes can be propagated over the surface; see Figure 7.8. More generally, any digital picture can be mapped onto the patch. The barycentric coordinates reference a point inside of the image to be used to influence a pixel. The picture or "texture" is treated as if it were painted onto the triangle; the lighting and reflectance properties are additionally taken into account for shading the object.

Another possibility is *normal mapping*, which alters the shading process by allowing the surface normal to be artificially varied over the triangle, even though geometrically it is impossible. Recall from Section 7.1 that the normal is used in the shading models. By allowing it to vary, simulated curvature can be given to an object. An important case of normal mapping is called *bump mapping*, which makes a flat surface look rough by irregularly perturbing the normals. If the normals appear to have texture, then the surface will look rough after shading is computed.

Aliasing Several artifacts arise due to discretization. Aliasing problems were mentioned in Section 5.4, which result in perceptible staircases in the place of straight lines, due to insufficient pixel density. Figure 7.10(a) shows the pixels selected inside of a small triangle by using (7.9). The point p usually corresponds to the center of the pixel, as shown in Figure 7.10(b). Note that the point may be inside of the triangle while the entire pixel is not. Likewise, part of the pixel might be inside of the triangle while the center is not. You may notice that Figure 7.10 is not entirely accurate due to the subpixel mosaics used in displays (recall

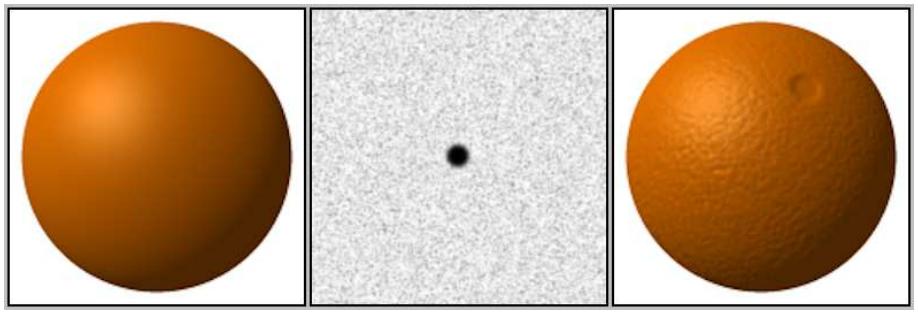


Figure 7.9: Bump mapping: By artificially altering the surface normals, the shading algorithms produce an effect that looks like a rough surface. (Figure by Brian Vibber.)

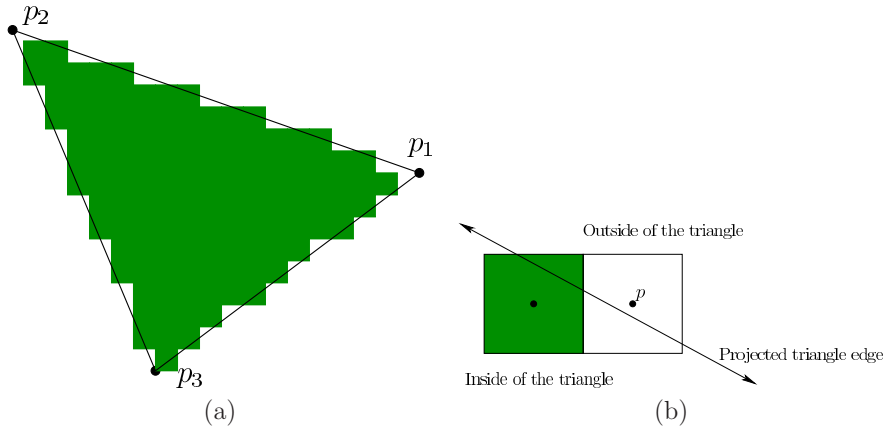


Figure 7.10: (a) The rasterization stage results in aliasing; straight edges appear to be staircases. (b) Pixels are selected for inclusion based on whether their center point p lies inside of the triangle.

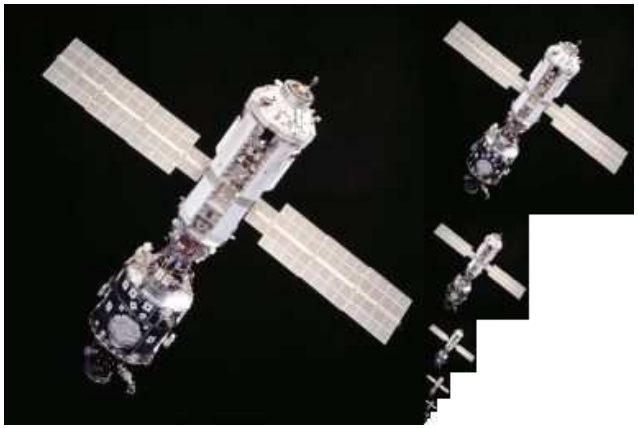
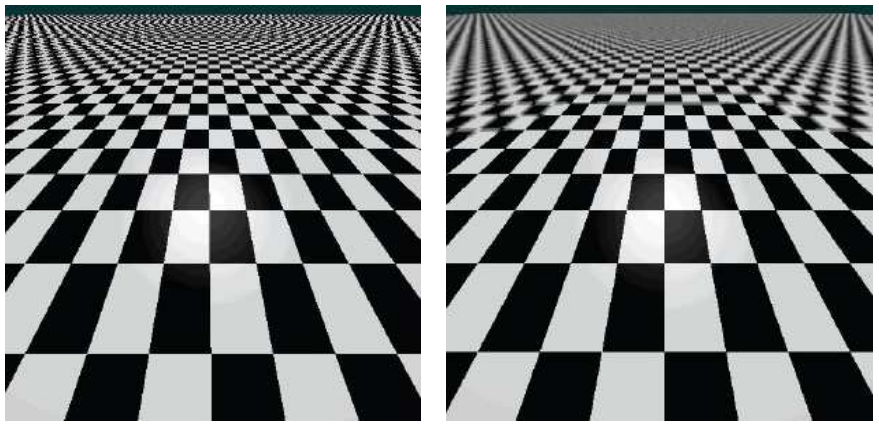


Figure 7.11: A mipmap stores the texture at multiple resolutions so that it can be appropriately scaled without causing signficant aliasing. The overhead for storing the extra image is typically only 1/3 the size of the original (largest) image. (The image is from NASA and the mipmap was created by Wikipedia user Mulad.)

Figure 5.22). To be more precise, aliasing analysis should take this into account as well.

By deciding to fully include or exclude the triangle based on the coordinates of p alone, the staircasing effect is unavoidable. A better way is to render the pixel according to the fraction of the pixel region that is covered by the triangle. This way its values could be blended from multiple triangles that are visible within the pixel region. Unfortunately, this requires *supersampling*, which means casting rays at a much higher density than the pixel density so that the triangle coverage fraction can be estimated. This dramatically increases cost. Commonly, a compromise is reached in a method called *multisample anti-aliasing* (or *MSAA*), in which only some values are calculated at the higher density. Typically, depth values are calculated for each sample, but shading is not.

A *spatial aliasing* problem results from texture mapping. The viewing transformation may dramatically reduce the size and aspect ratio of the original texture as it is mapped from the virtual world onto the screen. This may leave insufficient resolution to properly represent a repeating pattern in the texture; see Figure 7.12. This problem is often addressed in practice by pre-calculating and storing a *mipmap* for each texture; see Figure 7.11. The texture is calculated at various resolutions by performing high-density sampling and storing the rasterized result in images. Based on the size and viewpoint of the triangle on the screen, the appropriate scaled texture image is selected and mapped onto the triangle to reduce the aliasing artifacts.



(a) (b)

Figure 7.12: (a) Due to the perspective transformation, the tiled texture suffers from *spatial aliasing* as the distance increases. (b) The problem can be fixed by performing supersampling.

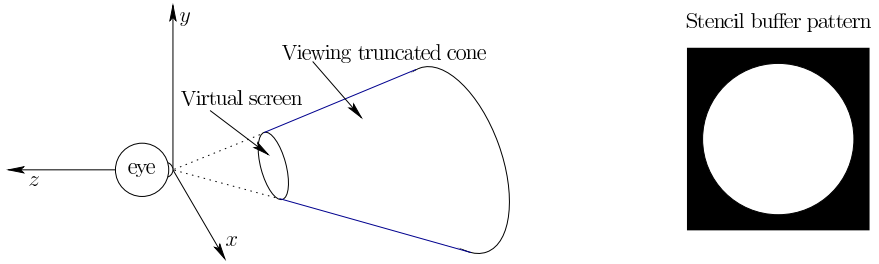


Figure 7.13: Due to optical system in front of the screen, the viewing frustum is replaced by a truncated cone in the case of a circularly symmetric view. Other cross-sectional shapes may be possible to account for the assyemetry of each eye view (for example, the nose is obstructing part of the view).

Culling In practice, many triangles can be quickly eliminated before attempting to render them. This results in a preprocessing phase of the rendering approach called *culling*, which dramatically improves performance and enables faster frame rates. The efficiency of this operation depends heavily on the data structure used to represent the triangles. Thousands of triangles could be eliminated with a single comparison of coordinates if they are all arranged in a hierarchical structure. The most basic form of culling is called *view volume culling*, which eliminates all triangles that are wholly outside of the viewing frustum (recall Figure 3.18). For a VR headset, the frustum may have a curved cross section due to the limits of the optical system (see Figure 7.13). In this case, the frustum must be replaced with a region that has the appropriate shape. In the case of a truncated cone, a simple geometic test can quickly eliminate all objects ouside of the view. For example, if

$$\frac{\sqrt{x^2 + y^2}}{-z} > \tan\theta, \tag{7.14}$$

in which 2θ is the angular field of view, then the point (x, y, z) is outside of the cone. Alternatively, the *stencil buffer* can be used in a GPU to mark all pixels that would be outside of the lens view. These are quickly eliminated from consideration by a simple test as each frame is rendered.

Another form is called *backface culling*, which removes triangles that have outward surface normals that point away from the focal point. These should not be rendered "from behind" if the model is consistently formed. Additionally, *occlusion culling* may be used to eliminate parts of the model that might be hidden from view by a closer object. This can get complicated because it once again considers the depth ordering problem. For complete details, see [3].

VR-specific rasterization problems The staircasing problem due to aliasing is expected to be worse for VR because current resolutions are well below the required retina display limit calculated in Section 5.4. The problem is made significantly worse by the continuously changing viewpoint due to head motion. Even as the user attempts to stare at an edge, the "stairs" appear to be more like an "escalator" because the exact choice of pixels to include in a triangle depends on subtle variations in the viewpoint. As part of our normal perceptual processes, our eyes are drawn toward this distracting motion. With stereo viewpoints, the situation is worse: The "escalator" from the right and left images will usually not match. As the brain attempts to fuse the two images into one coherent view, the aliasing artifacts provide a strong, moving mismatch. Reducing contrast at edges and using anti-aliasing techniques help alleviate the problem, but aliasing is likely to remain a significant problem until displays reach the required retina display density for VR.

A more serious difficulty is caused by the enhanced depth perception afforded by a VR system. Both head motions and stereo views enable us to perceive small differences in depth across surfaces. This should be a positive outcome; however, many tricks developed in computer graphics over the decades rely on the fact that

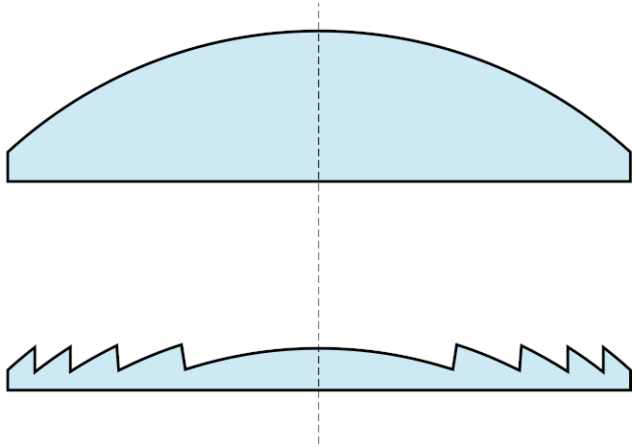


Figure 7.14: A Fresnel lens (pronounced like "frenelle") simulates a simple lens by making a corrugated surface. The convex surface on the top lens is implemented in the Fresnel lens shown on the bottom.

people cannot perceive these differences when a virtual world is rendered onto a fixed screen that is viewed from a significant distance. The result for VR is that texture maps may look fake. For example, texture mapping a picture of a carpet onto the floor might inadvertently cause the floor to look as it were simply painted. In the real world we would certainly be able to distinguish painted carpet from real carpet. The same problem occurs with normal mapping. A surface that might look rough in a single static image due to bump mapping could look completely flat in VR as both eyes converge onto the surface. Thus, as the quality of VR systems improves, we should expect the rendering quality requirements to increase, causing many old tricks to be modified or abandoned.

7.3 Correcting Optical Distortions

Recall from Section 4.3 that barrel and pincushion distortions are common for an optical system with a high field of view (Figure 4.20). When looking through the lens of a VR headset such as the Oculus Rift DK2, a pincushion distortion usually results. If the images are drawn on the screen without any correction, then the virtual world appears to be incorrectly warped. If the user yaws his head back and forth, then fixed lines in the world, such as walls, appear to dynamically change their curvature because the distortion in the periphery is much stronger than in the center. If it is not corrected, then the perception of stationarity will fail because static objects should not appear to be warping dynamically. Furthermore, contributions may be made to VR sickness because incorrect accelerations are being visually perceived near the periphery.

How can this problem be solved? One way to avoid this effect is to replace the classical optical system with *digital light processing* (*DLP*) technology that directly projects light into the eye using MEMS technology. Another way to greatly reduce this problem is to use a *Fresnel lens* (see Figure 7.14), which more accurately controls the bending of light rays by using a corrugated or sawtooth surface. This is used, for example, in the HTC Vive VR headset. One unfortunate side effect of Fresnel lenses is that glaring can be frequently observed as light scatters across the ridges along the surface.

Whether small or large, the distortion can also be corrected in software. One assumption is that the distortion is circularly symmetric. This means that the amount of distortion depends only on the distance from the lens center, and not the particular direction from the center. Even if the lens distortion is perfectly circularly symmetric, it must also be placed so that it is centered over the eye. Some headsets offer IPD adjustment, which allows the distance between the lenses to be adjusted so that they are matched to the user's eyes. If the eye is not centered on the lens, then asymmetric distortion arises. The situation is not perfect because as the eye rotates, the pupil moves along a spherical arc. As the position of the pupil over the lens changes laterally, the distortion varies and becomes asymmetric. This motivates making the lens as large as possible so that this problem is reduced. Another factor is that the distortion will change as the distance between the lens and the screen is altered. This adjustment may be useful to accommodate users with nearsightedness or farsightedness, as done in the Samsung Gear VR headset. The adjustment is also common in binoculars and binoculars, which explains why many people do not need their glasses to use them. To handle distortion correctly, the headset should sense this adjustment and take it into account.

To fix radially symmetric distortion, suppose that the transformation chain $T_{can}T_{eye}T_{rb}$ has been applied to the geometry, resulting in the canonical view volume, as covered in Section 3.5. All points that were inside of the viewing frustum now have x and y coordinates ranging from -1 to 1. Consider referring to these points using polar coordinates (r, θ):

$$\begin{aligned} r &= \sqrt{x^2 + y^2} \\ \theta &= \text{atan2}(y, x), \end{aligned} \tag{7.15}$$

in which atan2 represents the inverse tangent of y/x. This function is commonly used in programming languages to return an angle θ over the entire range from 0 to 2π. (The arctangent alone cannot do this because the quadrant that (x, y) came from is needed.)

We now express the lens distortion in terms of transforming the radius r, without affecting the direction θ (because of symmetry). Let f denote a function that applies to positive real numbers and distorts the radius. Let r_u denote the undistorted radius, and let r_d denote the distorted radius. Both pincushion and barrel distortion are commonly approximated using polynomials with odd powers, resulting in f being defined as

$$r_d = f(r_u) = r_u + c_1 r_u^3 + c_2 r_u^5, \tag{7.16}$$

in which c_1 and c_2 are suitably chosen constants. If $c_1 < 0$, then barrel distortion occurs. If $c_1 > 0$, then pincushion distortion results. Higher-order polynomials could also be used, such as adding a term $c_3 r_u^7$ on the right above; however, in practice this is often considered unnecessary.

Correcting the distortion involves two phases:

1. Determine the radial distortion function f for a particular headset, which involves a particular lens placed at a fixed distance from the screen. This is a regression or curve-fitting problem that involves an experimental setup that measures the distortion of many points and selects the coefficients c_1, c_2, and so on, that provide the best fit.

2. Determine the inverse of f so that it be applied to the rendered image before the lens causes its distortion. The composition of the inverse with f should cancel out the distortion function.

Unfortunately, polynomial functions generally do not have inverses that can be determined or even expressed in a closed form. Therefore, approximations are used. One commonly used approximation is [59]:

$$f^{-1}(r_d) \approx \frac{c_1 r_d^2 + c_2 r_d^4 + c_1^2 r_d^4 + c_2^2 r_d^8 + 2c_1 c_2 r_d^6}{1 + 4c_1 r_d^2 + 6c_2 r_d^4}. \tag{7.17}$$

Alternatively, the inverse can be calculated very accurately off-line and then stored in an array for fast access. It needs to be done only once per headset design. Linear interpolation can be used for improved accuracy. The inverse values can be accurately calculated using Newton's method, with initial guesses provided by simply plotting $f(r_u)$ against r_u and swapping the axes.

The transformation f^{-1} could be worked directly into the perspective transformation, thereby replacing T_p and T_{can} with a nonlinear operation. By leveraging the existing graphics rendering pipeline, it is instead handled as a post-processing step. The process of transforming the image is sometimes called *distortion shading* because it can be implemented as a shading operation in the GPU; it has nothing to do with "shading" as defined in Section 7.1. The rasterized image that was calculated using methods in Section 7.2 can be converted into a transformed image using (7.17), or another representation of f^{-1}, on a pixel-by-pixel basis. If compensating for a pincushion distortion, the resulting image will appear to have a barrel distortion; see Figure 7.15. To improve VR performance, *multiresolution shading* is used in the latest Nvidia GPUs []. One problem is that the resolution is effectively dropped near the periphery because of the transformed image (Figure 7.15). This results in wasted shading calculations in the original image. Instead, the image can be rendered before the transformation by taking into account the final resulting resolutions after the transformation. A lower-resolution image is rendered in a region that will become compressed by the transformation.

The methods described in this section may also be used for other optical distortions that are radially symmetric. For example, chromatic aberration can be

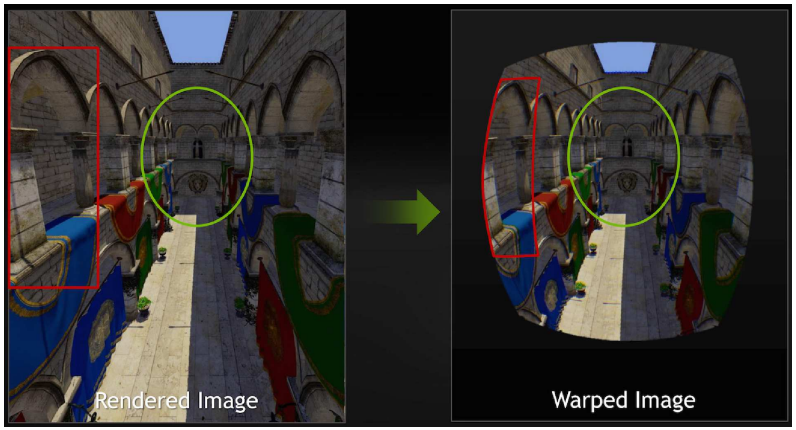


Figure 7.15: The rendered image appears to have a barrel distortion. Note that the resolution is effectively dropped near the periphery. (Figure by Nvidia.)

partially corrected by transforming the red, green, and blue subpixels differently. Each color is displaced radially by a different amount to compensate for the radial distortion that occurs based on its wavelength. If chromatic aberration correction is being used, then you can remove the lenses from a VR headset and see that the colors are not perfectly aligned in the images being rendered to the display. The system must create a distortion of pixel placements on the basis of color so that they will be moved closer to the correct places after they pass through the lens.

7.4 Improving Latency and Frame Rates

The *motion-to-photons* latency in a VR headset is the amount of time it takes to update the display in response to a change in head orientation and position. For example, suppose the user is fixating on a stationary feature in the virtual world. As the head yaws to the right, the image of the feature on the display must immediately shift to the left. Otherwise, the feature will appear to move if the eyes remain fixated on it. This breaks the perception of stationary.

The perfect system As a thought experiment, imagine the perfect VR system. As the head moves, the viewpoint must accordingly change for visual rendering. A magic oracle perfectly indicates the head position and orientation at any time. The VWG continuously maintains the positions and orientations of all objects in the virtual world. The visual rendering system maintains all perspective and viewport transformations, and the entire rasterization process continuously sets the RGB values on the display according to the shading models. Progressing with this fantasy, the display itself continuously updates, taking no time to switch the

pixels, and it has a VR retina display resolution, as described in Section 5.4. In this case, visual stimulation provided by the virtual world should match what would occur in a similar physical world in terms of the geometry. There would be no errors in time and space (although the physics might not match anyway due to assumptions about lighting, shading, material properties, color spaces, and so on).

Historical problems In practice, the perfect system is not realizable. All of these operations require time to propagate information and perform computations. In early VR systems, the total motion-to-photons latency was often over 100ms. In the 1990s, 60ms was considered an acceptable amount. Latency has been stated as one of the greatest causes of VR sickness, and therefore one of the main obstructions to widespread adoption over the past decades. People generally adapt to a fixed latency, which somewhat mitigates the problem, but then causes problems when they have to readjust to the world world []. Variable latencies are even worse due to the inability to adapt []. Fortunately, latency is no longer the main problem in most VR systems because of the latest-generation sensing, GPU, and display technology. The latency may be around 15 to 25ms, which is even compensated for by predictive methods in the tracking system. The result is that the *effective* latency is very close to zero. Thus, other factors are now contributing more strongly to VR sickness and fatigue, such as vection and optical aberrations.

A simple example Let d be the density of the display in pixels per degree. Let ω be the angular velocity of the head in degrees per second. Let ℓ be the latency in seconds. Due to latency ℓ and angular velocity ω, the image is shifted by $d\omega\ell$ pixels. For example, if $d = 40$ pixels per degree, $\omega = 50$ degrees per second, and $\ell = 0.02$ seconds, then the image is incorrectly displaced by $d\omega\ell = 4$ pixels. An extremely fast head turn might be at 300 degrees per second, which would result in a 24-pixel error.

Overview of latency reduction methods The following strategies are used together to both reduce the latency and to minimize the side effects of any remaining latency:

1. Lower the complexity of the virtual world.
2. Improve rendering pipeline performance.
3. Remove delays along the path from the rendered image to switching pixels.
4. Use prediction to estimate future viewpoints and world states.
5. Shift or distort the rendered image to compensate for last-moment viewpoint errors.

Each of these will be described in succession.

Figure 7.16: A variety of mesh simplification algorithms can be used to reduce the model complexity while retaining the most important structures. Shown here is a simplification of a hand model made by the open-source library CGAL. (Figure by Fernando Cacciola.)

Simplifying the virtual world Recall from Section 3.1 that the virtual world is composed of geometric primitives, which are usually 3D triangles arranged in a mesh. The chain of transformations and rasterization process must be applied for each triangle, resulting in a computational cost that is directly proportional to the number of triangles. Thus, a model that contains tens of millions of triangles will take orders of magnitude longer to render than one made of a few thousand. In many cases, we obtain models that are much larger than necessary. They can often be made much smaller (fewer triangles) with no perceptible difference, much in the same way that image, video, and audio compression works. Why are they too big in the first place? If the model was captured from a 3D scan of the real world, then it is likely to contain highly dense data. Capture systems such as the FARO Focus3D X Series capture large worlds while facing outside. Others, such as the Matter and Form MFSV1, capture a small object by rotating it on a turntable. As with cameras, systems that construct 3D models automatically are focused on producing highly accurate and dense representations, which maximize the model size. Even in the case of purely synthetic worlds, a modeling tool such as Maya or Blender will automatically construct a highly accurate mesh of triangles over a curved surface. Without taking specific care of later rendering burdens, the model could quickly become unwieldy. Fortunately, it is possible to reduce the model size by using *mesh simplification* algorithms; see Figure 7.16. In this case, one must be careful to make sure that the simplified model will have sufficient quality from all

viewpoints that might arise in the targeted VR system. In some systems, such as Unity 3D, reducing the number of different material properties across the model will also improve performance.

In addition to reducing the rendering time, a simplified model will also lower computational demands on the Virtual World Generator (VWG). For a *static world*, the VWG does not need to perform any updates after initialization. The user simply views the fixed world. For *dynamic worlds*, the VWG maintains a simulation of the virtual world that moves all geometric bodies while satisfying physical laws that mimic the real world. It must handle the motions of any avatars, falling objects, moving vehicles, swaying trees, and so on. Collision detection methods are needed to make bodies react appropriately when in contact. Differential equations that model motion laws may be integrated to place bodies correctly over time. These issues will be explained in Chapter 8, but for now it is sufficient to understand that the VWG must maintain a coherent snapshot of the virtual world each time a rendering request is made. Thus, the VWG has a frame rate in the same way as a display or visual rendering system. Each VWG frame corresponds to the placement of all geometric bodies for a common time instant. How many times per second can the VWG be updated? Can a high, constant rate of VWG frames be maintained? What happens when a rendering request is made while the VWG is in the middle of updating the world? If the rendering module does not wait for the VWG update to be completed, then some objects could be incorrectly placed because some are updated while others are not. Thus, the system should ideally wait until a complete VWG frame is finished before rendering. This suggests that the VWG update should be at least as fast as the rendering process, and the two should be carefully synchronized so that a complete, fresh VWG frame is always ready for rendering.

Improving rendering performance Any techniques that improve rendering performance in the broad field of computer graphics apply here; however, one must avoid cases in which side effects that were imperceptible on a computer display become noticeable in VR. It was already mentioned in Section 7.2 that texture and normal mapping methods are less effective in VR for this reason; many more discrepancies are likely to be revealed in coming years. Regarding improvements that are unique to VR, it was mentioned in Sections 7.2 and 7.3 that the stencil buffer and multiresolution shading can be used to improve rendering performance by exploiting the shape and distortion due to the lens in a VR headset. A further improvement is to perform rasterization for the left and right eyes in parallel in the GPU, using one processor for each. The two processes are completely independent. This represents an important first step, among many that are likely to come, in design of GPUs that are targeted specifically for VR.

From rendered image to switching pixels The problem of waiting for coherent VWG frames also arises in the process of rendering frames to the display: When it is time to scan out the rendered image to the display, it might not be

Figure 7.17: If a new frame is written to the video memory while a display scanout occurs, then *tearing* occurs, in which parts of two or more frames become visible at the same time.

finished yet. Recall from Section 5.4 that most displays have a rolling scanout that draws the rows of the rasterized image, which sits in the *video memory*, onto the screen one-by-one. This was motivated by the motion of the electron beam that lit phosphors on analog TV screens. The motion is left to right, and top to bottom, much in the same way we would write out a page of English text with a pencil and paper. Due to inductive inertia in the magnetic coils that bent the beam, there was a period of several milliseconds called VBLANK (*vertical blanking interval*) in which the beam moves from the lower right back to the upper left of the screen to start the next frame. During this time, the beam was turned off to avoid drawing a diagonal streak across the frame, hence, the name "blanking". Short blanking intervals also occurred as each horizontal line to bring the beam back from the right to the left.

In the era of digital displays, the scanning process in unnecessary, but it nevertheless persists and causes some trouble. Suppose that a display runs at 100 FPS. In this case, a request to draw a new rendered image is made every 10ms. Suppose that VBLANK occurs for 2ms and the remaining 8ms is spent drawing lines on the display. If the new rasterized image is written to the video memory during the 2ms of VBLANK, then it will be correctly drawn in the remaining 8ms. It is also possible to earn extra time through beam racing, which was mentioned in Section 5.4. However, if a new image is being written and passes where the beam is scanning it out, then a problem called *tearing* occurs because it appears as if is screen is torn into pieces; see Figure 7.17. If the VWG and rendering system produce frames at 300 FPS, then parts or 3 or 4 images could appear on the display because the image changes several times while the lines are being scanned out. One solution to this problem is a method called VSYNC (pronounced "vee

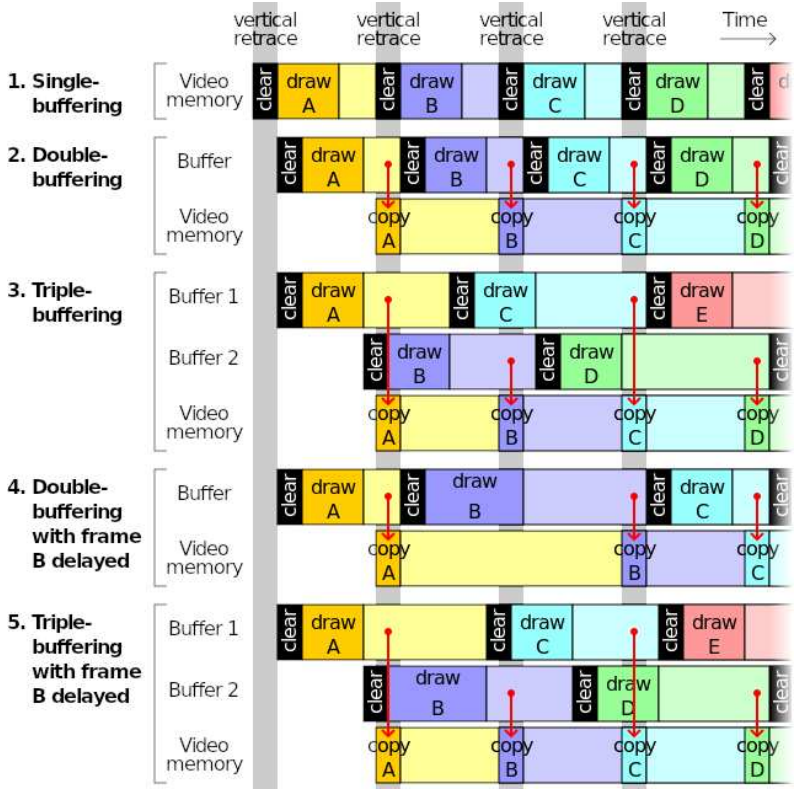


Figure 7.18: Buffering is commonly used in visual rendering pipelines to avoid tearing and lost frames; however, it introduces more latency, which is detrimental to VR. (Figure by Wikipedia user Cmglee.)

sink"), which is a flag that prevents the video memory from being written outside of the VBLANK interval.

Another strategy to avoid tearing is *buffering*, which is shown in Figure 7.18. The approach is simple for programmers because it allows the frames to be written in memory that is not being scanned for output to the display. The unfortunate side effect is that it increases the latency. For *double buffering*, a new frame is first drawn into the buffer and then transferred to the video memory during VBLANK. It is often difficult to control the rate at which frames are produced because the operating system may temporarily interrupt the process or alter its priority. In this case, *triple buffering* is an improvement that allows more time to render each frame. For avoiding tearing and providing smooth video game performance, buffering has been useful; however, it is detrimental to VR because of the increased latency.

Ideally, the displays should have a global scanout, in which all pixels are

switched at the same time. This allows a much longer interval to write to the video memory and avoids tearing. It would also reduce the latency in the time it takes to scan the first pixel to the last pixel. In our example, this was an 8ms interval. Finally, displays should reduce the pixel switching time as much as possible. In a smartphone LCD screen, it could take up to 20ms to switch pixels; however, OLED pixels can be switched in under 0.1ms.

The power of prediction For the rest of this section, we consider how to live with whatever latency remains. As another thought experiment, imagine that a fortune teller is able to accurately predict the future. With such a device, it should be possible to eliminate all latency problems. We would want to ask the fortune teller the following:

1. At what future time will the pixels be switching?

2. What will be the positions and orientations of all virtual world models at that time?

3. Where will the user be looking at that time?

Let t_s be answer to the first question. We need to ask the VWG to produce a frame for time t_s and then perform visual rendering for the user's viewpoint at time t_s. When the pixels are switched at time t_s, then the stimulus will be presented to the user at the exact time and place it is expected. In this case, there is *zero effective latency*.

Now consider what happens in practice. First note that using information from all three questions above implies significant time synchronization across the VR system: All operations must have access to a common clock. For the first question above, determining t_s should be feasible if the computer is powerful enough and the VR system has enough control from the operating system to ensure that VWG frames will be consistently produced and rendered at the frame rate. The second question is easy for the case of a static virtual world. In the case of a dynamic world, it might be straightforward for all bodies that move according to predictable physical laws. However, it is difficult to predict what humans will do in the virtual world. This complicates the answers to both the second and third questions. Fortunately, the latency is so small that *momentum* and *inertia* play a significant role; see Chapter 8. Bodies in the matched zone are following physical laws of motion from the real world. These motions are sensed and tracked according to methods covered in Chapter 9. Although it might be hard to predict where you will be looking in 5 seconds, it is possible to predict with very high accuracy where your head will be positioned and oriented in 20ms. You have no free will on the scale of 20ms! Instead, momentum dominates and the head motion can be accurately predicted. Some body parts, especially fingers, have much less inertia, and therefore become more difficult to predict; however, these are not as important as predicting head motion. The viewpoint depends only on the head motion, and

Perturbation	Image effect
$\Delta\alpha$ (yaw)	Horizontal shift
$\Delta\beta$ (pitch)	Vertical shift
$\Delta\gamma$ (roll)	Rotation about image center
x	Horizontal shift
y	Vertical shift
z	Contraction or expansion

Figure 7.19: Six cases of post-rendering image warp based on the degrees of freedom for a change in viewpoint. The first three correspond to an orientation change. The remaining three correspond to a position change. These operations can be visualized by turning on a digital camera and observing how the image changes under each of these perturbations.

latency reduction is most critical in this case to avoid perceptual problems that lead to fatigue and VR sickness.

Post-rendering image warp Due to both latency and imperfections in the prediction process, a last-moment adjustment might be needed before the frame is scanned out to the display. This is called *post-rendering image warp* [90] (it has also been rediscovered and called *time warp* in the recent VR industry []). At this stage, there is no time to perform complicated shading operations; therefore, a simple transformation is made to the image.

Suppose that an image has been rasterized for a particular viewpoint, expressed by position (x, y, z) and orientation given by yaw, pitch, and roll (α, β, γ). What would be different about the image if it were rasterized for a nearby viewpoint? Based on the degrees of freedom for viewpoints, there are six types of adjustment; see Figure 7.19. Each one of these has a direction that is not specified in the figure. For example, if $\Delta\alpha$ is positive, which corresponds to a small, counterclockwise yaw of the viewpoint, then the image is shifted horizontally *to the right*.

Figure 7.20 shows some examples of the image warp. Most cases require the rendered image to be larger than the targeted display; otherwise, there will be no data to shift into the warped image; see Figure 7.20(d). In this case, it is perhaps best to repeat pixels from the edge, rather than turning them black [90].

Flaws in the warped image Image warping due to orientation changes produces a correct image in the sense that it should be exactly what would have been rendered from scratch for that orientation (without taking aliasing issues into account). However, positional changes are incorrect! Perturbations in x and y do not account for motion parallax (recall from Section 6.1), which would require knowing the depths of the objects. Changes in z produce similarly incorrect images because nearby objects should expand or contract by a larger amount than further ones. To make matters worse, changes in viewpoint position might lead to a *visibility*

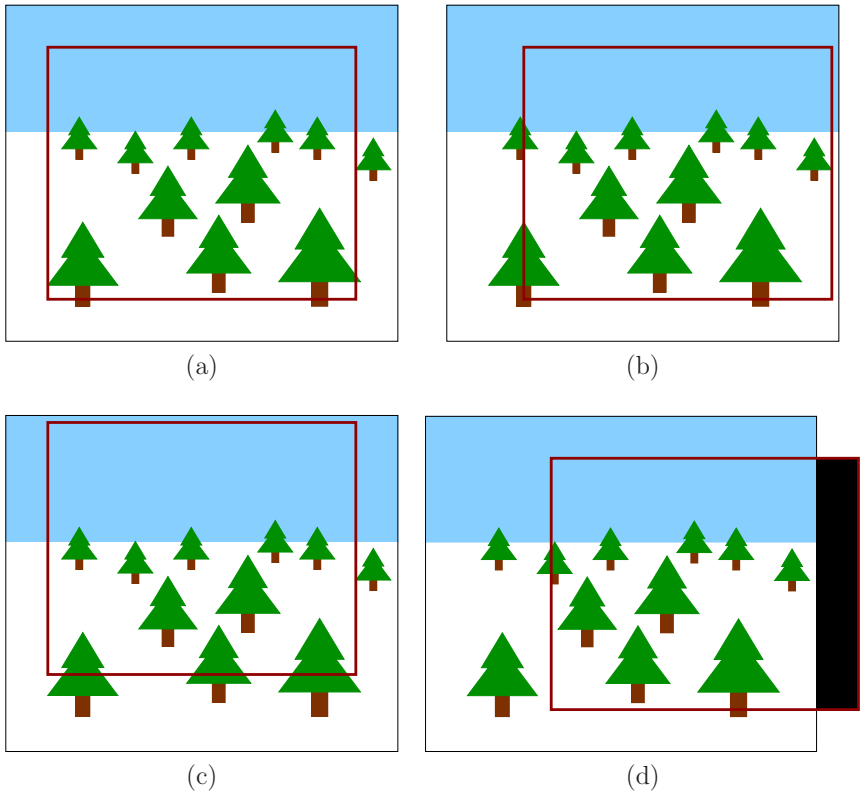


Figure 7.20: Several examples of post-rendering image warp: (a) Before warping, a larger image is rasterized. The red box shows the part that is intended to be sent to the display based on the viewpoint that was used at the time of rasterization; (b) A negative yaw (turning the head to the right) causes the red box to shift to the right. The image appears to shift to the left; (c) A positive pitch (looking upward) causes the box to shift upward; (d) In this case the yaw is too large and there is no rasterized data to use for part of the image (this region is shown as a black rectangle).

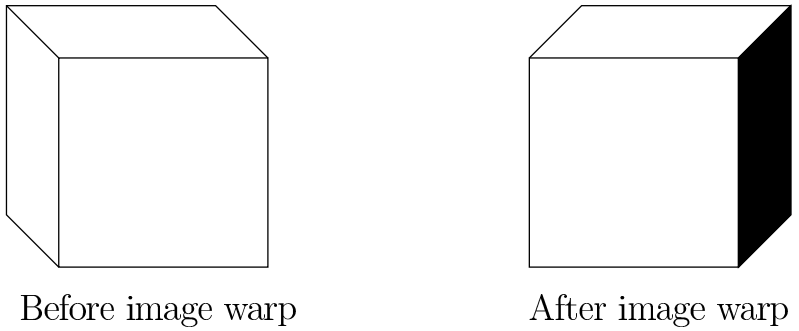


Figure 7.21: If the viewing position changes, then a *visibility event* might be encountered. This means that part of the object might suddenly become visible from the new perspective. In this sample, a horizontal shift in the viewpoint reveals a side of the cube that was originally hidden. Furthermore, the top of the cube changes its shape.

event, in which part of an object may become visible only in the new viewpoint; see Figure 7.21. Data structures such as an *aspect graph* [] and *visibility complex* [] are designed to maintain such events, but are usually not included in the rendering process. As latencies become shorter and prediction becomes better, the amount of perturbation is reduced. Careful perceptual studies are needed to evaluate conditions under which image warping errors are perceptible or cause discomfort. An alternative to image warping is to use parallel processing to sample several future viewpoints and render images for all of them. The most correct image can then be selected, to greatly reduce the image warping artifacts.

Increasing the frame rate Post-rendering image warp can also be used to artificially increase the frame rate. For example, suppose that only one rasterized image is produced every 100 milliseconds by a weak computer or GPU. This would result in poor performance at 10 FPS. Suppose we would like to increase this to 100 FPS. In this case, a single rasterized image can be warped to produce frames every 10ms until the next rasterized image is computed. In this case, 10 warped frames are used for every rasterized image that is computed.

7.5 Immersive Photos and Videos

Up until now, this chapter has focused on rendering a virtual world that was constructed synthetically from geometric models. The methods developed over decades of computer graphics research have targeted this case. The trend has recently changed, though, toward capturing real-world images and video, which are then easily embedded into VR experiences. This change is mostly due to smartphone industry, which has led to hundreds of millions of people carrying

(a) (b)

Figure 7.22: (a) As of 2015, Netflix offers online movie streaming onto a large virtual TV screen while the user appears to sit in a living room. (b) The movies are texture-mapped onto the TV screen, frame by frame. Furthermore, the gaze pointer allows the user to look in a particular direction to select content.

high resolution cameras with them everywhere. Furthermore, 3D camera technology continues to advance, which provides distance information in addition to color and light intensity. All of this technology is quickly converging to the case of *panoramas*, which contained captured image data from all possible viewing directions. A current challenge is to also capture data within a region of all possible viewing *positions and orientations*.

Texture mapping onto a virtual screen Putting a photo or video into a virtual world is an application of texture mapping. Figure 7.22 shows a commercial use in which Netflix offers online movie streaming through the Samsung Gear VR headset. The virtual screen is a single rectangle, which may be viewed as a simple mesh consisting of two triangles. A photo can be mapped across any triangular mesh in the virtual world. In the case of a movie, each frame is treated as a photo that is texture-mapped to the mesh. The movie frame rate is usually much lower than that of the VR headset (recall Figure 6.17). As an example, suppose the movie was recorded at 24 FPS and the headset runs at 96 FPS. In this case, each movie frame is rendered for four frames on the headset display. Most often, the frame rates are not perfectly divisible, which causes the number of repeated frames to alternate in a pattern. An old example of this is called *3:2 pull down*, in which 24 FPS movies were converted to NTSC TV format at 30 FPS. Interestingly, a 3D movie (stereoscopic) experience can even be simulated. For the left eye on the headset display, the left-eye movie frame is rendered to the virtual screen. Likewise, the right-eye movie frame is rendered to the right-eyed portion of the headset display. The result is that the user perceives it as a 3D movie, without wearing the special glasses! Of course, she would be wearing a VR headset.

Capturing a wider field of view Mapping onto a rectangle makes it easy to bring pictures or movies that were captured with ordinary cameras into VR; however, the VR medium itself allows great opportunities to expand the experience. Unlike life in the real world, the size of the virtual screen can be expanded without any significant cost. To fill the field of view of the user, it makes sense to curve the virtual screen and put the user at the center. Such curving already exists in the real world; examples are the 1950s Cinerama experience, which was shown in Figure 1.25(d), and modern curved displays. In the limiting case, we obtain a panoramic photo, sometimes called a *photosphere*. Displaying many photospheres per second leads to a panoramic movie, which we may call a *moviesphere*.

Recalling the way cameras work from Section 4.5, it is impossible to capture a photosphere from a single camera in a single instant of time. Two obvious choices exist:

1. Take multiple images with one camera by pointing it in different directions each time, until the entire sphere of all viewing directions is covered.

2. Use multiple cameras, pointing in various viewing directions, so that all directions are covered by taking synchronized pictures.

The first case leads to a well-studied problem in computer vision and computational photography called *image stitching*. A hard version of the problem can be made by stitching together an arbitrary collection of images, from various cameras and times. This might be appropriate, for example, to build a photosphere of a popular tourist site from online photo collections. More commonly, a smartphone user may capture a photosphere by pointing the outward-facing camera in enough directions. In this case, a software app builds the photosphere dynamically while images are taken in rapid succession. For the hard version, a difficult optimization problem arises in which features need to be identified and matched across overlapping parts of multiple images while unknown, intrinsic camera parameters are taken into account. Differences in perspective, optical aberrations, lighting conditions, exposure time, and changes in the scene over different times must be taken into account. In the case of using a smartphone app, the same camera is being used and the relative time between images is short; therefore, the task is much easier. Furthermore, by taking rapid images in succession and using internal smartphone sensors, it is much easier to match the overlapping image parts. Most flaws in such hand-generated photospheres are due to the user inadvertently changing the position of the camera while pointing it in various directions.

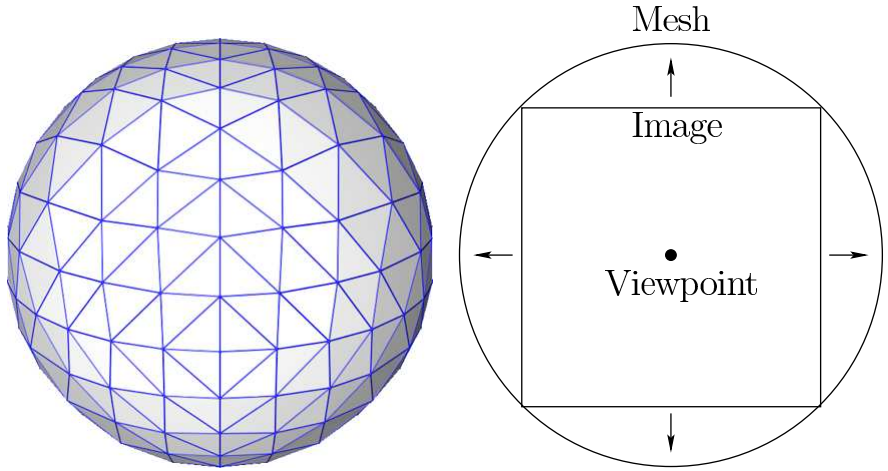
For the second case, a rig of identical cameras can be carefully designed so that all viewing directions are covered; see Figure 7.23. Once it is calibrated so that the relative positions and orientations of the cameras are precisely known, stitching the images together becomes straightforward. Corrections may nevertheless be applied to account for variations in lighting or calibration; otherwise, the seams in the stitching may become perceptible. A tradeoff exists in terms of the number of cameras. By using many cameras, very high resolution captures can be made with

(a) (b)

Figure 7.23: (a) The 360Heros Pro10 HD is a rig that mounts ten GoPro cameras in opposing directions to capture panoramic images. (b) The Ricoh Theta S captures panoramic photos and videos using only two cameras, each with a lens that provides a field of view larger than 180 degrees.

relatively little optical distortion because each camera contributes a narrow field-of-view image to the photoshpere. At the other extreme, as few as two cameras are sufficient, as in the case of the Ricoh Theta S. The cameras are pointed 180 degrees apart and a fish-eyed lens is able to capture a view that is larger than 180 degrees. This design dramatically reduces costs, but requires significant unwarping of the two captured images.

Mapping onto a sphere The well-known *map projection* problem from cartography would be confronted to map the photosphere onto a screen; however, this does not arise when rendering a photosphere in VR because it is mapped directly onto a sphere in the virtual world. The sphere of all possible viewing directions maps to the virtual-world sphere without distortions. To directly use texture mapping techniques, the virtual-world sphere can be approximated by uniform triangles, as shown in Figure 7.24(a). The photosphere itself should be stored in a way that does not degrade its resolution in some places. We cannot simply use latitude and longitude coordinates to index the pixels because the difference in resolution between the poles and the equator would be too large. We could use coordinates that are similar to the way quaternions cover the sphere by using indices (a, b, c) and requiring that $a^2 + b^2 + c^2 = 1$; however, the structure of neighboring pixels (up, down, left, and right) is not clear. A simple and efficient compromise is to represent the photosphere as six square images, each corresponding to the face of a cube. This is like a virtual version of a six-sided CAVE projection system. Each image can then be easily mapped onto the mesh with little loss in resolution,



(a) (b)

Figure 7.24: (a) The photophere is texture-mapped onto the interior of a sphere that is modeled as a triangular mesh. (b) A photosphere stored as a cube of six images can be quickly mapped to the sphere with relatively small loss of resolution; a cross section is shown here.

as shown in Figure 7.24(b).

Once the photosphere (or moviesphere) is rendered onto the virtual sphere, the rendering process is very similar to post-rendering image warp. The image presented to the user is shifted for the rotational cases that were described in Figure 7.19. In fact, the entire rasterization process could be performed only once, for the entire sphere, while the image rendered to the display is adjusted based on the viewing direction. Further optimizations could be made by even bypassing the mesh and directly forming the rasterized image from the captured images.

Perceptual issues Does the virtual world appear to be "3D" when viewing a photosphere or moviesphere? Recall from Section 6.1 that there are many more monocular depth cues than stereo cues. Due to the high field-of-view of modern VR headsets and monocular depth cues, a surprisingly immersive experience is obtained. Thus, it may feel more "3D" than people expect, even if the same part of the panoramic image is presented to both eyes. Many interesting questions remain for future research regarding the perception of panoramas. If different viewpoints are presented to the left and right eyes, then what should the radius of the virtual sphere be for comfortable and realistic viewing? Continuing further, suppose positional head tracking is used. This might improve viewing comfort, but the virtual world will appear more flat because parallax is not functioning. For example, closer objects will not move more quickly as the head moves from

Figure 7.25: The Pantopticam prototype from Figure Digital uses dozens of cameras to improve the ability to approximate more viewpoints so that stereo viewing and parallax from position changes can be simulated.

side to side. Can simple transformations be performed to the images so that depth perception is enhanced? Can limited depth data, which could even be extracted automatically from the images, greatly improve parallax and depth perception? Another issue is designing interfaces inside of photospheres. Suppose we would like to have a shared experience with other users inside of the sphere. In this case, how do we perceive virtual objects inserted into the sphere, such as menus or avatars? How well would a virtual laser pointer work to select objects?

Panoramic light fields Panoramic images are simple to construct, but are clearly flawed because they do not account how the surround world would appear from any viewpoint that could be obtained by user movement. To accurately determine this, the ideal situation would be to capture the entire *light field* of energy inside of whatever viewing volume that user is allowed to move. A light field provides both the spectral power and direction of light propagation at every point in space. If the user is able to walk around in the physical world while wearing a VR headset, then this seems to be an impossible task. How can a rig of cameras capture the light energy in all possible locations at the same instant in an entire room? If the user is constrained to a small area, then the light field can be approximately captured by a rig of cameras arranged on a sphere; a prototype is shown in Figure 7.25. In this case, dozens of cameras may be necessary, and image warping techniques are used to approximate viewpoints between the cameras or from the interior the spherical rig. To further improve the experience, *light-field cameras* (also called *plenoptic cameras*) offer the ability to capture both the intensity of light rays and the direction that they are traveling through space. This offers many advantages, such as refocusing images to different depths, after the light field has already been captured.

Further Reading

Close connections exist between VR and computer graphics because both are required to push visual information onto a display. For more, consult basic textbooks [91]. For high performance rendering, see [3].

Eurographics Symposium on Rendering (2006), Tomas Akenine-Möller and Wolfgang Heidrich (Editors), Instant Ray Tracing: The Bounding Interval Hierarchy, Carsten Wächter and Alexander Keller

A survey of ray tracing data structures from a computational geometry perspective: [25]

The BRDF was first introduced in [107]. For comprehensive coverage, see [3, 12].

Texture mapping was introduced in [24].

Correcting Lens Distortions in Digital Photographs, Wolfgang Hugemann, 2010

Vass, Perlaki, Lens distortion (This one is a more complex model. The inverse is not closed form.)

R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. IEEE Journal of Robotics and Automation, 3 (4): 323344, 1987.

A Rational Function Lens Distortion Model for General Cameras, David Claus and Andrew W. Fitzgibbon

Geometric camera calibration using circular control points, Janne Heikkilä IEEE T. PAMI 22:10, 2002.

Precise Radial Un-distortion of Images, John Mallon, Paul F. Whelan, 2004.

Chromatic aberration correction: Correcting the Chromatic Aberration in Barrel Distortion of Endoscopic Images, Ng, Kwong.

Automatic panorama stitching: [22, 137, 144]

Linear and Angular Head Accelerations in Daily Life, William R. Bussone. William R. Bussone, MS Thesis, Virginia Polytechnic, 2005.

Chapter 8

Motion in Real and Virtual Worlds

Chapter Status	Taken from *Virtual Reality*, S. M. LaValle
	This online chapter is not the final version! Check **http://vr.cs.uiuc.edu/** for information on the latest draft version. This draft was compiled on June 29, 2016.

Up to this point, the discussion of movement has been confined to specialized topics. Section 5.3 covered eye movements and Section 6.2 covered the perception of motion. The transformations from Chapter 3 indicate how to place bodies and change viewpoints, but precise mathematical descriptions of motions have not yet been necessary. We now want to model motions more accurately because the physics of both real and virtual worlds impact VR experiences. The accelerations and velocities of moving bodies impact simulations in the VWG and tracking methods used to capture user motions in the physical world. Therefore, this chapter provides foundations that will become useful for reading Chapter 9 on tracking, and Chapter 10 on interfaces.

Section 8.1 introduces fundamental concepts from math and physics, including velocities, accelerations, and the movement of rigid bodies. Section 8.2 presents the physiology and perceptual issues from the human vestibular system, which senses velocities and accelerations. Section 8.3 then describes how motions are described and produced in a VWG. This includes numerical integration and collision detection. Section 8.4 focuses on vection, which is a source of VR sickness that arises due to sensory conflict between the visual and vestibular systems: The eyes may perceive motion while the vestibular system is not fooled. This can be considered as competition between the physics of the real and virtual worlds.

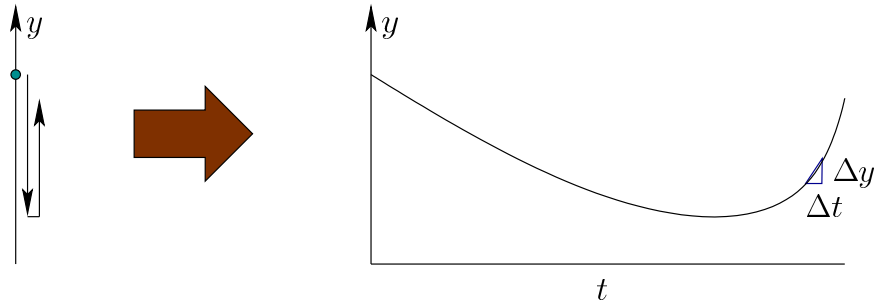


Figure 8.1: A point moving in a one-dimensional world.

8.1 Velocities and Accelerations

8.1.1 A one-dimensional world

We start with the simplest case, which is shown in Figure 8.1. Imagine a 1D world in which motion is only possible in the vertical direction. Let y be coordinate of a moving point. Its position at any time t is indicated by $y(t)$, meaning that y actually defines a function of time. It is now as if y were an animated point, with an infinite number of frames per second!

Velocity How fast is the point moving? Using calculus, its *velocity*, v, is defined as the derivative of y with respect to time:

$$v = \frac{dy(t)}{dt}. \tag{8.1}$$

Using numerical computations, v is approximately equal to $\Delta y/\Delta t$, in which Δt denotes a small change in time and $\Delta y = y(t + \Delta t) - y(t)$. In other words, Δy is the change in y from the start to the end of the time change. The velocity v can be used to estimate the change in y over Δt:

$$\Delta y \approx v \Delta t. \tag{8.2}$$

The approximation quality improves as Δt becomes smaller and v itself varies less during the time from t to $t + \Delta t$.

We can write $v(t)$ to indicate that velocity may change over time. The position can be calculated for any time t from the velocity using integration as[1]

$$y(t) = y(0) + \int_0^t v(s) ds, \tag{8.3}$$

[1]The parameter s is used instead of t to indicate that it is integrated away, much like the index in a summation.

which assumes that y was known at the starting time $t = 0$. If $v(t)$ is constant for all time, represented as v, then $y(t) = y(0) + vt$. The integral in (8.3) accounts for $v(t)$ being allowed to vary.

Acceleration The next step is to mathematically describe the change in velocity, which results in the *acceleration*, a; this is defined as:

$$a = \frac{dv(t)}{dt}. \tag{8.4}$$

The form is the same as (8.1), except that y has been replaced by v. Approximations can similarly be made. For example, $\Delta v \approx a\Delta t$.

The acceleration itself can vary over time, resulting in $a(t)$. The following integral relates acceleration and velocity (compare to (8.3)):

$$v(t) = v(0) + \int_0^t a(s)ds. \tag{8.5}$$

Since acceleration may vary, you may wonder whether the naming process continues. It could, with the next derivative called jerk, followed by snap, crackle, and pop. In most cases, however, these higher-order derivatives are not necessary. One of the main reasons is that motions from classical physics are sufficiently characterized through forces and accelerations. For example, Newton's Second Law states that $F = ma$, in which F is the force acting on a point, m is its mass, and a is the acceleration.

For a simple example that should be familiar, consider acceleration due to gravity, $g = 9.8\text{m/s}^2$. It is as if the ground were accelerating upward by g; hence, the point accelerates downward relative to the Earth. Using (8.5) to integrate the acceleration, the velocity over time is $v(t) = v(0) - gt$. Using (8.3) to integrate the velocity and supposing $v(0) = 0$, we obtain

$$y(t) = y(0) - \frac{1}{2}gt^2. \tag{8.6}$$

8.1.2 Motion in a 3D world

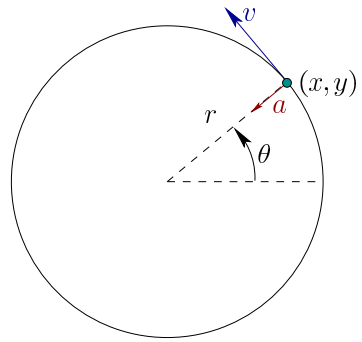
A moving point Now consider the motion of a point in a 3D world $\mathbb{R}^3$. Imagine that a geometric model, as defined in Section 3.1, is moving over time. This causes each point (x, y, z) on the model to move, resulting a function of time for each coordinate of each point:

$$(x(t), y(t), z(t)). \tag{8.7}$$

The velocity v and acceleration a from Section 8.1.1 must therefore expand to have three coordinates. The velocity v is replaced by (v_x, v_y, v_z) to indicate velocity with respect to the x, y, and z coordinates, respectively. The magnitude of v is called the *speed*:

$$\sqrt{v_x^2 + v_y^2 + v_z^2} \tag{8.8}$$

(a)



(b)

Figure 8.2: (a) Consider a merry-go-round that rotates at constant angular velocity ω. (Picture by Seattle Parks and Recreation.) (b) In a top-down view, the velocity vector, v, for a point on the merry-go-round is tangent to the circle that contains it; the circle is centered on the axis of rotation and the acceleration vector, a, points toward its center.

Continuing further, the acceleration also expands to include three components: (a_x, a_y, a_z).

Rigid-body motion Now suppose that a rigid body is moving through $\mathbb{R}^3$. In this case, all its points move together. How can we easily describe this motion? Recall from Section 3.2 that translations or rotations may be applied. First, consider a simple case. Suppose that rotations are prohibited, and the body is only allowed to translate through space. In this limited setting, knowing the position over time for one point on the body is sufficient for easily determining the positions of all points on the body over time. If one point has changed its position by some (x_t, y_t, z_t), then *all* points have changed by the same amount. More importantly, the velocity and acceleration of every point would be identical.

Once rotation is allowed, this simple behavior breaks. As a body rotates, the points no longer maintain the same velocities and accelerations. This becomes crucial to understanding VR sickness in Section 8.4 and how tracking methods estimate positions and orientations from sensors embedded in the world, which will be discussed in Chapter 9.

Angular velocity To understand the issues, consider the simple case of a spinning *merry-go-round*, as shown in Figure 8.2(a). Its orientation at every time can be described by $\theta(t)$; see Figure 8.2(b). Let ω denote its *angular velocity*:

$$\omega = \frac{d\theta(t)}{dt}. \tag{8.9}$$

By default, ω has units of radians per second. If $\omega = 2\pi$, then the rigid body returns to the same orientation after one second.

Assuming $\theta(0) = 0$, the orientation at time t is given by $\theta = \omega t$. To describe the motion of a point on the body, it will be convenient to use polar coordinates r and θ:

$$x = r\cos\theta \text{ and } y = r\sin\theta. \tag{8.10}$$

Substituting $\theta = \omega t$ yields

$$x = r\cos\omega t \text{ and } y = r\sin\omega t. \tag{8.11}$$

Taking the derivative with respect to time yields[2]

$$v_x = -r\omega\sin\omega t \text{ and } v_y = r\omega\cos\omega t. \tag{8.12}$$

The velocity is a 2D vector that when placed at the point is tangent to the circle that contains the point (x, y); see Figure 8.2(b).

This makes intuitive sense because the point is heading in that direction; however, the direction quickly changes because it must move along a circle. This change in velocity implies that a nonzero acceleration occurs. The acceleration of the point (x, y) is obtained by taking the derivative again:

$$a_x = -r\omega^2\cos\omega t \text{ and } a_y = -r\omega^2\sin\omega t. \tag{8.13}$$

The result is a 2D acceleration vector that is pointing toward the center (Figure 8.2(b)), which is rotation axis. This is called *centripetal acceleration*. If you were standing at that point, then you would feel a pull in the opposite direction, as if nature were attempting to fling you away from the center. This is precisely how artificial gravity can be achieved in a rotating space station.

3D angular velocity Now consider the rotation of a 3D rigid body. Recall from Section 3.3 that Euler's rotation theorem implies that every 3D rotation can be described as a rotation θ about an axis $v = (v_1, v_2, v_3)$ though the origin. As the orientation of the body changes over a short period of time Δt, imagine the axis that corresponds to the change in rotation. In the case of the merry-go-round, the axis would be $v = (0, 1, 0)$. More generally, v could be any unit vector.

The 3D angular velocity is therefore expressed as a 3D vector:

$$(\omega_x, \omega_y, \omega_z), \tag{8.14}$$

which can be imagined as taking the original ω from the 2D case and multiplying it by the vector v. This weights the components according to the coordinate axes. Thus, the components could be considered as $\omega_x = \omega v_1$, $\omega_y = \omega v_2$, and $\omega_z = \omega v_3$. The ω_x, ω_y, and ω_z components also correspond to the rotation rate in terms of pitch, roll, and yaw, respectively. We avoided these representations in Section 3.3 due to noncommutativity and kinematic singularities; however, it turns out that for velocities these problems do not exist [139]. Thus, we can avoid quaternions at this stage.

[2]If this is unfamiliar, then look up the derivatives of sines and cosines, and the chain rule, from standard calculus sources.

Angular acceleration If ω is allowed to vary over time, then we must consider *angular acceleration*. In the 2D case, this is defined as

$$\alpha = \frac{d\omega(t)}{dt}. \tag{8.15}$$

For the 3D case, there are three components, which results in

$$(\alpha_x, \alpha_y, \alpha_z). \tag{8.16}$$

These can be interpreted as accelerations of pitch, yaw, and roll angles, respectively.

8.2 The Vestibular System

As mentioned in Section 2.3, the *balance sense* (or *vestibular sense*) provides information to the brain about how the head is oriented or how it is moving in general. This is accomplished through *vestibular organs* that measure both linear and angular accelerations of the head. These organs, together with their associated neural pathways, will be referred to as the *vestibular system*. This system plays a crucial role for bodily functions that involve motion, from ordinary activity such as walking or running, to activities that require substantial talent and training, such as gymnastics or ballet dancing. Recall from Section 5.3 that it also enables eye motions that counteract head movements via the VOR.

The vestibular system is important to VR because it is usually neglected, which leads to a mismatch of perceptual cues (recall this problem from Section 6.4). In current VR systems, there is no engineered device that renders vestibular signals to a display that interfaces with the vestibular organs. Some possibilities may exist in the future with *galvanic vestibular stimulation*, which provides electrical stimulation to the organ [42, 41]; however, it may take many years before such techniques are sufficiently accurate, comfortable, and generally approved for safe use by the masses. Another possibility is to stimulate the vestibular system through low-frequency vibrations [], which at the very least provides some distraction.

Physiology Figure 8.4 shows the location of the vestibular organs inside of the human head. As in the cases of eyes and ears, there are two symmetric organs, corresponding to the right and left sides. Figure 8.3 shows the physiology of each vestibular organ. The *cochlea* handles hearing, which is covered in Section 11.2, and the remaining parts belong to the vestibular system. The *utricle* and *saccule* measure linear acceleration; together they form the *otolith system*. When the head is not tilted, the sensing surface of the utricle mostly lies in the horizontal plane (or xz plane in our common coordinate systems), whereas the corresponding surface of the saccule lies in a vertical plane that is aligned in the forward direction (this is called the *saggital plane*, or yz plane). As will be explained shortly, the utricle

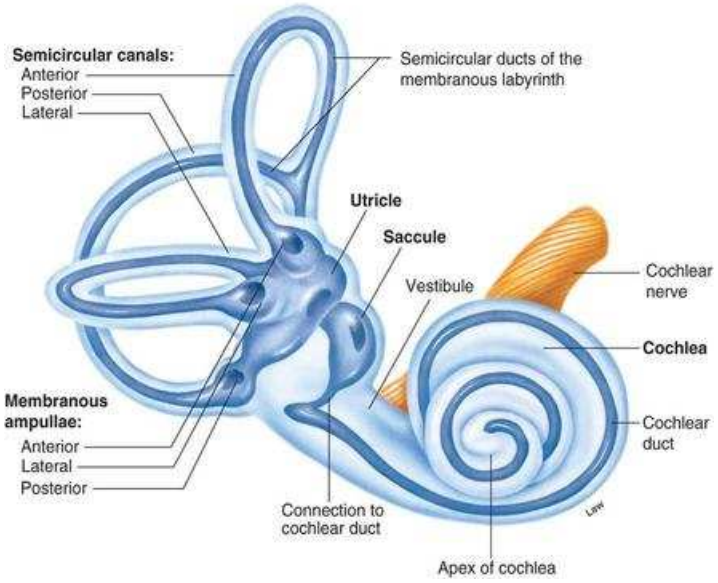


Figure 8.3: The vestibular organ.

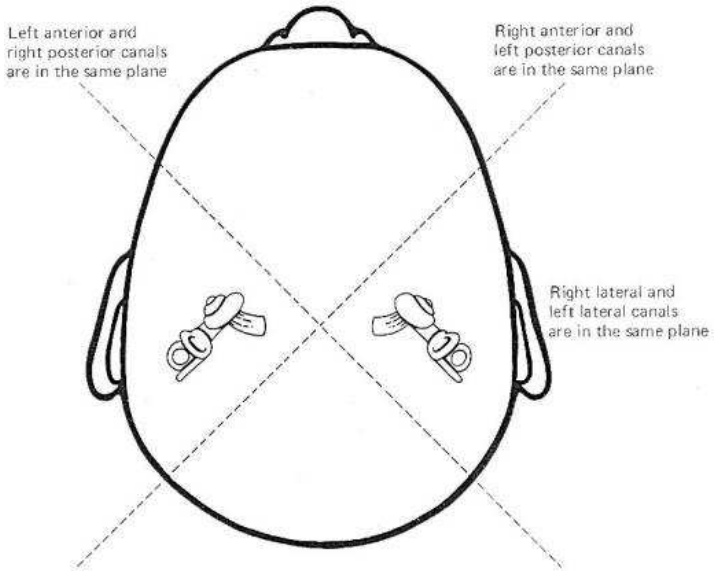


Figure 8.4: The vestibular organs are located behind the ears. (Figure from CNS Clinic Jordan.)

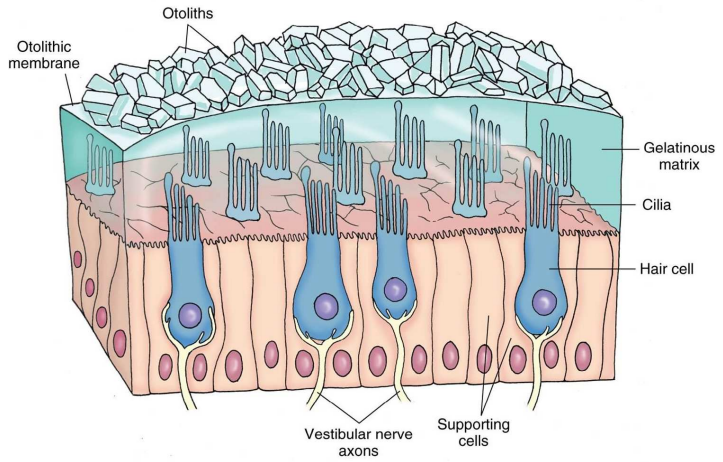


Figure 8.5: A depiction of an otolith organ (utricle or saccule), which senses linear acceleration. (Figure by Lippincott Williams & Wilkins.)

senses acceleration components a_x and a_z, and the saccule senses a_y and a_z (a_z is redundantly sensed).

The *semicircular canals* measure angular acceleration. Each canal has a diameter of about 0.2 to 0.3mm, and is bent along a circular arc with a diameter of about 2 to 3mm. Amazingly, the three canals are roughly perpendicular so that they independently measure three components of angular velocity. The *anterior canal* lies in the saggital plane (or yz plane), enabling it to measure pitch acceleration, α_x (changing the rate of rotation about the x axis). Similarly, the *posterior canal* lies in the perpendicular vertical plane, also called *frontal plane* or xy plane, and measures roll acceleration, α_z. Lastly, the *lateral canal* lies in the horizontal or xz plane, and measures yaw acceleration, α_y. Note from Figure 8.4 that each set of canals is rotated by 45 degrees with respect to the vertical axis. Thus, the anterior canal of the left ear aligns with the posterior canal of the right ear. Likewise, the posterior canal of the left ear aligns with the anterior canal of the right ear.

Sensing linear acceleration To understand how accelerations are sensed, we start with the case of the otolith system. Figure 8.5 shows a schematic representation of an otolith organ, which may be either the utricle or saccule. Mechanoreceptors, in the form of *hair cells*, convert acceleration into neural signals. Each hair cell has *cilia* that are embedded in a gelatinous matrix. Heavy weights lie on top of the matrix so that when acceleration occurs laterally, the shifting weight applies a shearing force that causes the cilia to bend. The higher the acceleration magnitude, the larger the bending, and a higher rate of neural impulses become transmitted. Two dimensions of lateral deflection are possible. For example, in the case of the utricle, linear acceleration in any direction in the xz plane would

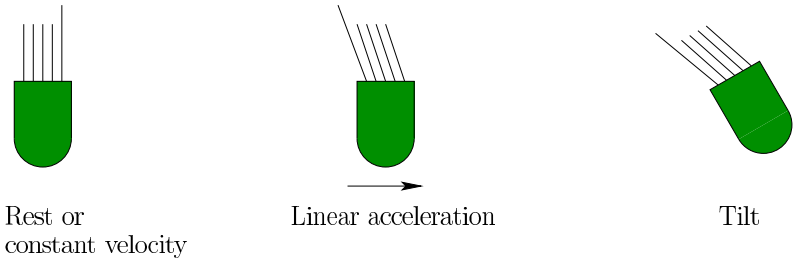


Figure 8.6: The otolith organs cannot distinguish linear acceleration of the head from tilt with respect to gravity. In either case, the cilia deflect in the same way, sending equivalent signals to the neural structures.

cause the cilia to bend. To distinguish between particular directions inside of this plane, the cilia are *polarized* so that each cell is sensitive to one particular direction. This is accomplished by a thicker, lead hair called the *kinocilium*, to which all other hairs of the cell are attached by a ribbon across their tips so that they all bend together.

One major sensing limitation arises because of a fundamental law from physics: The *Einstein equivalence principle*. In addition to the vestibular system, it also impacts VR tracking systems (see Section **??**). The problem is gravity. If we were deep in space, far away from any gravitational forces, then linear accelerations measured by a sensor would correspond to pure accelerations with respect to a fixed coordinate frame. On the Earth, we also experience force due to gravity, which feels as if were on a rocket ship accelerating upward at roughly 9.8m/s^2. The equivalence principle states that the effects of gravity and true linear accelerations on a body are indistinguishable. Figure 8.6 shows the result in terms of the otolith organs. The same signals are sent to the brain whether the head is tilted or it is linearly accelerating. If you close your eyes or wear a VR headset, then you should not be able to distinguish tilt from acceleration. In most settings, we are not confused because the vestibular signals are accompanied by other stimuli when accelerating, such as a revving engine.

Sensing angular acceleration The semicircular canals use the same principle as the otolith organs. They measure acceleration by bending cilia at the end of hair cells. A viscous fluid moves inside of each canal. A flexible structure called the *cupula* blocks one small section of the canal and contains the hair cells; see Figure 8.7. Compare the rotation of a canal to the merry-go-round. If we were to place a liquid-filled tube around the periphery of the merry-go-round, then the fluid would remain fairly stable at a constant angular velocity. However, if angular acceleration is applied, then due to friction between the fluid and the tube and also internal fluid viscosity, the fluid would start to travel inside the tube. In the semicircular canal, the moving fluid applies pressure to the cupula, causing it to deform and bend the cilia on hair cells inside of it. Note that a constant angular

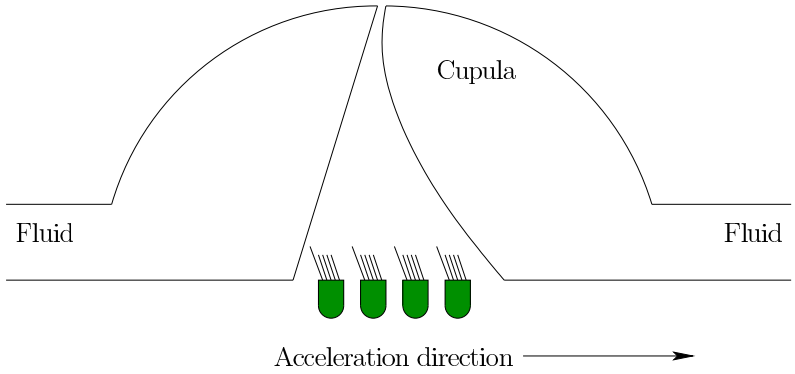


Figure 8.7: The cupula contains a center membrane that houses the cilia. If angular acceleration occurs that is aligned with the canal direction, then the pressure is applied to the cupula, whic causes the cilia to bend and send neural signals.

velocity does not, in principle, cause pressure on the cupula; thus, the semicircular canals measure angular *acceleration* as opposed to velocity.

Each canal is polarized in the sense that it responds mainly to rotations about an axis perpendicular to the plane that contains the entire canal. The left and right lateral canals lie in a common, horizontal plane to respond to the yaw component of angular acceleration. The other canals lie in vertical planes that are appromixately oriented 45 degrees with respect to the forward direction, as shown in Figure 8.4.

Impact on perception Cues from the vestibular system are generally weak in comparison to other senses, especially vision. For example, a common danger for a skier buried in an avalanche is that he cannot easily determine which way is up without visual cues to accompany the perception of gravity from the vestibular system. Thus, the vestibular system functions well when providing consistent cues with other systems, including vision and proprioception. Mismatched cues are problematic. For example, some people may experience *vertigo* when the vestibular system is not functioning correctly. In this case, they feel as if the world around them is spinning or swaying. Common symptoms are nausea, vomiting, sweating, and difficulties walking. This may even impact eye movements because of the VOR. Section 8.4 explains a bad side-effect that results from mismatched vestibular and visual cues in VR.

8.3 Physics in the Virtual World

8.3.1 Tailoring the Physics to the Experience

If we expect to fool our brains into believing that we inhabit the virtual world, then many of our expectations from our experiences in the real world should be

matched in the virtual world. We have already seen this in the case of the physics of light, from Chapter 4, applying to visual rendering of virtual worlds, covered in Chapter 7. Motions in the virtual world should also behave in a familiar way.

This implies that the VWG contains a *physics engine* that governs the motions of bodies in the virtual world by following physical principles from the real world. Issues such as forces acting on bodies, gravity, fluid flows, and collisions between bodies should be handled in perceptually convincing ways. Physics engines arise throughout engineering and physics in the context of any simulation. In video games, computer graphics, and film, these engines perform operations that are very close to our needs for VR. This is why popular game engines such as Unity 3D and Unreal Engine have been quickly adapted for use in VR. As stated in Section 2.2, we have not yet arrived at an era in which general and widely adopted VR engines exist; therefore, modern game engines are worth understanding and utilizing at present.

To determine what kind of physics engine needs to be borrowed, adapted, or constructed from scratch, one should think about the desired VR experience and determine the kinds of motions that will arise. Some common, generic questions are:

- Will the matched zone remain fixed, or will the user need to be moved by locomotion? If locomotion is needed, then will the user walk, run, swim, drive cars, or fly spaceships?
- Will the user interact with objects? If so, then what kind of interaction is needed? Possibilities include carrying weapons, opening doors, tossing objects, pouring drinks, operating machinery, and assembling structures.
- Will multiple users be sharing the same virtual space? If so, then how will their motions be coordinated or constrained?
- Will the virtual world contain entities that appear to move autonomously, such as robots, animals, or humans?
- Will the user be immersed in a familiar or exotic setting? A familiar setting could be a home, classroom, park, or city streets. An exotic setting might be scuba diving, lunar exploration, or traveling through blood vessels.

In addition to the physics engine, these questions will also guide the design of the interface, which is addressed in Chapter 10.

Based on the answers to these questions, the physics engine design may be simple and efficient, or completely overwhelming. As mentioned in Section 7.4, a key challenge is to keep the virtual world frequently updated so that interactions between users and objects are well synchronized and renderers provide a low-latency projection onto displays.

Furthermore, the goal may not always be to perfectly match what would happen in the physical world. In a familiar setting, we might expect significant matching; however, in exotic settings, it often becomes more important to make a comfortable experience, rather than matching reality perfectly. Even in the case of simulating walking around in the world, we often want to deviate from real-world physics because of vection, which causes VR sickness and is discussed in Section 8.4.

(a) (b)

Figure 8.8: (a) A virtual car (Cheetah) that appears in the game Grand Theft Auto; how many degrees of freedom should it have? (b) A human skeleton, with rigid bodies connected via joints, commonly underlies the motions of an avatar. (Figure from SoftKinetic).

The remainder of this section covers some fundamental aspects that commonly arise: 1) numerical simulation of physical systems, 2) the control of systems using human input, and 3) collision detection, which determines whether bodies are interfering with each other.

8.3.2 Numerical simulation

The state of the virtual world Imagine a virtual world that contains many moving rigid bodies. For each body, think about its degrees of freedom (DOFs), which corresponds to the number of independent parameters needed to uniquely determine its position and orientation. We would like to know the complete list of parameters needed to put every body in its proper place in a single time instant. A specification of values for all of these parameters is defined as the *state* of the virtual world.

The job of the physics engine can then be described as calculating the virtual world state for every time instant or "snapshot" of the virtual world that would be needed by a rendering system. Once the state is determined, the mathematical transforms of Chapter 3 are used to place the bodies correctly in the world and calculate how they should appear on displays.

Degrees of freedom How many parameters are there in a virtual world model? As discussed in Section 3.2, a free-floating body has 6 DOFs. In many cases, DOFs are lost due to constraints. For example, a ball that rolls on the ground has only 5 DOFs because it can achieve any 2D position along the ground and also have any 3D orientation. It might be sufficient to describe a car with 3 DOFs by specifying the position along the ground (two parameters) and the direction it is facing (one parameter); see Figure 8.8(a). However, if the car is allowed to fly through the air while performing stunts or crashing, then all 6 DOFs are needed.

For many models, rigid bodies are attached together in a way that allows relative motions. This is called *multibody kinematics* [77, 139]. For example, a car usually has 4 wheels which can each roll to provide one rotational DOF. Furthermore, the front wheels can be steered to provide an additional DOF. Steering usually turns the front wheels in unison, which implies that one DOF is sufficient to describe both wheels. If the car has a complicated suspension system, then it cannot be treated as a single rigid body, which would add many more DOFs.

Similarly, an animated character can be made by attaching rigid bodies to form a skeleton; see Figure 8.8(b). Each rigid body in the skeleton is attached to one or more other bodies by a *joint*. For example, a simple human character can be formed by attaching arms, legs, and a neck to a rigid torso. The upper left arm is attached to the torso by a shoulder joint. The lower part of the arm is attached by an elbow joint, and so on. Some joints allow more DOFs than others. For example, the shoulder joint has 3 DOFs because it can yaw, pitch, and roll with respect to the torso, but an elbow joint has only one DOF.

To fully model the flexibility of the human body, 244 DOFs are needed, which are controlled by 630 muscles [169]. In many settings, this would be too much detail, which might lead to high computational complexity and difficult implementation. Furthermore, one should always beware of the uncanny valley (mentioned in Section 1.1), in which more realism might lead to increased perceived creepiness of the character. Thus, having more DOFs is not clearly better, and it is up to a VR content creator to determine how much mobility is needed to bring a character to life, in a way that is compelling for a targeted purpose.

In the extreme case, rigid bodies are not sufficient to model the world. We might want to see waves rippling realistically across a lake, or hair gently flowing in the breeze. In these general settings, *nonrigid models* are used, in which case the state can be considered as a continuous function. For example, a function of the form $y = f(x, z)$ could describe the surface of the water. Without making some limiting simplifications, the result could effectively be an infinite number of DOFs. Motions in this setting are typically described using *partial differential equations* (*PDEs*), which are integrated numerically to obtain the state at a desired time. Usually, the computational cost is high enough to prohibit their use in interactive VR experiences, unless shortcuts are taken by precomputing motions or dramatically simplifying the model [].

Differential equations We now introduce some basic differential equations to model motions. The resulting description is often called a *dynamical system*. The first step is to describe rigid body velocities in terms of state. Returning to models that involve one or more rigid bodies, the state corresponds to a finite number of parameters. Let

$$x = (x_1, x_2, \ldots, x_n) \tag{8.17}$$

denote an n-dimensional *state vector*. If each x_i corresponds to a position or orientation parameter for a rigid body, then the state vector puts all bodies in their place. Let

$$\dot{x}_i = \frac{dx_i}{dt} \tag{8.18}$$

represent the time derivative, or velocity, for each parameter.

To obtain the state at any time t, the velocities need to be integrated over time. Following (8.3), the integration of each state variable determines the value at time t:

$$x_i(t) = x_i(0) + \int_0^t \dot{x}_i(s)ds, \tag{8.19}$$

in which $x_i(0)$ is the value of x_i at time $t = 0$.

Two main problems arise with (8.19):

1. The integral almost always must be evaluated numerically.
2. The velocity $\dot{x}_i(t)$ must be specified at each time t.

Sampling rate For the first problem, time is discretized into steps, in which Δt is the *step size* or *sampling rate*. For example, Δt might be 1ms, in which case the state can be calculated for times $t = 0, 0.001, 0.002, \ldots$, in terms of seconds. This can be considered as a kind of frame rate for the physics engine. Each Δt corresponds to the production of a new frame.

As mentioned in Section 7.4, the VWG should synchronize the production of virtual world frames with rendering processes so that the world is not caught in an intermediate state with some variables updated to the new time and others stuck at the previous time. This is a kind of tearing in the virtual world. This does not, however, imply that the frame rates are the same between renderers and the physics engine. Typically, the frame rate for the physics engine is much higher to improve numerical accuracy.

Using the sampling date Δt, (8.19) is approximated as

$$x((k+1)\Delta t) \approx x(k\Delta t) + \sum_{j=1}^{k} \dot{x}_i(j\Delta t)\Delta t. \tag{8.20}$$

It is simpler to view (8.20) one step at a time. Let $x[k]$ denote $x(k\Delta t)$, which is the state at time $t = k\Delta t$. The following is an update law that expresses the new state $x[k+1]$ in terms of the old state $x[k]$:

$$x[k+1] \approx x[k] + \dot{x}(k\Delta t)\Delta t, \tag{8.21}$$

which starts with $x[0] = x(0)$.

Runge-Kutta integration The approximation used in (8.20) is known as Euler integration. It is the simplest approximation, but does not perform well enough in many practical settings. One of the most common improvements is the fourth-order *Runge-Kutta integration* method, which expresses the new state as

$$x[k+1] \approx x[k] + \frac{\Delta t}{6}(w_1 + 2w_2 + 2w_3 + w_4), \tag{8.22}$$

in which

$$\begin{aligned} w_1 &= f(\dot{x}(k\Delta t)) \\ w_2 &= f(\dot{x}(k\Delta t + \tfrac{1}{2}\Delta t) + \tfrac{1}{2}\Delta t\ w_1) \\ w_3 &= f(\dot{x}(k\Delta t + \tfrac{1}{2}\Delta t) + \tfrac{1}{2}\Delta t\ w_2) \\ w_4 &= f(\dot{x}(k\Delta t + \Delta t) + \Delta t\ w_3). \end{aligned} \tag{8.23}$$

Although this is more expensive than Euler integration, the improved accuracy is usually worthwhile in practice. Many other methods exist, with varying performance depending on the particular ways in which $\dot{x}$ is expressed and varies over time [].

Time-invariant dynamical systems The second problem from (8.19) is to determine an expression for $\dot{x}(t)$. This is where the laws of physics, such as the acceleration of rigid bodies due to applied forces and gravity. The most common case is *time-invariant dynamical systems*, in which $\dot{x}$ depends only on the current state and not the particular time. This means each component x_i is expressed as

$$\dot{x}_i = f_i(x_1, x_2, \ldots, x_n), \tag{8.24}$$

for some given function f. This can be written in compressed form by using x and $\dot{x}$ to represent n-dimensional vectors:

$$\dot{x} = f(x). \tag{8.25}$$

The expression above is often called the *state transition equation* because it indicates the rate of change of state.

Here is a simple, one-dimensional example of a state transition equation:

$$\dot{x} = 2x - 1. \tag{8.26}$$

This is called a *linear* differential equation. The velocity $\dot{x}$ roughly doubles with the value of x. Fortunately, linear problems can be fully solved "on paper". The solution to (8.26) is of the general form

$$x(t) = \frac{1}{2} + ce^{2t}, \tag{8.27}$$

in which c is a constant that depends on the given value for $x(0)$.

The phase space Unfortunately, motions are usually described in terms of accelerations (and sometimes higher-order derivatives), which need to be integrated twice. This leads to higher-order differential equations, which are difficult to work with. For this reason, *phase space* representations were developed in physics and engineering. In this case, the velocities of the state variables are themselves treated as state variables. That way, the accelerations become the velocities of the velocity variables.

For example, suppose that a position x_1 is acted upon by gravity, which generates an acceleration $a = -9.8\text{m/s}^2$. We introduce a second variable x_2, which is defined as the velocity of x_1. Thus, by definition, $\dot{x}_1 = x_2$. Furthermore, $\dot{x}_2 = a$ because the derivative of velocity is acceleration. Both of these equations fit the form of (8.24). Generally, the number of states increases to incorporate accelerations (or even higher-order derivatives), but the resulting dynamics are expressed in the form (8.24), which is easier to work with.

Handling user input Now consider the case in which a user commands an object to move. Examples include driving a car, flying a spaceship, or walking an avatar around. This introduces some new parameters, called the *controls*, actions, or inputs to the dynamical system. Differential equations that include these new parameters are called *control systems* [9].

Let $u = (u_1, u_2, \ldots, u_m)$ be a vector of controls. The state transition equation in (8.25) is simply extended to include u:

$$\dot{x} = f(x, u). \tag{8.28}$$

Figure 8.9 shows a useful example, which involves driving a car. The control u_s determines the speed of the car. For example, $u_s = 1$ drives forward, and $u_s = -1$ drives in reverse. Setting $u_s = 10$ drives forward at a much faster rate. The control u_ϕ determines how the front wheels are steered. The state vector is (x, z, θ), which corresponds to the position and orientation of the car in the horizontal, xz plane.

The state transition equation is:

$$\begin{aligned} \dot{x} &= u_s \cos\theta \\ \dot{z} &= u_s \sin\theta \\ \dot{\theta} &= \frac{u_s}{L} \tan u_\phi. \end{aligned} \tag{8.29}$$

Using Runge-Kutta integration, or a similar numerical method, the future states can be calculated for the car, given inputs u_s and u_ϕ applied over time.

This model can also be used to steer the virtual walking of a VR user from first-person perspective. The viewpoint then changes according to (x, z, θ), while the height y remains fixed. For the model in (8.29), the car must drive forward or backward to change its orientation. By changing the third component to $\theta = u_\omega$, the user could instead specify the angular velocity directly. This would cause the user to rotate in place, as if on a merry-go-round. Many more examples like these appear in Chapter 13 of [77], including bodies that are controlled via accelerations.

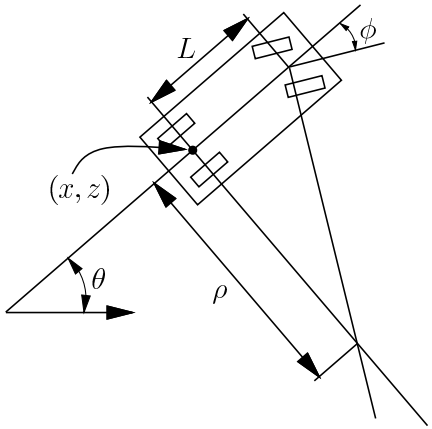


Figure 8.9: A top-down view of a simple, steerable car. Its position and orientation are given by (x, y, θ). The parameter ρ is the minimum turning radius, which depends on the maximum allowable steering angle ϕ. This model can also be used to "steer" human avatars, by placing the viewpoint above the center of the rear axle.

It is sometimes helpful conceptually to define the motions in terms of stages. Using numerical integration of (8.28), we can think about applying a control u over time Δt to obtain a new state $x[k+1]$:

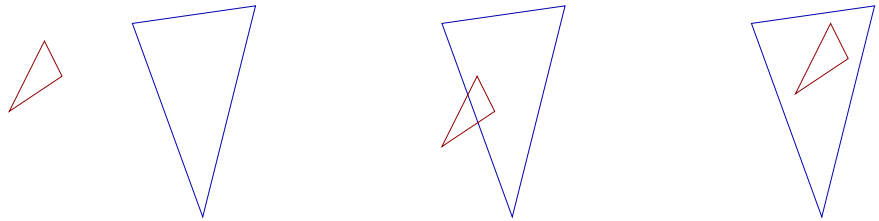
$$x[k+1] = F(x[k], u[k]). \tag{8.30}$$

The function F is obtained by integrating (8.28) over Δt. Thus, if the state is $x[k]$, and $u[k]$ is applied, then F calculates $x[k+1]$ as the next state.

8.3.3 Collision detection

One of the greatest challenges in building a physics engine is handling collisions between bodies. Standard laws of motion from physics or engineering usually do not take into account such interactions. Therefore, specialized algorithms are used to detect when such collisions occur and respond appropriately. Collision detection methods and corresponding software are plentiful because of widespread needs in computer graphics simulations and video games, and also for motion planning of robots.

Solid or boundary model? Figure 8.10 shows one the first difficulties with collision detection, in terms of two models in a 2D world. The first two cases (Figures 8.10(a) and 8.10(b)) show obvious cases; however, the third case Figures 8.10(c) could be ambiguous. If one triangle is wholly inside of another, then is this collision? If we interpret the outer triangle as a solid model, then YES. If the outer



(a) Collision free (b) In collision (c) Not sure

Figure 8.10: Three interesting cases for collision detection (these are 2D examples). The last case may or not cause collision, depending on the model.

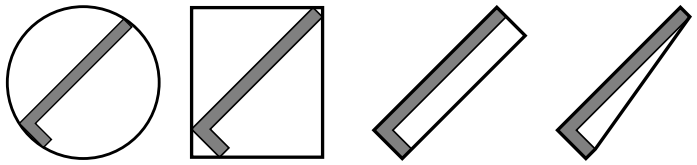
triangle is only the boundary edges, and is meant to have an empty interior, then the answer is NO. This is why emphasis was placed on having a coherent model in Section 3.1; otherwise, the boundary might not be established well enough to distinguish the inside from the outside.

Distance functions Many collision detection methods benefit from maintaining a distance function, which keeps track of how far the bodies are from colliding. For example, let A and B denote the set of all points occupied in $\mathbb{R}^3$ by two different models. If they are in collision, then their intersection $A \cap B$ is not empty. If they are not in collision, then the *Hausdorff distance* between A and B is the Euclidean distance between the closest pair of points, taking one from A and one from B.[3] Let $d(A, B)$ denote this distance. If A and B intersect, then $d(A, B) = 0$ because any point in $A \cap B$ will yield zero distance. If A and B do not intersect, then $d(A, B) > 0$, which implies that they are not in collision (in other words, collision free).

If $d(A, B)$ is large, then A and B are mostly likely to be collision free in the near future, even if one or both are moving. This leads to a family of collision detection methods called *incremental distance computation*, which assumes that between successive calls to the algorithm, the bodies move only a small amount. Under this assumption the algorithm achieves "almost constant time" performance for the case of convex polyhedral bodies [83, 98]. Nonconvex bodies can be decomposed into convex components.

A concept related to distance is *penetration depth*, which indicates how far one model is poking into another [**?**]. This is useful for setting a threshold on how much interference between the two bodies is allowed. For example, the user might be able to poke his head two centimeters into a wall, but beyond that, an action

[3] This assumes models contain all of the points on their boundary and that they have finite extent; otherwise, topological difficulties arise [61, 77]



(a) (b) (c) (d)

Figure 8.11: Four different kinds of bounding regions: (a) sphere, (b) axis-aligned bounding box (AABB), (c) oriented bounding box (OBB), and (d) convex hull. Each usually provides a tighter approximation than the previous one but is more expensive to test for overlapping pairs.

should be taken.

Simple collision tests At the lowest level, collision detection usually requires testing a pair of model primitives to determine whether they intersect. In the case of models formed from 3D triangles, then we need a method that determines whether two triangles intersect. This is similar to the ray-triangle intersection test that was needed for visual rendering in Section 7.1, and involves basic tools from analytic geometry, such as cross products and plane equations. Efficient methods are given in [54, 100].

Broad and narrow phases Suppose that a virtual world has been defined with millions of triangles. If two complicated, nonconvex bodies are to be checked for collision, then the computational cost may be high. For this complicated situation, collision detection often becomes a two-phase process:

1. **Broad Phase:** In the *broad phase*, the task is to avoid performing expensive computations for bodies that are far away from each other. Simple bounding boxes can be placed around each of the bodies, and simple tests can be performed to avoid costly collision checking unless the boxes overlap. Hashing schemes can be employed in some cases to greatly reduce the number of pairs of boxes that have to be tested for overlap [99].

2. **Narrow Phase:** In the *narrow phase* individual pairs of bodies are each checked carefully for collision. This involves the expensive tests, such as triangle-triangle intersection.

In the broad phase, *hierarchical methods* generally decompose each body into a tree. Each vertex in the tree represents a *bounding region* that contains some subset of the body. The bounding region of the root vertex contains the whole body. Two opposing criteria that guide the selection of the type of bounding region:

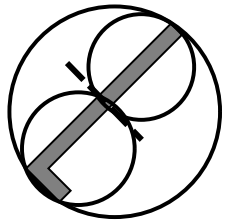


Figure 8.12: The large circle shows the bounding region for a vertex that covers an L-shaped body. After performing a split along the dashed line, two smaller circles are used to cover the two halves of the body. Each circle corresponds to a child vertex.

1. The region should fit the intended model points as tightly as possible.

2. The intersection test for two regions should be as efficient as possible.

Several popular choices are shown in Figure 8.11, for the case of an L-shaped body. Hierarchical methods are also useful for quickly eliminating many triangles from consideration in visual rendering, as mentioned in Section 7.1.

The tree is constructed for a body, A (or alternatively, B) recursively as follows. For each vertex, consider the set X of all points in A that are contained in the bounding region. Two child vertices are constructed by defining two smaller bounding regions whose union covers X. The split is made so that the portion covered by each child is of similar size. If the geometric model consists of primitives such as triangles, then a split could be made to separate the triangles into two sets of roughly the same number of triangles. A bounding region is then computed for each of the children. Figure 8.12 shows an example of a split for the case of an L-shaped body. Children are generated recursively by making splits until very simple sets are obtained. For example, in the case of triangles in space, a split is made unless the vertex represents a single triangle. In this case, it is easy to test for the intersection of two triangles.

Consider the problem of determining whether bodies A and B are in collision. Suppose that the trees T_a and T_b have been constructed for A and B, respectively. If the bounding regions of the root vertices of T_a and T_b do not intersect, then it is known that T_a and T_b are not in collision without performing any additional computation. If the bounding regions do intersect, then the bounding regions of the children of T_a are compared to the bounding region of T_b. If either of these intersect, then the bounding region of T_b is replaced with the bounding regions of its children, and the process continues recursively. As long as the bounding regions overlap, lower levels of the trees are traversed, until eventually the leaves are reached. At the leaves the algorithm tests the individual triangles for collision, instead of bounding regions. Note that as the trees are traversed, if a bounding region from the vertex v_1 of T_a does not intersect the bounding region from a vertex, v_2, of T_b, then no children of v_1 have to be compared to children of v_1.

Usually, this dramatically reduces the number of comparisons, relative to a naive approach that tests all pairs of triangles for intersection.

Mismatched obstacles in VR Although collision detection is a standard, well-solved problem, VR once again poses unusual challenges. One of the main difficulties is the matched zone, in which the real and virtual worlds share the same space. This leads to three interesting cases:

1. **Real obstacle only:** In this case, an obstacle exists in the real world, but not in the virtual world. This is potentially dangerous! For example, you could move your arm and knock over a real, hot cup of coffee that is not represented in the virtual world. If you were walking with a VR headset, then imagine what would happen if a set of real downward stairs were not represented. At the very least, the boundary of the matched zone should be rendered if the user were close to it. This mismatch motivated the introduction of the Chaperone system in the HTC Vive headset, in which an outward-facing camera is used to detect and render real objects that may obstruct user motion.

2. **Virtual obstacle only:** This case is not dangerous, but can be extremely frustrating. The user could poke her head through through a wall in VR without feeling any response in the real world. This should not be allowed in most cases. The VWG could simply stop moving the viewpoint in the virtual world as the virtual wall is contacted; however, this generates a mismatch between the real and virtual motions, which is uncomfortable for the user. It remains a difficult challenge to keep users comfortable while trying to guide them away from interference with virtual obstacles.

3. **Real and virtual obstacle:** If obstacles are matched in both real and virtual worlds, then the effect is perceptually powerful. For example, you might stand on a slightly raised platform in the real world while the virtual world shows you standing on a building rooftop. If the roof and platform edges align perfectly, then you could feel the edge with your feet. Would you be afraid to step over the edge? A simpler case is to render a virtual chair that matches the real chair that a user might be sitting in.

8.4 Mismatched Motion and Vection

Vection was mentioned in Section 2.3 as an illusion of self motion that is caused by varying visual stimuli. In other words, the brain is tricked into believing that the head is moving based on what is seen, even though no motion actually occurs. Figure 2.20 showed the haunted swing illusion, which convinced people that were swinging upside down; however, the room was moving while they were stationary. Vection is also commonly induced in VR by moving the user's viewpoint while there is no corresponding motion in the real world.

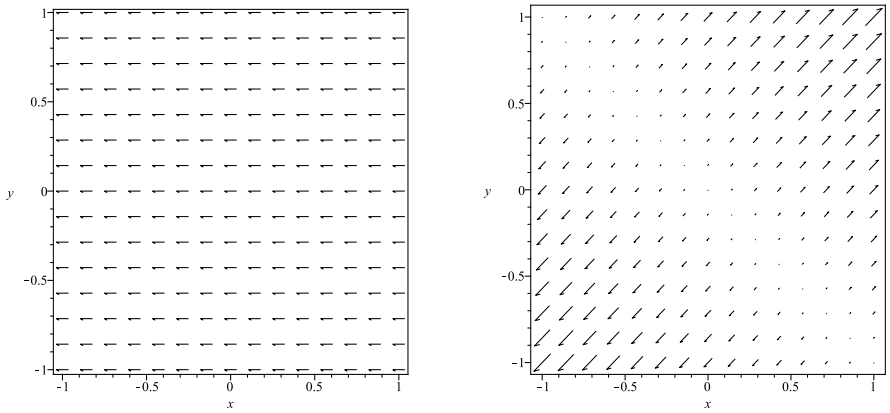
Figure 8.13: The optical flow of features in an image due to motion in the world. These were computed automatically using image processing algorithms. (Image by Andreas Geiger, from Max Planck Institute for Intelligent Systems in Tübingen.)

Vection is a prime example of mismatched cues, which were discussed in Section 6.4. Whereas the McGurk effect has no harmful side effects, vection unfortunately leads many people to experience sickness symptoms, such as dizziness, nausea, and occasionally even vomiting. Thus, it should be used very sparingly, if at all, for VR experiences. Furthermore, if it is used, attempts should be made to alter the content so that the side effects are minimized. Industry leaders often proclaim that their latest VR headset has beaten the VR sickness problem; however, this neglects the following counterintuitive behavior:

> **If a headset is better in terms of spatial resolution, frame rate, tracking accuracy, field of view, and latency, then the potential is higher for making people sick through vection and other mismatched cues.**

Put simply and intuitively, if the headset more accurately mimics reality, then the sensory cues are stronger, and our perceptual systems become more confident about mismatched cues. It may even have the ability to emulate poorer headsets, resulting in a way to comparatively assess side effects of earlier VR systems. In some cases, the mismatch of cues may be harmless (although possibly leading to a decreased sense of presence). In other cases, the mismatches may lead to greater fatigue as the brain works harder to resolve minor conflicts. In the worst case, VR sickness emerges, with vection being the largest culprit based on VR experiences being made today. One of the worst cases is the straightforward adaptation of first-person shooter games to VR, in which the vection occurs almost all the time as the avatar explores the hostile environment.

Optical flow Recall from Section 6.2, that the human visual system has neural structures dedicated to detecting the motion of visual features in the field of view; see Figure 8.13. It is actually the images of these features that move across the retina. It is therefore useful to have a mathematical concept that describes the



(a) (b)

Figure 8.14: Example vector fields: (a) A constant vector field, for which every vector is $(-1, 0)$, regardless of the location. (b) In this vector field, $(x, y) \mapsto (x + y, x + y)$, the vectors point away from the diagonal line from $(-1, 1)$ to $(1, -1)$, and their length is proportional to the distance from it.

velocities of moving points over a surface. We therefore define a *vector field*, which assigns a velocity vector at every point along a surface. If the surface is the xy plane, then a velocity vector

$$(v_x, v_y) = \left(\frac{dx}{dt}, \frac{dy}{dt} \right) \tag{8.31}$$

is assigned at every point (x, y). For example,

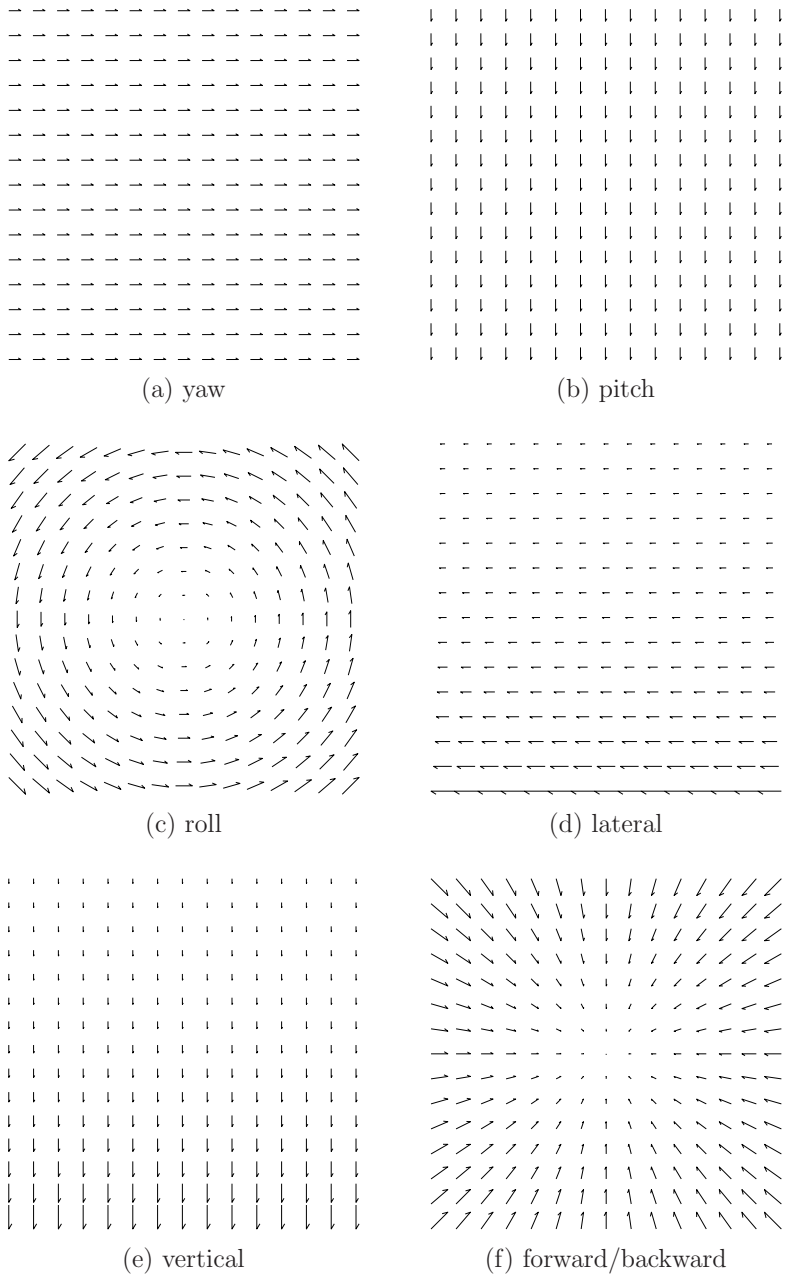
$$(x, y) \mapsto (-1, 0) \tag{8.32}$$

is a *constant vector field*, which assigns $v_x = -1$ and $v_y = 0$ everywhere; see Figure 8.14(a). The vector field

$$(x, y) \mapsto (x + y, x + y) \tag{8.33}$$

is non-constant, and assigns $v_x = v_y = x + y$ at each point (x, y); see Figure 8.14(b). For this vector field, the velocity direction is always diagonal, but the length of the vector (speed) depends on $x + y$.

To most accurately describe the motion of features along the retina, the vector field should be defined over a spherical surface that corresponds to the locations of the photoreceptors. Instead, we will describe vector fields over a square region, with the understanding that that it should be transformed onto a sphere for greater accuracy.



(a) yaw (b) pitch

(c) roll (d) lateral

(e) vertical (f) forward/backward

Figure 8.15: Six different types of optical flows, based on six degrees of freedom for motion of a rigid body. Each of these is a contributing component of vection.

Types of vection Vection can be caused by any combination of angular and linear velocities of the viewpoint in the virtual world. To characterize the effects of different kinds of motions effectively, it is convenient to decompose the viewpoint velocities into the three linear components, v_x, v_y, and v_z, and three angular components, ω_x, ω_y, and ω_z. Therefore, we consider the optical flow for each of these six cases (see Figure 8.15):

1. **Yaw vection:** If the viewpoint is rotated counterclockwise about the y axis (positive ω_y), then all visual features move from right to left at the same velocity. Equivalently, the virtual world is rotating clockwise around the user; however, self motion in the opposite direction is perceived. This causes the user to feel as if she is riding a merry-go-round (recall Figure 8.2).

2. **Pitch vection:** By rotating the viewpoint counterclockwise about the x axis (positive ω_x), the features move downward at the same velocity.

3. **Roll vection:** Rotating the viewpoint counterclockwise about z, the optical axis (positive ω_z), causes the features to rotate clockwise around the center of the image. The velocity of each feature is tangent to the circle that contains it, and the speed is proportional to the distance from the feature to the image center.

4. **Lateral vection:** In this case, the viewpoint is translated to the right, corresponding to positive v_x. As a result, the features move horizontally; however, there is an important distinction with respect to yaw vection: Features that correspond to closer objects move more quickly than those from distant objects. Figure 8.15(d) depicts the field by assuming vertical position of the feature corresonds to its depth (lower in the depth field is closer). This is a reappearance of *parallax*, which in this case gives the illusion of lateral motion and distinguishes it from yaw motion.

5. **Vertical vection:** The viewpoint is translated upward, corresponding to positive v_x, and resulting in downward flow. Once again, parallax causes the speed of features to depend on the distance of the corresponding object. This enables vertical vection to be distinguished from pitch vection.

6. **Forward/backward vection:** If the viewpoint is translated along the optical axis away from the scene (positive v_z), then the features flow inward toward the image center, as shown in Figure 8.15(f). Their speed depends on *both* their distance from the image center and the distance of their corresponding objects in the virtual world. The resulting illusion is backward motion. Translation in the negative z direction results in perceived forward motion (as in the case of the Millennium Falcon spaceship after its jump to hyperspace in the Star Wars movies).

The first two are sometimes called *circular vection*, and the last three are known as *linear vection*. Since our eyes are drawn toward moving features, changing the viewpoint may trigger smooth pursuit eye movements (recall from Section 5.3). In this case, the optical flows shown in Figure 8.15 would not correspond to the motions of the features on the retina. Thus, our characterization so far ignores eye movements, which are often designed to counteract optical flow to provide stable images on the retina. Nevertheless, due the proprioception, the brain is aware of these eye rotations, which results in an equivalent perception of self motion.

All forms of vection cause perceived velocity, but the perception of acceleration is more complicated. First consider pure rotation of the viewpoint. Angular acceleration is perceived if the rotation rate of yaw, pitch, and roll vection are varied. Linear acceleration is also perceived, even in the case of yaw, pitch, or roll vection at constant angular velocity. This is due to the merry-go-round effect, which was shown in Figure 8.2(b).

Now consider pure linear vection (no rotation). Any linear acceleration of the viewpoint will be perceived as an acceleration. However, if the viewpoint moves at constant velocity, then this is the only form of vection in which there is no perceived acceleration. In a VR headset, the user may nevertheless perceive accelerations due to optical distortions or other imperfections in the rendering and display.

Vestibular mismatch We have not yet considered the effect of each of these six cases in terms of their mismatch with vestibular cues. If the user is not moving relative to the Earth, then only gravity should be sensed by the vestibular organ (in particular, the otolith organs). Suppose the user is facing forward without any tilt. In this case, any perceived acceleration from vection would cause a mismatch. For example, yaw vection should cause a perceived constant acceleration toward the rotation center (recall Figure 8.2(b)), which mismatches the vestibular gravity cue. As another example, downward vertical vection should cause the user to feel like he is falling, but the vestibular cue would indicate otherwise.

For cases of yaw, pitch, and roll vection at constant angular velocity, there may not be a conflict with rotation sensed by the vestibular organ because the semicircular canals measure angular accelerations. Thus, the angular velocity of the viewpoint must change to cause mismatch with this part of the vestibular system.

If the head is actually moving, then the vestibular organ is stimulated. This case is more complicated to understand because vestibular cues that correspond to linear and angular accelerations in the real world are combined with visual cues that indicate different accelerations. In some cases, these cues may be more consistent, and in other cases, they may diverge further.

Factors that affect sensitity The potency of vection is affected by many factors:

- **Percentage of field of view:** If only a small part of the visual field is moving, then we tend to perceive that it is caused by a moving object. However, if most of the visual field is moving, then we perceive *ourselves* as

moving. Our visual system actually includes neurons with receptive fields that cover a large fraction of the retina for the purpose of detecting self motion [23]. As VR headsets have increased their field of view, they project onto a larger region of the retina, thereby strengthening vection cues.

- **Distance from center view:** Recall from Section 5.1 that the photoreceptors are not uniformly distributed, with the highest density being at the innermost part of the fovea. Thus, detection may seem stronger near the center. However, in the cases of yaw and forward/backward vection, the optical flow vectors are stronger at the periphery, which indicates that detection may be stronger at the periphery. Sensitivity to the optical flow may therefore be strongest somewhere between the center view and the periphery, depending on the viewpoint velocities, distances to objects, photoreceptor densities, and neural detection mechanisms.

- **Exposure time:** The perception of self motion due to vection increases with the time of exposure to the optical flow. If the period of exposure is very brief, such as a few milliseconds, then no vection may occur.

- **Spatial frequency:** If the virtual world is complicated, with many small structures or textures, then the number of visual features will be greatly increased and the optical flow becomes a stronger signal. As the VR headset display resolution increases, higher spatial frequencies can be generated.

- **Contrast:** With higher levels of contrast, the optical flow signal is stronger because the feature are more distinguishable. Therefore, vection typically occurs more quickly [**?**].

- **Other sensory cues:** Recall from Section 6.4 that a perceptual phenomenon depends on the combination of many cues. Vection can be enhanced by providing additional consistent cues. For example, forward vection could be accompanied by a fan blowing in the user's face, a rumbling engine, and the sounds of stationary objects in the virtual world racing by. Likewise, vection can be weakened by providing cues that are consistent with the real world, where no corresponding motion is occurring.

- **Prior knowledge:** Just by knowing beforehand what kind of motion should be perceived will affect the onset of vection []. This induces a prior bias that might take longer to overcome if the bias is against self motion, but less time to overcome it is consistent with self motion. The prior bias could be from someone telling the user what is going to happen, or it could simply by from an accumulation of similar visual experiences through the user's lifetime. Furthermore, the user might expect the motion as the result of an action taken, such as turning the steering wheel of a virtual car.

- **Attention:** If the user is distracted by another activity, such as aiming a virtual weapon or selecting a menu option, then vection and its side effects may be mitigated [].

- **Prior training or adaptation:** With enough exposure, the body may learn to distinguish vection from true motion to the point that vection becomes comfortable. Thus, many users can be trained to overcome VR sickness through repeated, prolonged exposure [].

Due to all of these factors, and the imperfections of modern VR headsets, it becomes extremely difficult to characterize the potency of vection and its resulting side effects on user comfort.

Further Reading

All about game engines: [52]

Vestibular system: [69]

Kinematics and kinetics of Human Motion: [167, 168]

Velocity verlet method of numerical integration? RK methods

The role of central and peripheral vision in perceiving the direction of self-motion, William H. Warren, Kenneth J. Kurtz, Perception & Psychophysics September 1992, Volume 51, Issue 5, pp 443-454

Vestibular Sensitivity and Vection Chronometry along the Spinal Axis in Erect Man, Jean-Claude Lepecq Irini Giannopulu Sophie Mertz Pierre-Marie Baudonnire

Cognitive Effects on Visually Induced Body Motion in Children Jean-Claude Lepecq Irini Giannopulu Pierre-Marie Baudonniere

Spatiotemporal boundaries of linear vection Xavier M. Sauvan, Claude Bonnet, Perception & Psychophysics, January 1995, Volume 57, Issue 6, pp 898-904

Viewing angle effects from wide field video projection images on the human equilibrium, Masaki Emoto, Kenichiro Masaoka, Masayuki Sugawara, Fumio Okano, Displays, Volume 26, Issue 1, January 2005, Pages 914

Separate attentional resources for vision and audition, David Alais, Concetta Morrone, David Burr, Proc. Royal Socierty B, June 2006.

Effects of Field of View on Performance with Head-Mounted Displays, PhD thesis, University of North Carolina at Chapel Hill, by K. W. Arthur - 2000

Collision detection survey: [84]

OBB trees: [51]

Chapter 9

Tracking

Chapter Status	Taken from *Virtual Reality*, S. M. LaValle
	This online chapter is not the final version! Check **http://vr.cs.uiuc.edu/** for information on the latest draft version. This draft was compiled on June 29, 2016.

Keeping track of moving bodies in the physical world is a crucial part of any VR system. Tracking was one of the largest obstacles to overcome to bring VR headsets into consumer electronics, and it will remain a major challenge due to our desire to expand and improve VR experiences. Highly accurate tracking methods have been mostly enabled by commodity hardware components, such as inertial measurement units (IMUs) and cameras, that have plummeted in size and cost due to the smartphone industry.

Three categories of tracking may appear in VR systems:

1. **The user's sense organs:** Recall from Section 2.1 that sense organs, such as eyes and ears, have DOFs that are controlled by the body. If artificial stimuli are attached to a sense organ but they are meant to be perceived as attached to the surrounding world, then the position and orientation of the organ needs to be tracked. The inverse of the tracked transformation is applied to the stimulus to correctly "undo" these DOFs. Most of the focus is on *head tracking*, which is sufficient for visual and aural components of VR; however, the visual system may further require *eye tracking* if the rendering and display technology requires compensating for the eye movements discussed in Section 5.3.

2. **The user's other body parts:** If the user would like to see a compelling representation of his body in the virtual world, then its motion needs to be tracked so that it can be reproduced in the matched zone. Perhaps facial expressions or hand gestures are needed for interaction. Although perfect matching is ideal for tracking sense organs, it is not required for tracking other parts. Small movements in the real world could convert into larger virtual world motions so that the user exerts less energy. In the limiting case, the user could simply press a button to change the body configuration. For example, she might grasp an object in her virtual hand by a single click.

3. **The rest of the environment:** Physical objects in the real world that surrounds the user may need to be tracked. For objects that exist in the physical world but not the virtual world, it might be necessary to alert the user to their presence for safety reasons. Imagine the user is about to hit a wall, or trip over a toddler. In some VR applications, the tracked physical objects may be matched in VR so that the user receives touch feedback while interacting with them. In other applications, such as telepresence, a large part of the physical world could be "brought into" the virtual world through live capture.

Section 9.1 covers the easy case of tracking rotation around a single axis to prepare for Section 9.2, which extends the framework to tracking the 3-DOF orientation of a 3D rigid body. This relies mainly on the angular velocity readings of an IMU. The most common use is to track the head that wears a VR headset. Section 9.3 addresses the tracking of position and orientation together, which in most systems requires line-of-sight visibility between a fixed part of the physical world and the object being tracked. Section 9.4 discusses the case of tracking multiple bodies that are attached together by joints. Finally, Section 9.5 covers the case of using sensors to build a representation of the physical world so that it can be brought into the virtual world.

9.1 Tracking 2D Orientation

This section explains how the orientation of a rigid body is estimated using an IMU. The main application is determining the viewpoint orientation, R_{eye} from Section 3.4 while the user is wearing a VR headset. Another application is estimating the orientation of a hand-held controller. For example, suppose we would like to make a laser pointer that works in the virtual world, based on a direction indicated by the user. The location of a bright red dot in the scene would be determined by the estimated orientation of a controller. More generally, the orientation of any human body part or moving object in the physical world can be determined if it has an attached IMU.

To estimate orientation, we first consider the 2D case by closely following the merry-go-round model of Section 8.1.2. The main issues are easy to visualize in this case, and extend to the more important case of 3D rotations. Thus, imagine that we mount a gyroscope on a spinning merry-go-round. Its job is to measure the angular velocity as the merry-to-round spins. It will convenient throughout this chapter to distinguish a true parameter value from an estimate. To accomplish

this, a "hat" will be placed over estimates. Thus, let $\hat{\omega}$ correspond to the estimated or measured angular velocity, which may not be same as ω, the *true* value.

How are $\hat{\omega}$ and ω related? If the gyroscope were functioning perfectly, then $\hat{\omega}$ would equal ω; however, in the real world this cannot be achieved. The main contributor to the discrepancy between $\hat{\omega}$ and ω is *calibration error*. The quality of calibration is the largest differentiator between an expensive (thousands of dollars) and cheap (one dollar) IMU.

We now define a simple model of calibration error. The following *sensor mapping* indicates how the sensor output is related to true angular velocity:

$$\hat{\omega} = a + b\,\omega. \tag{9.1}$$

Above, a and b are called the *offset* and *scale*, respectively. They are unknown constants that interfere with the measurement. If ω were perfectly measured, then we would have $a = 0$ and $b = 1$.

Consider the effect of calibration error. Comparing the measured and true angular velocities yields:

$$\hat{\omega} - \omega = a + b\,\omega - \omega = a + \omega(b - 1). \tag{9.2}$$

Now imagine using the sensor to estimate the orientation of the merry-go-round. We would like to understand the difference between the true orientation θ and an estimate $\hat{\theta}$ computed using the sensor output. Let $d(t)$ denote a function of time called the *drift error*:

$$d(t) = \theta(t) - \hat{\theta}(t). \tag{9.3}$$

Suppose it is initially given that $\theta(0) = 0$, and to keep it simple, the angular velocity ω is constant. By integrating (9.2) over time, drift error is

$$d(t) = (\hat{\omega} - \omega)t = (a + b\,\omega - \omega)t = (a + \omega(b - 1))t. \tag{9.4}$$

Of course, the drift error grows as a is larger or b deviates from one; however, note that the second component is proportional to ω. Ignoring a, this means that the drift error is proportional to the speed of the merry-go-round. In terms of tracking a VR headset using a gyroscope, this means that tracking error increases as a faster rate as the head rotates more quickly [81].

At this point, four general problems must be solved to make an effective tracking system, even for this simple case:

1. **Calibration:** If a better sensor is available, then the two can be closely paired so that the outputs of the worse sensor are transformed to behave as close to the better sensor as possible.

2. **Integration:** Every sensor provides measurements at discrete points in time, resulting in a *sampling rate*. The orientation is estimated by aggregating or integrating the measurements.

3. **Registration:** The initial orientation must somehow be determined, either by an additional sensor or a clever default assumption or startup procedure.

4. **Drift correction:** Other sensors directly estimate the drift error. The resulting information is used to gradually eliminate it.

All of these issues remain throughout this chapter for the more complicated settings. The process of combining information from multiple sensor readings is often called *sensor fusion* or *filtering*.

We discuss each of these for the 2D case, before extending the ideas to the 3D case in Section 9.2.

Calibration You could buy a sensor and start using it with the assumption that it is already well calibrated. For cheaper sensor, however, the calibration is often unreliable. Suppose we have an expensive, well-calibrated sensor that reports angular velocities with very little error. Let $\hat{\omega}'$ denote its output, to distinguish it from the forever unknown true value ω. This measurement could be provided by an expensive turntable, calibration rig, or robot.

Calibration involves taking many samples, typically thousands, and comparing $\hat{\omega}'$ to $\hat{\omega}$. A common criterion is the *sum of squares error*, which is given by

$$\sum_{i=1}^{n} (\hat{\omega}_i - \hat{\omega}'_i)^2 \tag{9.5}$$

for n samples of the angular velocity. The task is to determine a transformation to apply to the sensor outputs $\hat{\omega}$ so that it behaves as closely as possible to the expensive sensor which produced $\hat{\omega}'$. Thus, select constants c_1 and c_2 that optimize the error:

$$\sum_{i=1}^{n} (c_1 + c_2\,\hat{\omega} - \hat{\omega}')^2. \tag{9.6}$$

This is a classical regression problem referred to as *linear least-squares*. It is typically solved by calculating the Moore-Penrose pseudoinverse of an non-square matrix that contains the sampled data [155].

Once c_1 and c_2 are calculated, every future sensor reading is transformed as

$$\hat{\omega}_{cal} = c_1 + c_2\,\hat{\omega}, \tag{9.7}$$

in which $\hat{\omega}$ is the original, raw sensor output, and $\hat{\omega}_{cal}$ is the calibrated output. Thus, the calibration produces a kind of invisible wrapper around the original sensor outputs so that the better sensor is simulated. The raw sensor outputs are no longer visible to outside processes. The calibrated outputs will therefore simply be referred to as $\hat{\omega}$ in future discussions.

Integration Sensor outputs usually arrive at a regular sampling rate. For example, the Oculus Rift CV1 gyroscope provides a measurement every 1ms (yielding a 1000Hz sampling rate). Let $\hat{\omega}[k]$ refer to the kth sample, which arrives at time $k\Delta t$.

The orientation $\theta(t)$ at time $t = k\Delta t$ can be estimated by integration as:

$$\hat{\theta}[k] = \theta(0) + \sum_{i=1}^{k} \hat{\omega}[i]\Delta t. \tag{9.8}$$

Each output $\hat{\omega}[i]$ causes a rotation of $\Delta\theta[i] = \hat{\omega}[i]\Delta t$. It is sometimes more convient to write (9.8) in an incremental form, which indicates the update to $\hat{\theta}$ after each new sensor output arrives:

$$\hat{\theta}[k] = \hat{\omega}[k]\Delta t + \hat{\theta}[k-1]. \tag{9.9}$$

For the first case, $\hat{\theta}[0] = \theta(0)$.

If $\omega(t)$ varies substantially between $\theta(k\Delta t)$ and $\theta((k+1)\Delta t)$, then it is helpful to know what $\hat{\omega}[k]$ corresponds to. It could be angular velocity at the start of the interval Δt, the end of the interval, or an average over the interval. If it is the start or end, then a trapezoidal approximation to the integral may yield less error over time [].

Registration In (9.8), the initial orientation $\theta(0)$ was assumed to be known. In practice, this corresponds to a *registration* problem, which is the initial alignment between the real and virtual worlds. To understand the issue, suppose that θ represents the yaw direction for a VR headset. One possibility is to assign $\theta(0) = 0$, which corresponds to whichever direction the headset is facing when the tracking system is turned on. This might be when the system is booted, or if the headset has an "on head" sensor, then it could start when the user attaches the headset to his head. Often times, the forward direction could be unintentionally set in a bad way. For example, if one person starts a VR demo and hands the headset to someone else, who is facing another direction, then in VR the user would not be facing in the intended forward direction. This could be fixed by a simple option that causes "forward" (and hance $\theta(t)$) to be redefined as whichever direction the user is facing at present.

An alternative to this entire problem is to declare $\theta(0) = 0$ to correspond to a direction that is fixed in the physical world. For example, if the user is sitting at a desk in front of a computer monitor, then the forward direction could be defined as the yaw angle for which the user and headset are facing the monitor. Implementing this solution requires a sensor that can measure the yaw orientation with respect to the surrounding physical world. For example, with the Oculus Rift CV1, the user faces a stationary camera, which correponds to the forward direction.

Drift correction To make a useful tracking system, the drift error (9.3) cannot be allowed to accumulate. Even if the sensor were perfectly calibrated, drift error would nevertheless grow due to other factors such as quantized output values, sampling rate limitations, and unmodeled noise. The first problem is to estimate the drift error, which is usually accomplished with an additional sensor. Practical examples of this will be given in Section 9.2. For the simple merry-go-round example, imagine that an overhead camera takes a picture once in a while to measure the orientation. Based on this, let $\hat{d}[k]$ denote the estimated drift error.

The problem is that there are now two conflicting sources of information. A classic way to blend these two sources is via a *complementary filter*. This gives

$$\hat{\theta}_c[k] = \alpha\hat{d}[k] + (1-\alpha)\hat{\theta}[k], \tag{9.10}$$

in which α is a *gain* parameter, with $0 < \alpha < 1$. Above, $\hat{\theta}_c$ denotes the corrected estimate. The filter essentially interpolates between the two sources. Since the gyroscope is usually accrurate over short times but gradually drifts, α is chosen to be close to zero (for example, $\alpha = 0.001$). This causes the drift correction term $\hat{d}[k]$ to be gradually applied. It is important to select the gain α to be high enough so that the drift is correct, but low enough so that the user does not perceive the corrections. The gain could be selected "optimally" by employing a Kalman filter [27, 65, 72]; however, the optimality only holds if we have a linear stochastic system, which is not the case in rigid body tracking. The relationship between Kalman and complementary filters, for the exact models used in this chapter, appears in [60].

9.2 Tracking 3D Orientation

IMUs Recall from Section 2.1 (Figure 2.9) that IMUs have recently gone from large, heavy mechanical systems to cheap, microscopic MEMS circuits. This progression was a key enabler to high-quality orientation tracking. The gyroscope measures angular velocity along three orthogonal axes, to obtain $\hat{\omega}_x$, $\hat{\omega}_y$, and $\hat{\omega}_z$. For each axis, the sensing elements lie in the perpendicular plane, much like the semicircular canals in the vestibular organ (Section 8.2). The sensing elements in each case are micromachined mechanical elements that vibrate and operate like a tuning fork. If the sensor rotates in its direction of sensitivity, the elements experience Coriolis forces, which are converted into electrical signals that are calibrated to produce an output in degrees or radians per second; see Figure 9.1.

IMUs usually contain additional sensors that are useful for detecting drift errors. Most commonly, accelerometers measure linear acceleration along three axes to obtain $\hat{a}_x$, $\hat{a}_y$, and $\hat{a}_z$. The principle of their operation is shown in Figure 9.2. MEMS magnetometers also appear on many modern IMUs, which measure magnetic field strength along the three perpendicular axis. This is often accomplished by the mechanical motion of a MEMS structure that is subject to Lorentz force as it conducts inside of a magnetic field.

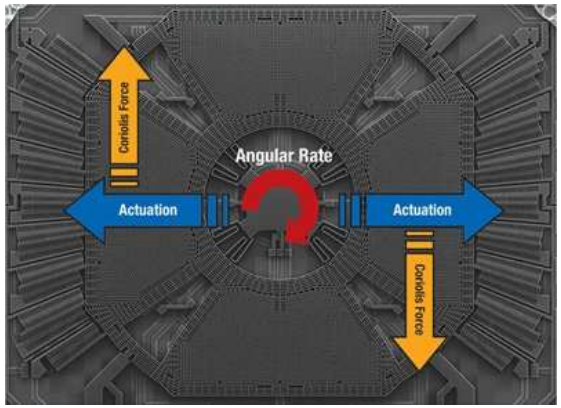


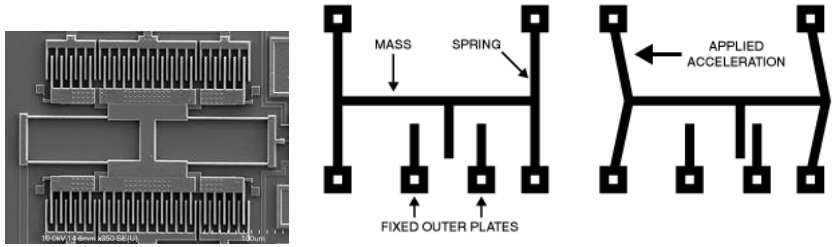
Figure 9.1: The vibrating MEMS elements respond to Coriolis forces during rotation, which are converted into an electrical signal. (Figure by Fabio Pasolini.)

Calibration Recall from Section 9.1 that the sensor outputs are distorted due to calibration issues. In the one-dimensional angular velocity case, there were only two parameters, for scale and offset, which appear in (9.1). In the 3D setting, this would naturally extend to 3 scale and 3 offset parameters; however, the situation is worse because there may also be errors due to non-orthogonality of the MEMS elements. All of these can be accounted for by 12 parameters arranged in a homogeneous transform matrix:

$$\begin{bmatrix} \hat{\omega}_x \\ \hat{\omega}_y \\ \hat{\omega}_z \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c & j \\ d & e & f & k \\ g & h & i & \ell \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \\ 1 \end{bmatrix} \tag{9.11}$$

There are 12 and not 6 DOFs because the upper left, 3-by-3, matrix is not constrained to be a rotation matrix. The j, k, and ℓ parameters correspond to offset, whereas all others handle scale and non-orthogonality. Following the same methodology as in Section 9.1, the inverse of this transform can be estimated by minimizing the least squares error with respect to a better sensor that provides ground truth. The outputs of the MEMS sensor are then adjusted by applying the estimated homogeneous transform to improve performance (this is an extension of (9.7) to the 12-parameter case. This general methodology applies to calibrating gyroscopes and accelerometers. Magnetometers may also be calibrated in this way, but have further complications such as *soft iron bias*.

An additional challenge with MEMS sensors is dealing with other subtle dependencies. For example, the outputs are sensitive to the particular temperature of the MEMS elements. If a headset heats up during use, then calibration parameters are needed for every temperature that might arise in practice. Fortunately, IMUs usually contain a temperature sensor that can be used to associate the calibration



(a) (b)

Figure 9.2: (a) A MEMS element for sensing linear acceleration. (b) Due to linear acceleration in one direction, the plates shift and cause a change in capacitance as measured between the outer plates. (Figure by David Askew.)

parameters with the corresponding temperatures. Finally, MEMS elements may be sensitive to forces acting on the circuit board, which could be changed, for example, by a dangling connector. Care must be given to isolate external board forces from the MEMS circuit.

Integration Now consider the problem converting the sequence of sensor outputs into an estimate of the 3D orientation. At each stage k a vector

$$\hat{\omega}[k] = (\hat{\omega}_x[k], \hat{\omega}_y[k], \hat{\omega}_z[k]) \tag{9.12}$$

arrives from the sensor. In Section 9.1, the sensor output $\hat{\omega}[k]$ was converted to a change $\Delta\theta[k]$ in orientation. For the 3D case, the change in orientation is expressed as a *quaternion*.

Let $q(v, \theta)$ be the quaternion obtained by the axis-angle conversion formula (3.29). Recall from Section 8.1.2 that the instantaneous axis of rotation is the magnitude of the angular velocity. Thus, if $\hat{\omega}[k]$ is the sensor output at stage k, then the estimated rotation axis is

$$\hat{v}[k] = \hat{\omega}[k] / \|\hat{\omega}[k]\|. \tag{9.13}$$

Furthermore, the estimated amount of rotation that occurs during time Δt is

$$\Delta\hat{\theta}[k] = \|\hat{\omega}[k]\| \Delta t. \tag{9.14}$$

Using the estimated rotation axis (9.13) and amount (9.14), the orientation change over time Δt is estimated to be

$$\Delta\hat{q}[k] = q(\hat{v}[k], \Delta\hat{\theta}[k]). \tag{9.15}$$

Using (9.15) at each stage, the estimated orientation $\hat{q}[k]$ after obtaining the latest sensor output is calculated incrementally from $\hat{q}[k-1]$ as

$$\hat{q}[k] = \Delta\hat{q}[k] * \hat{q}[k-1], \tag{9.16}$$

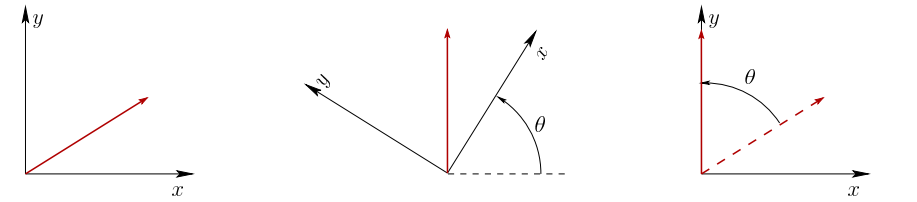


Figure 9.3: If "up" is perfectly sensed by a sensor that is rotated by θ, then its output needs to be rotated by θ to view it from the world frame.

in which $*$ denotes quaternion multiplication. This is the 3D generalization of (9.9), at which point simple addition could be used to combine rotations in the 2D case. In (9.16), quaternion multiplication is needed to aggregate the change in orientation (simple addition is commutative, which is inappropriate for 3D rotations).

Registration The registration problem for the yaw component is the same as in Section 9.2. The forward direction may be chosen from the initial orientation of the rigid body or it could be determined with respect to a fixed direction in the world. The pitch and roll components should be determined so that they align with gravity. The virtual world should not appear to be tilted with respect to the real world (unless that is the desired effect, which is rarely the case).

Tilt correction The drift error $d(t)$ in (9.3) was a single angle, which could be positive or negative. If added to the estimate $\hat{\theta}(t)$, the true orientation $\theta(t)$ would be obtained. It is similar for the 3D case, but with quaternion algebra. The 3D drift error is expressed as

$$d(t) = q(t) * \hat{q}^{-1}(t), \tag{9.17}$$

which is equal to the identity rotation if $q(t) = \hat{q}(t)$. Furthermore, note that applying the drift error to the estimate yields $q(t) = d(t) * \hat{q}(t)$.

Since the drift error is 3D rotation, it could be constructed as the product of a yaw, pitch, and a roll. Let *tilt error* refer to the part of the drift error that corresponds to pitch and roll. This will be detected using an "up" sensor. Let *yaw error* refer to the remaining part of the drift error, which will be detecting using a "compass". In reality, there do not exist perfect "up" and "compass" sensors, which will be addressed later.

Suppose that a sensor attached to the rigid body always reports an "up" vector that is parallel to y axis in the fixed, world coordinate frame. In other words, it would be parallel to gravity. Since the sensor is mounted to the body, it reports its values in the coordinate frame of the body. For example, if the body were rolled 90 degrees so that its x axis is pointing straight up, then the "up" vector would be reported as $(0, 0, 1)$, instead of $(0, 1, 0)$. To fix this, it would be convenient to

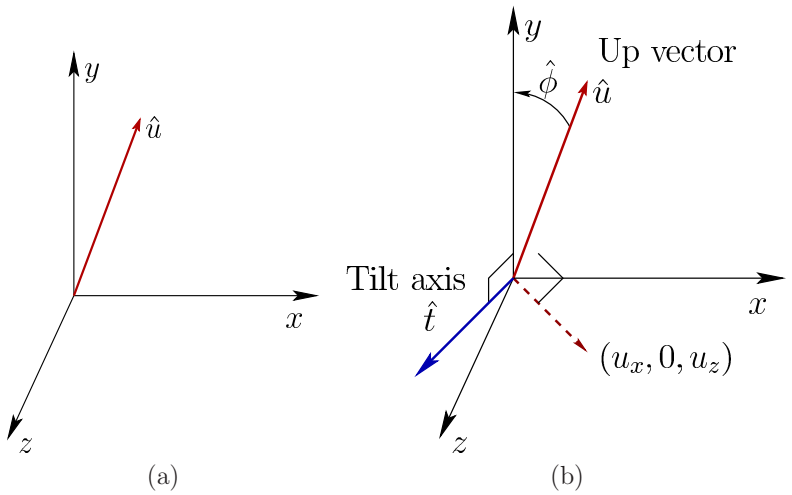


Figure 9.4: (a) Tilt error causes a discrepancy between the y axis and the sensed up vector that is rotated using the estimate $\hat{q}[k]$ to obtain $\hat{u}$. (b) The *tilt axis* is normal to $\hat{u}$; a rotation of $-\phi$ about the tilt axis would bring them into alignment, thereby eliminating the tilt error.

transform the sensor output into the world frame. This involves rotating it by q, the body orientation. For our example, this roll rotation would transform $(0, 0, 1)$ into $(0, 1, 0)$. Figure 9.3 shows a 2D example of this problem.

Now suppose that drift error has occured and that $\hat{q}[k]$ is the estimated orientation. If this transform is applied to the "up" vector, then due to drift error, it might not be aligned with the y axis, as shown Figure 9.4. The up vector $\hat{u}$ is projected into the xz plane to obtain $(\hat{u}_x, 0, \hat{u}_z)$. The *tilt axis* lies in the xz plane and is constructed as the normal to the projected up vector: $\hat{t} = (\hat{u}_z, 0, -\hat{u}_x)$. Performing a rotation of ϕ about the axis $\hat{t}$ would move the up vector into alignment with the y axis. Thus, the tilt error portion of the drift error is the quaternion $q(\hat{t}, \phi)$.

Unfortunately, there is no "up" sensor. In practice, the accelerometer is used to measure the "up" direction because gravity acts on the sensor, causing the sensation of upward acceleration at roughly $9.8\text{m}/s^2$. The problem is that it also responds to true linear acceleration of the rigid body, and this cannot be separated from gravity due to the Einstein equivalence principle. It measures the vector sum of gravity and true linear acceleration, as shown in Figure 9.5. A simple heuristic is to trust accelometer outputs as an estimate of the "up" direction only if its magnitude is close to 9.8m^2 [39]. This could correspond to the common case in which the rigid body is stationary. However, this assumption is unreliable because downward and lateral linear accelerations can be combined to provide an output

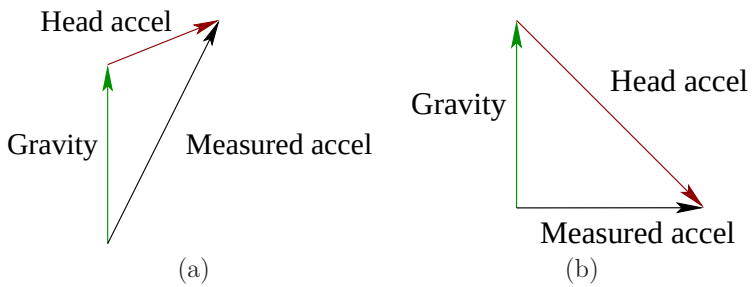


Figure 9.5: (a) There is no gravity sensor; the accelerometer measures the vector sum of apparent acceleration due to gravity and the true acceleration of the body. (b) A simple heuristic of accepting the reading as gravity only if the magnitude is appromimately 9.8m^2 will fail in some cases.

that is close to 9.8m^2, but with a direction that is far from gravity. Better heuristics may be built from simultaneously considering the outputs of other sensors or the rate at which "up" appears to change.

Assuming the accelerometer is producing a reliable estimate of the gravity direction, the up vector $\hat{u}$ is calculated from the accelerometer output $\hat{a}$ by using (3.33), to obtain

$$\hat{u} = \hat{q}[k] * \hat{a} * \hat{q}^{-1}[k]^{-1}. \tag{9.18}$$

Yaw correction The remaining drift error component is detected by a "compass", which outputs a vector that lies in the world xz plane and always points "north". Suppose this is $\hat{n} = (0, 0, -1)$. Once again, the sensor output occurs in the coordinate frame of the body, and needs to be transformed by $\hat{q}[k]$. The difference between $\hat{n}$ and the $-z$ axis is the resulting yaw drift error.

As in the case of the "up" sensor, there is no "compass" in the real world. Instead, there is a magnetometer, which measures a 3D magnetic field vector: $(\hat{m}_x, \hat{m}_y, \hat{m}_z)$. Suppose this is used to measure the Earth's magnetic field. It turns out that the field vectors do not "point" to the North pole. The Earth's magnetic field produces 3D vectors that generally do not lie in the horizontal plane, resulting in an *inclination angle*. Thus, the first problem is that the sensor output must be projected into the xz plane. Residents of Ecuador may enjoy magnetic field vectors that are nearly horizontal; however, in Finland they are closer to vertical; see Figure 9.6. If the magnetic field vector is close to vertical, then the horizontal component may become too small to be useful.

Another issue is that the projected vector in the horizontal plane does not point north, resulting in a *declination angle*; this is the deviation from north. Fortuntely, reference to the true north is not important. It only matters that the sensor output is recorded in the registration stage to provide a fixed yaw reference.

The most significant problem is that the magnetometer measures the vector

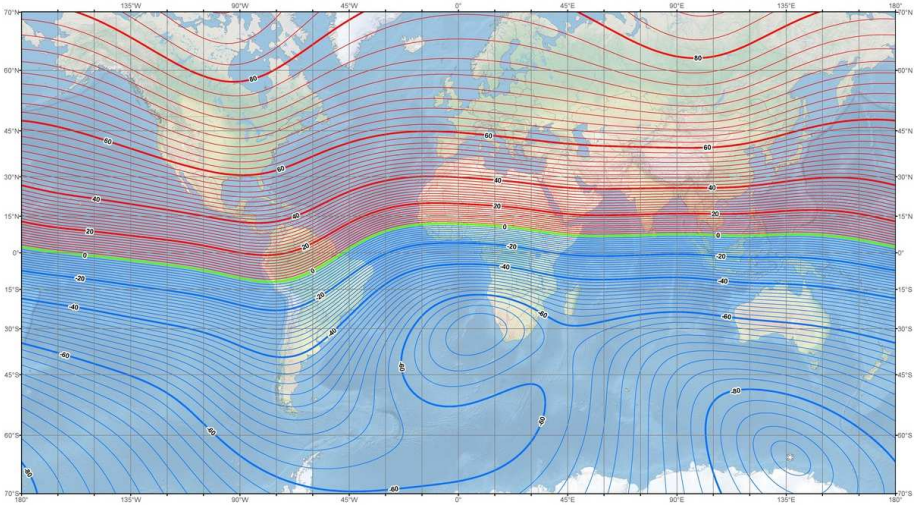


Figure 9.6: The inclination angle of the Earth's magnetic field vector varies greatly over the Earth. (Map developed by NOAA/NGDC and CIRES.)

sum of *all* magnetic field sources. In addition to the Earth's field, a building generates its own field due to ferromagnetic metals. Furthermore, such materials usually exist on the circuit board that contains the sensor. For this case, the field moves with the sensor, generating a constant vector offset. Materials that serve as a source of magnetic fields are called *hard iron*. Other materials distort magnetic fields that pass through them; these are called *soft iron*. Magnetometer calibration methods mainly take into account offsets due to hard-iron bias and eccentricities due to soft-iron bias [47, 71].

After all calibrations have been performed, the yaw drift error can be estimated from most locations with a few degrees of accuracy, which is sufficient to keep yaw errors from gradually accumulating. There are still problems. If a strong field is placed near the sensor, then it will be non-constant. This could cause the measured direction to change as the rigid body translates back and forth. Another problem is that in some building locations, vector sum of the Earth's magnetic field and the field generated by the building could by approximately zero (they are of similar magnitude and pointing in opposite directions). In this unfortunate case, the magnetometer cannot produce useful outputs for yaw drift error detection.

Filtering Using the detected drift error, filtering works in the same way as described in Section 9.1. The complementay filter (9.10) is upgraded to work with quaternions:

$$\hat{q}_c[k] = \alpha\hat{d}[k] * (1 - \alpha)\hat{q}[k]. \tag{9.19}$$

Yaw	Pitch	Roll	Error
+	+	+	None
-	+	+	L/R mix, flipped x
+	-	+	L/R mix, flipped y
+	+	-	L/R mix, flipped z
+	-	-	Inverse and L/R mix, flipped x
-	+	-	Inverse and L/R mix, flipped y
-	-	+	Inverse and L/R mix, flipped z
-	-	-	Inverse

Figure 9.7: A table to help debug common viewpoint transform errors. Each + means that the virtual world appears to move the correct way when performing the yaw, pitch, or roll. Each − means it moves in the opposite way. The first case is correct. All others are bugs. "L/R mix" means that left and right-handed coordinate systems are getting mixed; the axis that was flipped is indicated.

The estimated drift error $\hat{d}[k]$ is obtained by multiplying the estimated tilt and yaw errors. Alternatively, they could contribute separately, with different gains for each, and even combined with drift error estimates from more sources [88].

Setting the viewpoint The viewpoint is set using the estimated orientation $\hat{q}[k]$, although it might be adjusted to account for alternative timings, for the purpose of prediction or image warping, as discussed in Section 7.4. Let $\hat{q}(t)$ denote the estimated orientation for time t. In terms of Section 3.4, we have just estimated R_{eye}. To calculate the correct viewpoint, the inverse is needed. Thus, $\hat{q}^{-1}(t)$ would correctly transform models to take the estimated viewpoint into account.

A debugging tip Programmers often make mistakes when connecting the tracked orientation to the viewpoint. Figure 9.7 shows a table of the common mistakes. To determine whether the transform has been correctly applied, one should put on the headset and try rotating about the three canonical axes: A pure yaw, a pure pitch, and a pure roll. Let + denote that the world is moving corrected with respect to a head rotation. Let − denote that it seems to move in the opposite direction. Figure 9.7 shows a table of the eight possible outcomes and the most likely cause of each problem.

A head model The translation part of the head motion has not been addressed. Ideally, the head should be the same height in the virtual world as in the real world. This can be handled by the translation part of the T_{eye} matrix (3.35).

We must also account for the fact that as the head rotates, the eyes change their positions. For example, in a yaw head movement (nodding "no"), the pupils displace a few centimeters in the x direction. More accurately, they travel along

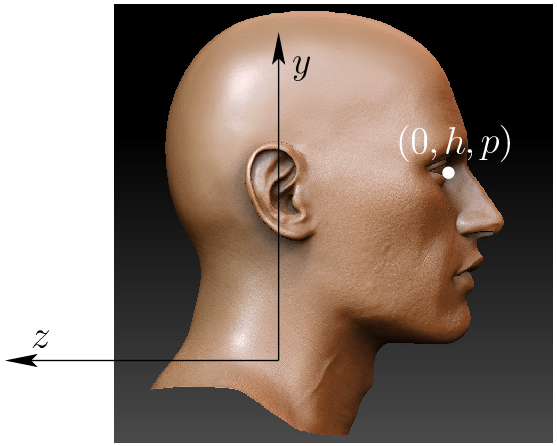


Figure 9.8: To obtain a *head model*, the rotation center is moved so that orientation changes induce a plausible translation of the eyes. The height h is along the y axis, and the protrusion p is along the z axis (which leads a negative number).

a circular arc in a horizontal plane. To more closely mimic the real world, the movements of the eyes through space can be simulated by changing the center of rotation according to a fictitious *head model*. This trick is needed until Section 9.3, where position is instead estimated from sensors.

Recall from Section 3.5 that the cyclopean viewpoint was first considered and then modified to handle left and right eyes by applying horizontal offsets by inserting T_{left} (3.49) and T_{right} (3.51). In a similar way, offsets in the y and z directions can be added to account for displacement that would come from a rotating head. The result is to insert the following before or after T_{right} and T_{left}:

$$T_{head} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & h \\ 0 & 0 & 1 & p \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{9.20}$$

in which h is a height parameter and p is a protrusion parameter. The idea is to choose h and p that would correspond to the center of rotation of the head. The parameter h is the distance from the rotation center to the eye height, along the y axis. A typical value is $h = 0.15$m. The protrusion p is the distance from the rotation center to the cyclopean eye. A typical value is $p = -0.10$m, which is negative because it extends opposite to the z axis. Using a fake head model approximates the eye locations as the user rotates her head; however, it is far from perfect. If the torso moves, then this model completely breaks, resulting in a large mismatch between the real and virtual world head motions. Nevertheless, this head model is currently used in popular headsets, such as Samsung Gear VR.

An issue also exists with the y height of the head center. They user may be seated in the real world, but standing in the virtual world. This mismatch might be uncomfortable. The brain knows that the body is seated because of proprioception, regardless of the visual stimuli provided by VR. If the user is standing, then the head-center height could be set so that the eyes are at the same height as in the real world. This issue even exists for the case of full six-DOF tracking, which is covered next; the user might be sitting, and a vertical offset is added to make him appear to be standing in VR.

9.3 Tracking Position and Orientation

This section covers tracking of all 6 DOFs for a moving rigid body, with the most important case being head tracking. For convenience, we will refer to the position and orientation of a body as its *pose*. Six-DOF tracking enables T_{eye} from 3.4 to be fully derived from sensor data, rather than inventing positions from plausible head models, as in (9.20). By estimating the position, the powerful depth cue of parallax becomes much stronger as the user moves his head from side to side. He could even approach a small object and look at it from any viewpoint, such as from above, below, or the sides. The methods in this section are also useful for tracking hands in space or objects that are manipulated during a VR experience.

Why not just integrate the accelerometer? It seems natural to try to accomplish 6-DOF tracking with an IMU alone. Recall from Figure 9.5 that the accelerometer measures the vector sum of true linear acceleration and acceleration due to gravity. If the gravity component is subtracted away from the output, as is heuristically accomplished for tilt correction, then it seems that the residual part is pure body acceleration. Why not simply integrate this acceleration twice to obtain position estimates? The trouble is that the drift error rate is much larger than in the case of a gyroscope. A simple calibration error leads to linear drift in the gyroscope case because it is the result of a single integration. After a double integration, a calibration error leads to quadratic drift error. This becomes unbearable in practice after a fraction of a second. Furthermore, the true body acceleration cannot be accurately extracted, especially when the body quickly rotates. Finally, as drift accumulates, what sensors can be used to detect the positional drift error? The IMU alone cannot help. Note that it cannot even distinguish motions at constant velocity, including zero motion; this is the same as our vestibular organs. Despite its shortcomings, modern IMUs remain an important part of 6-DOF tracking systems because of their high sampling rates and ability to accurately handle the rotational component.

Make your own waves The IMU approach was *passive* in the sense that it relied on sources of information that already exist in the environment. Instead, an *active* approach can be taken by transmitting waves into the environment. Since

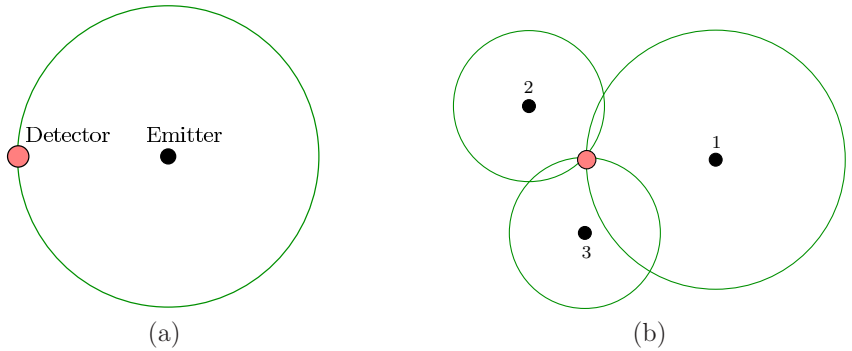


Figure 9.9: The principle of trilateration enables the detector location to be determined from estimates of distances to known emitter. A 2D example is shown: (a) from a single emitter, the detector could be anywhere along a circle; (b) using three emitters, the position is uniquely determined.

humans operate in the same environment, waves that are perceptible, such as light and sound, are not preferred. Instead, common energy sources in active tracking systems include infrared, ultrasound, and electromagnetic fields.

Consider transmitting an ultrasound pulse (above 20,000 Hz) from a speaker and using a microphone to listen for its arrival. This is an example of an *emitter-detector pair*. The speaker is the emitter, and the microphone is the detector. If time measurement is synchronized between source and destination, then the *time of arrival* (*TOA* or *time of flight*) can be calculated. This is the time that it took for the pulse to travel the distance d between the emitter and detector. Based on the known propagation speed in the medium (330 m/s for ultrasound), the distance $\hat{d}$ is estimated. One frustrating limitation of ultrasound systems is reverberation between surfaces, causing the pulse to be received multiple times at each detector.

When functioning correctly, the position of the detector could then be narrowed down to a sphere of radius $\hat{d}$, centered at the transmitter; see Figure 9.9(a). By using two transmitters and one microphone, the position is narrowed down to the intersection of two spheres, resulting in a circle (assuming the transmitter locations are known). With three transmitters, the position is narrowed down to two points, and with four or more transmitters, the position is uniquely determined.[1] The emitter and detector roles could easily be reversed so that the object being tracked carries the emitter, and several receivers are placed around it. The method of combining these measurements to determine position is called *trilateration*. If electromagnetic waves, such as radio, light, or infrared, are used instead of ultrasound, trilateration could still be applied even though the impossible to measure

[1] Global positioning systems (*GPS*) work in this way, but using radio signals, the Earth surface constraint, and at least one more satellite eliminate time synchronization errors.

(a) (b)

Figure 9.10: (a) A magnetic dipole offers a field that varies its magnitude and direction as the position changes. (b) The Razer Hydra, a game controller that generated a weak magnetic field using a base station, enabling it to track its position.

the propagation time directly. If the transmitter amplitude is known then distance can be estimated based on power degradation, rather than TOA. Alternatively, a time-varying signal can be emitted and its reflected phase shift can be estimated when the received signal is superimposed onto the transmitted signal.

If the detectors do not know the precise time that the pulse started, then they could compare differences in arrival times between themselves; this is called *time difference of arrival* (*TDOA*). The set of possible locations is a hyperboloid instead of a sphere. Nevertheless, the hyperboloid sheets can be intersected for multiple emitter-detector pairs to obtain the method of *multilateration*. This was used in the Decca Navigation System in World War II to locate ships and aircraft. This principle is also used by our ears to localize the source of sounds, which will be covered in Section **??**.

Finally, some methods could track position by emitting a complicated field that varies over the tracking area. For example, by creating a magnetic dipole, perhaps coded with a signal to distinguish it from background fields, the position and orientation of a body in the field could be distinguished in the field; see Figure 9.10(a). This principle was used for video games in the Razer Hydra tracking system in a base station that generated a magnetic field; see Figure 9.10(b). One drawback is that the field may become unpredictably warped in each environment, causing straight-line motions to be estimated as curved. Note that the requirements are the opposite of what was needed to use a magnetometer for yaw correction in Section 9.2; in that setting the field needed to be constant over the tracking area. For estimating position, the field should vary greatly in different locations.

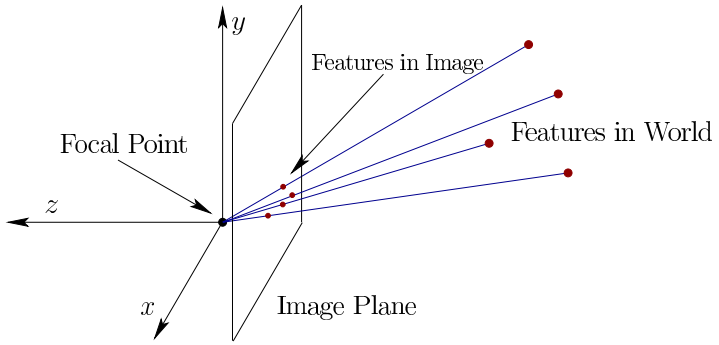


Figure 9.11: The real world contains special *features*, which are determined to lie along a line segment via perspective projection.

The power of visibility The most powerful paradigm for 6-DOF tracking is *visibility*. The idea is to identify special parts of the physical world called *features* and calculate their position along a line-of-sight ray to a known location. Figure 9.11 shows an example inspired by a camera, but other methods could be used. One crucial aspect for tracking is *distinguishability*. If all features appear the same, then it may become difficult to determine and maintain "which is which" during the tracking process. Each feature should be assigned a unique label that is invariant over time, as rigid bodies in the world move. Confusing features with each other could cause catastrophically bad estimates to be made regarding the body pose.

The most common sensor used to detect features is a digital camera. Detecting, labeling, and tracking features are common tasks in computer vision or image processing. There are two options for features:

1. **Natural:** The features are automatically discovered, assigned labels, and maintained during the tracking process.

2. **Artificial:** The features are engineered and placed into the environment so that they can be easily detected, matched to preassigned labels, and tracked.

Natural features are advantageous because there are no setup costs. The environment does not need to be engineered. Unfortunately, they are also much more unreliable. Using a camera, this is considered to be a hard computer vision problem because it may be as challenging as it is for the human visual system. For some objects, textures, and lighting conditions, it could work well, but it is extremely hard to make it work reliably for *all* possible settings. Therefore, artificial features are much more common.

For artificial features, one of the simplest solutions is to print a special tag onto the object to be tracked. For example, we could print bright red dots onto the object and easily scan for their appearance as red blobs in the image. To solve

Figure 9.12: A sample QR code, which could be printed and used as an artificial feature. (Picture from Wikipedia.)

Figure 9.13: The Oculus Rift DK2 headset contained IR LEDs hidden behind IR-transparent plastic. (Photo by Paul Dempsey.)

the distinguishability problem, we might have to use multiple colors, such as red, green, blue, and yellow dots. Trouble may occur if these colors exist naturally in other parts of the image. A more reliable method is to design a specific *tag* that is clearly distinct from the rest of the image. Such tags can be coded to contain large amounts of information, including a unique identification number. One of the most common coded tags is the *QR code*, an example of which is shown in Figure 9.12.

The features described so far are called *passive* because they do not emit energy. The hope is that sufficient light is in the world so that enough reflects off of the feature and enters the camera sensor. A more reliable alternative is to engineer *active* features that emit their own light. For example, colored LEDs can be mounted on the surface of a headset or controller. This comes at the expense of requiring a power source and increasing overall object cost and weight. Furthermore, its industrial design may be compromised because it would be lit up like a Christmas tree.

Fortunately, all of these tricks can be moved to the infrared (IR) part of the spectrum so that features are visible to cameras, but not to humans. Patterns can be painted onto objects that highly reflect IR energy. Alternatively, IR LEDs can be mounted onto devices. This is the case for Oculus Rift DK2 and later models, and the IR LEDs are even hidden behind plastic that is transparent for IR energy, but appears black to humans; see Figure 9.13.

In some settings, it might be difficult to mount LEDs on the objects, as in the case of tracking the subtle motions of an entire human body. This is called *MOCAP* or *m*otion capture, which is described in Section 9.4. In MOCAP systems, powerful IR LEDs are positioned around the camera so that they illuminate *retroreflective markers* that are placed in the scene. Each marker can be imagined as a spherical mirror in the IR part of the spectrum. One unfortunate drawback is that the range is limited because IR energy must travel from the camera location to the target and back again. Since energy dissipates quadratically as function of distance, doubling the distance results on one-fourth of the energy level arriving at the camera.

At this point, it is natural to wonder why an entire image is being captured if the resulting image processing problem is trivial. The main reason is the proliferation of low-cost digital cameras and image processing software. Why not simply design an emitter-detector pair that produces a binary reading, indicating whether the visibility beam is occluded? This is precisely how the detection beam works in an automatic garage door system to ensure the door does not close on someone: An IR LED emits energy to a detection *photodiode*, which is essentially a switch that activate when it receives a sufficient level of energy for its target wavelength (in this case IR). To reduce the amount of energy dissipation, mirrors or lenses could be used to focus the energy.

Even better, an IR laser can be aimed directly at the detector. The next task is to use lenses and moving mirrors so that every detector that is visible from a fixed location will become illuminated at some point. The beam can be spread from a dot to a line using a lens, and then the line is moved through space using a spinning mirror. This is the basis of the *lighthouse tracking* system for the HTC Vive VR headset, which is covered later in this section.

The Perspective-n-Point (PnP) problem A moving rigid body needs to be "pinned down" using n observed features. This is called the *Perspective-n-Point* (or *PnP*) problem. We can borrow much of the math from Chapter 3; however, here we consider the placement of bodies in the *real* world, rather than the virtual world. Furthermore, we have an *inverse problem*, which is to determine the body placement based on points in the image. Up until now, the opposite problem was considered. For visual rendering in Chapter 7, an image was produced based on the known body placement in the (virtual) world.

The features could be placed on the body or in the surrounding world, depending on the sensing method. Suppose for now that they are on the body. Each feature corresponds to a point $p = (x, y, z)$ with coordinates defined in the frame of the body. Let T_{rb} be a homogeneous transformation matrix that contains pose

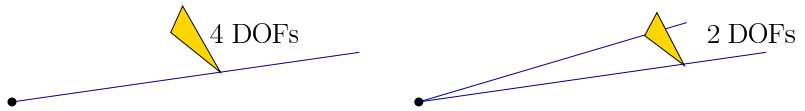


Figure 9.14: Each feature that is visible eliminates 2 DOFs. On the left, a single feature is visible, and the resulting rigid body has only 4 DOFs remaining. On the right, two features are visible, resulting in only 2 DOFs. This can be visualized as follows. The edge that touches both segments can be moved back and forth while preserving its length if some rotation is also applied. Rotation about an axis common to the edge provides the second DOF.

parameters, which are assumed to be unknown. Applying the transform T_{rb} to the point p_i as in (3.22) could place it anywhere in the real would. Recall the chain of transformations (3.40), which furthermore determines where each point on the body would appear in an image. The matrix T_{eye} held the camera pose, whereas T_{vp} and T_{can} contained the perspective projection and transformed the projected point into image coordinates.

Now suppose that a feature has been observed to be at location (i, j) in image coordinates. If T_{rb} is unknown, but all other transforms are given, then there would be six independent parameters to estimate, corresponding to the 6 DOFs. Observing (i, j) provides two independent constraints to the chain of transforms (3.40), one i and one for j. The rigid body therefore loses 2 DOFs, as shown in Figure 9.14. This was the P1P problem because n, the number of features, is one.

The P2P problem corresponds to observing two features in the image and results in four constraints. In this case, each constraint eliminates two DOFs, resulting in only two remaining DOFs; see Figure 9.14. Continuing further, if three features are observed, then for the P3P problem, zero DOFs remain (except for the case in which collinear features are chosen on the body). It may seem that the problem is completely solved; however, zero DOFs allows for isolated point solutions. The P3P problem corresponds to trying to place a given triangle into a pyramid formed by rays so that each triangle vertex touches a different ray. This can be generally accomplished in four ways, which are hard to visualize. Imagine trying to slice a tall, thin pyramid (simplex) made of cheese so that four different slices have the exact same triangular size and shape. The cases of P4P and P5P also result in ambiguous solutions. Finally, in the case of P6P, unique solutions are always obtained if no four features are coplanar. All of the mathematical details are worked out in [161].

The PnP problem has been described in the ideal case of having perfect coordinate assignments to the feature points on the body and the perfect observation of those through the imaging process. In practice, small errors are made due to factors such as sensor noise, image quantization, and manufacturing tolerances. This results in ambiguities and errors in the estimated pose, which could deviate substantially from the correct answer [128]. Therefore, many more features may

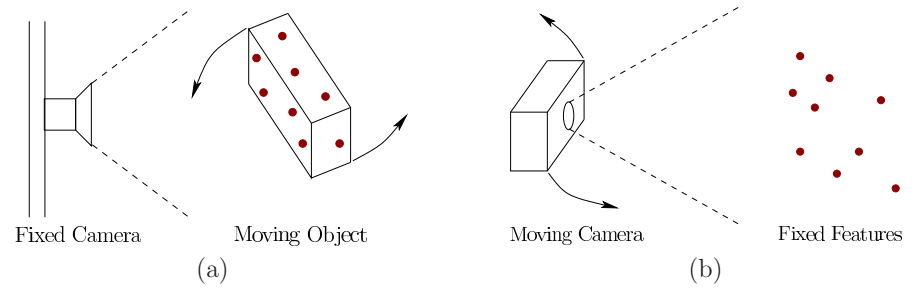


Figure 9.15: Two cases for camera placement: (a) A *world-fixed camera* is stationary, and the motions of objects relative to it are estimated using features on the objects. (b) An *object-fixed camera* is frequently under motion and features are ideally fixed to the world coordinate frame.

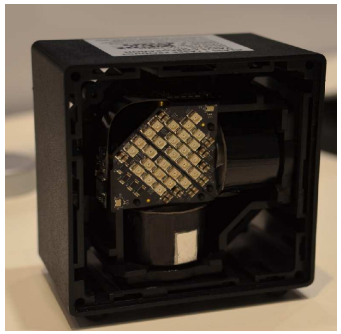
be used in practice to improve accuracy. Furthermore, a calibration procedure, such as *bundle adjustment* [56], may be applied before the device is used so that the feature point locations can be more accurately assigned before pose estimation is performed.

Camera-based implementation The visibility problem may be solved using a camera in two general ways, as indicated in Figure 9.15. Consider the *camera frame*, which is analogous to the eye frame from Figure 3.14 in Chapter 3. A *world-fixed camera* is usually stationary, meaning that the camera frame does not move relative to the world. A single transformation may be used to convert an object pose as estimated from the camera frame into a convenient world frame. For example, in the case of the Oculus Rift CV1, the head pose could be converted to a world frame in which the $-z$ direction is pointing at the camera, y is "up", and the position is in the center of the camera's tracking region or a suitable default based on the user's initial head position. For an *object-fixed camera*, the estimated pose, derived from features that remain fixed in the world, is the transformation from the camera frame to the world frame. This case would be obtained, for example, if we placed QR codes on the walls.

As in the case of an IMU, calibration is important for improving sensing accuracy. The following homogeneous transformation matrix can be applied to the image produced by a camera:

$$\begin{bmatrix} \alpha_x & \gamma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9.21}$$

The five variables appearing in the matrix are called *intrinsic parameters* of the camera. The α_x and α_y parameters handle scaling, γ handles shearing, and u_0 and v_0 handle offset of the optical axis. These parameters are typically estimated



(a)

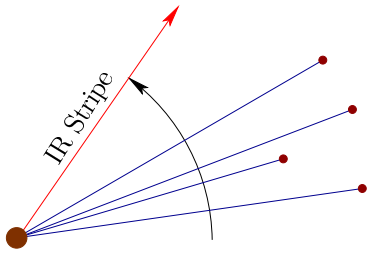
(b)

Figure 9.16: The laser-based tracking approach used in the HTC Vive VR headset: (a) A base station contains spinning drums that emit horizontal and vertical sheets of IR light. An array of IR LEDs appears in the upper left, which provide a synchronization flash. (b) Photodiodes in pockets on the front of the headset detect the incident IR light.

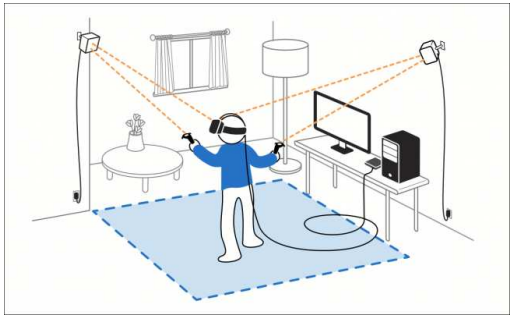
by taking images of an object for which all dimensions and distances have been carefully measured, and performing least-squares estimation to select the parameters that reduce the sum-of-squares error (as described in Section 9.1). For a wide-angle lens, further calibration may be needed to overcome optical distortions (recall Section 7.3).

Now suppose that a feature as been observed in the image, perhaps using some form of *blob detection* to extract the pixels its image covers from the rest of the image []. This is easiest for a global shutter camera because all pixels will correspond to the same instant of time. In the case of a rolling shutter, the image may need to be transformed to undo the effects of motion (recall Figure 4.33). The location of the observed feature is calculated as a statistic of the blob pixel locations. Most commonly, the average over all blob pixels is used, resulting in non-integer image coordinates. Many issues affect performance: 1) quantization errors arise due to image coordinates for each blob pixel being integers; 2) if the feature does not cover enough pixels, then the quantization errors are worse; 3) changing in lighting conditions may make it difficult to extract the feature, especially in the case of natural features; 4) at some angles, two or more features may become close in the image, making it difficult to separate their corresponding blobs; 5) as various features enter or leave the camera view, the resulting estimated pose may jump.

Laser-based implementation By designing a special emitter-detector pair, the visibility problem can be accurately solved over great distances. This was accomplished by the *lighthouse tracking* system of the 2016 HTC Vive headset,



(a)



(b)

Figure 9.17: (a) This is a 2D view of the angular sweep of the IR stripe in the laser-based tracking approach (as in HTC Vive). This could correspond to a top-down view, in which a vertical stripe spins with a yaw rotation about the base. In this case, the angular locations in the horizontal direction are observed, similar to column coordinates of a camera image. This could also correspond to a side view, in which case the vertical stripe spins with a pitch rotation and the angular locations in the vertical direction. As the beam hits the features, which are photodiodes, the direction is known because of the spinning rate and time since the synchronization flash. (b) By putting two case stations on top of poles at the corners of the tracking area, a large region can be accurately tracked for a headset and controllers. (Drawing by Chris Stobing.)

and the *Minnesota scanner* from 1989 [138]. Figure 9.16 shows the lighthouse tracking hardware for the HTC Vive. The operation of a camera is effectively simulated, as shown in Figure 9.17(a).

If the base station were a camera, then the sweeping vertical stripe would correspond to estimating the row of the pixel that corresponds to the feature; see Figure 9.17(a). Likewise, the sweeping horizontal stripe corresponds to the pixel column. The rotation rate of the spinning drum is known and is analogous to the camera frame rate. The precise timing is recorded as the beam hits each photodiode.

Think about polar coordinates (distance and angle) relative to the base station. Using the angular velocity of the sweep and the relative timing differences, the angle between the features as "observed" from the base station can be easily estimated. Although the angle between features is easily determined, their angles relative to some fixed direction from the base station must be determined. This is accomplished by an array of IR LEDs that are pulsed on simultaneously so that all photodiodes detect the flash (visible in Figure 9.17(a)). This could correspond, for example, to the instant of time at which each beam is at the 0 orientation. Based on the time from the flash until the beam hits a photodiode, and the known angular velocity, the angle of the observed feature is determined. To reduce temporal drift

error, the flash may be periodically used during operation.

As in the case of the camera, the distances from the base station to the features are not known, but can be determined by solving the PnP problem. Multiple base stations can be used as well, in a way that is comparable to using multiple cameras or multiple eyes to infer depth. The result is accurate tracking over a large area, as shown in Figure 9.17(b).

Filtering As in Section 9.2, outputs from sensors are combined over time by a filtering method to maintain the estimate. In the current setting, the pose can be maintained by combining both visibility information and outputs of an IMU. For the orientation component of the pose, the complementary filter from (9.10) could be used. The camera provides an additional source for detecting orientation drift error. The camera optical axis is a straightforward reference for yaw error estimation detection, which makes it a clear replacement for the magnetometer. If the camera tilt is known, then the camera can also provide accurate tilt error estimation.

The IMU was crucial for obtaining highly accurate orientation tracking because of accurate, high-frequency estimates of angular velocity provided by the gyroscope. If the frame rate for a camera or lighthouse system is very high, then sufficient sensor data may exist for accurate position tracking; however, it is preferable to directly measure derivatives. Unfortunately, IMUs do not measure linear velocity. However, the output of the linear accelerometer could be used as suggested in the beginning of this section. Suppose that the accelerometer estimates the body acceleration as

$$\hat{a}[k] = (\hat{a}_x[k], \hat{a}_y[k], \hat{a}_z[k]) \tag{9.22}$$

in the world frame (this assumes the gravity component has been subtracted from the accelerometer output).

By numerical integration, the velocity $\hat{v}[k]$ can be estimated from $\hat{a}[k]$. The position $\hat{p}[k]$ is estimated by integrating the velocity estimate. The update equations using simple Euler integration are

$$\begin{aligned} \hat{v}[k] &= \hat{a}[k]\Delta t + \hat{v}[k-1] \\ \hat{p}[k] &= \hat{v}[k]\Delta t + \hat{p}[k-1]. \end{aligned} \tag{9.23}$$

Note that each equation actually handles three components, x, y, and z, at the same time. The accuracy of the second equation can be further improved by adding $\frac{1}{2}\hat{a}[k]\Delta t^2$ to the right side.

As stated earlier, double integration of the acceleration leads to rapidly growing position drift error, denoted by $\hat{d}_p[k]$. The error detected from PnP solutions provide an estimate of $\hat{d}_p[k]$, but perhaps at a much lower rate than the IMU produces observations. For example, a camera might take pictures at 60 FPS and the IMU might report accelerations at 1000 FPS.

The complementary filter from (9.10) can be extended to the case of double integration to obtain

$$\begin{aligned} p_c[k] &= \alpha_p \hat{d}_p[k] + (1-\alpha_p)\hat{p}[k] \\ v_c[k] &= \alpha_v \hat{d}_p[k] + (1-\alpha_v)\hat{v}[k]. \end{aligned} \tag{9.24}$$

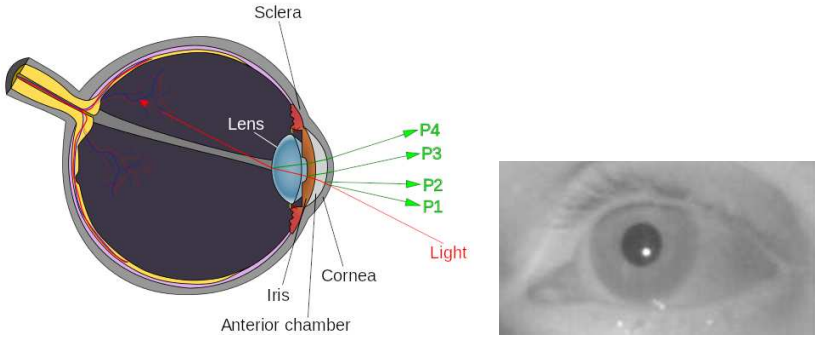
Above, $p_c[k]$ and $v_c[k]$ are the corrected position and velocity, respectively, which are each calculated by a complementary filter. The estimates $\hat{p}[k]$ and $\hat{v}[k]$ are calculated using (9.23). The parameters α_p and α_v control the amount of importance given to the drift error estimate in comparison to IMU updates.

Equation (9.24) is actually equivalent to a *Kalman filter*, which is the optimal filter (providing the most accurate estimates possible) for the case of a linear dynamical system with Gaussian noise, and sensors that also suffer from Gaussian noise. Let ω_d^2 represent the variance of the estimated Gaussian noise in the dynamical system, and let ω_s^2 repesent the sensor noise variance. The complementary filter (9.24) is equivalent to the Kalman filter if the parameters are chosen as $\alpha_1 = \sqrt{2\omega_d/\omega_s}$ and $\alpha_2 = \omega_d/\omega_s$ [60]. A large variety of alternative filtering methods exist; however, the impact of using different filtering methods is usually small relative to calibration, sensor error models, and dynamical system models that are particular to the setup. Furthermore, the performance requirements are mainly *perceptually* based, which could be different than the classical criteria around which filtering methods were designed.

Once the filter is running, its pose estimates can be used to aid the PnP problem. The PnP problem can be solved incrementally by perturbing the pose estimated by the filter, using the most recent accelerater outputs, so that the observed features are perfectly matched. Small adjustments can be made to the pose so that the sum-of-squares error is reduced to an acceptable level. In most case, this improves reliability when there are so few features visible that the PnP problem has ambiguous solutions. Without determining the pose incrementally, a catastrophic jump to another PnP solution might occur.

9.4 Tracking Attached Bodies

Many tracking problems involve estimating the motion of one body relative to another. For example, an eye rotates inside of its socket, which is part of the skull. Although the eye may have six DOFs when treated as a rigid body in space, its position and orientation are sufficiently characterized with two or three parameters once the head pose is given. Other examples include the head relative to the torso, a hand relative to the wrist, and the tip of a finger relative to its middle bone. The entire human body can even be arranged into a tree of attached bodies, based on a skeleton. Furthermore, bodies may be attached in a similar way for other organisms, such as dogs or monkeys, and machinery, such as robots or cars. In the case of a car, the wheels rotate relative to the body. The result is a *multibody system*. The mathematical characterization of the poses of bodies



(a) (b)

Figure 9.18: (a) The first and sometimes the fourth Purkinje images of an IR light source are used for eye tracking. (Figure from Wikipedia.) (b) The first Purkinje image generates a bright reflection as shown. (Picture from Massimo Gneo, Maurizio Schmid, Silvia Conforto, and Tomasso D'Alessio.)

relative to each other is called *multibody kinematics*, and the full determination of their velocities and accelerations is called *multibody dynamics*.

Eye tracking Eye tracking systems been used by vision scientists for over a century to study eye movements. Three main uses for VR are: 1) To accomplish foveated rendering, as mentioned in Section 5.4, so that high-resolution rendering need only be performed for the part of the image that lands on the fovea. 2) To study human behavior by recording tracking data so that insights may be gained into VR sickness, attention, and effectiveness of experiences. 3) To render the eye orientations in VR so that social interaction may be improved by offering eye-contact and indicating someone's focus of attention; see Section 10.4.

Three general categories of eye-tracking approaches have been developed. The first is *electro-oculography* (*EOG*), which obtains measurements from several electrodes placed on the facial skin around each eye. The recorded potentials correspond to eye muscle activity, from which the eye orientation relative to the head is determined through filtering. The second approach uses a contact lens, which contains a tiny magnetic coil that causes a potential change in a surrounding electromagnetic field. The third approach is called *video-oculography* (*VOG*), which shines IR light onto the eye and senses its *corneal reflection* using a camera or photodiodes. The reflection is based on *Purkinje images*, as shown in Figure 9.18. Because of its low cost and minimal invasiveness, this is the most commonly used method today. The contact lens approach is the most accurate; however, it is also the most uncomfortable.

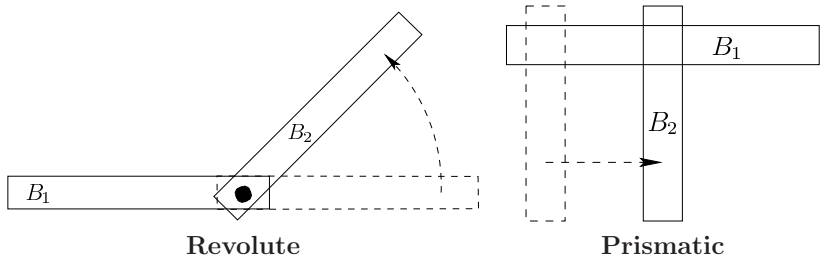


Figure 9.19: Two types of 2D joints: a revolute joint allows one link to rotate with respect to the other, and a prismatic joint allows one link to translate with respect to the other.

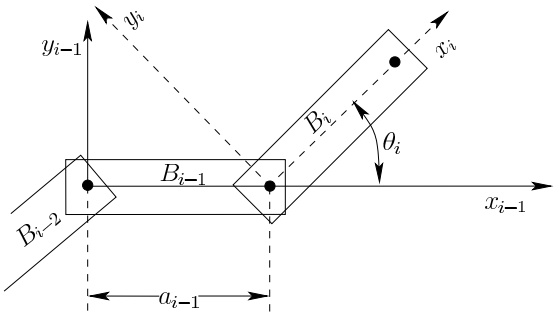
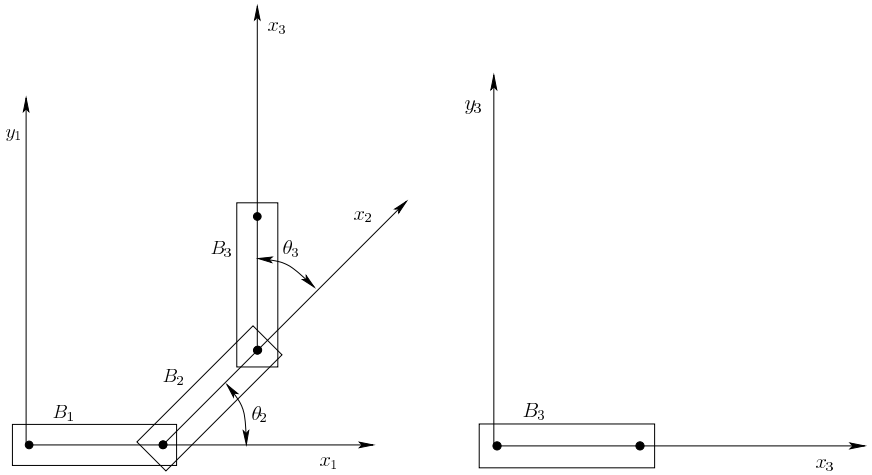


Figure 9.20: The body frame of each B_i, for $1 < i < m$, is based on the joints that connect B_i to B_{i-1} and B_{i+1}.

Forward kinematics Suppose that an eye tracking method has estimated the eye orientation relative to the human skull and it needs to be placed accordingly in the virtual world. This transformation must involve a combination of the head and eye transforms. For a more complicated problem, consider placing the right index finger in the world by using pose of the torso along with all of the angles formed between bones at each joint. To understand how these and other related problems are solved, it is helpful to first consider 2D examples.

Each body of a multibody system is called a *link*, and a pair of bodies are attached at a *joint*, which allows one or more DOFs of motion between them. Figure 9.19 shows two common ways that one planar body might move while attached to another. The *revolute joint* is most common and characterizes the motion allowed by a human elbow.

Consider defining a chain of m links, B_1 to B_m, and determining the location of a point on the last link. The points on each link are defined using coordinates of its own *body frame*. In this frame, the body appears as shown for B_{i-1} in Figure 9.20, with the origin at the joint that connects B_{i-1} to B_{i-2} and the x axis pointing



(a) A three-link chain (b) B_3 in its body frame

(c) T_3 puts B_3 in B_2's body frame (d) T_2T_3 puts B_3 in B_1's body frame

Figure 9.21: Applying the transformation T_2T_3 to the model of B_3. If T_1 is the identity matrix, then this yields the location in the virtual world of points in B_3.

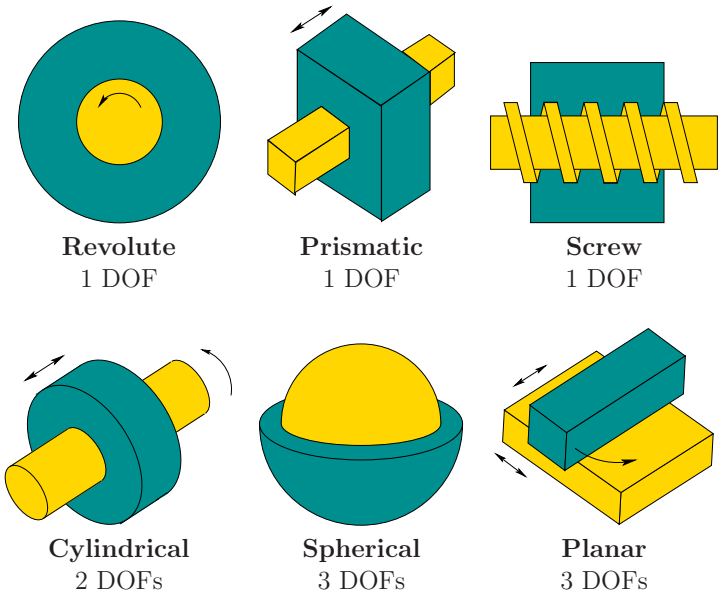


Figure 9.22: Types of 3D joints arising from the 2D surface contact between two bodies.

through the joint that connects B_{i-1} to B_i. To move the points on B_i to the proper location in the body frame of B_{i-1}, the homogeneous transform

$$T_i = \begin{pmatrix} \cos\theta_i & -\sin\theta_i & a_{i-1} \\ \sin\theta_i & \cos\theta_i & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{9.25}$$

is applied. This rotates B_i by θ_i, and then translates it along the x axis by a_{i-1}. For a revolute joint, θ_i is a variable, and a_{i-1} is a constant. For a prismatic joint, θ_i is constant and a_{i-1} is a variable.

Points on B_i are moved into the body frame for B_1 by applying the product $T_2 \cdots T_i$. A three-link example is shown in Figure 9.21. To move the first link B_1 into the world frame, a general 2D homogeneous transform can be applied:

$$T_1 = \begin{pmatrix} \cos\theta_i & -\sin\theta_i & x_t \\ \sin\theta_i & \cos\theta_i & y_t \\ 0 & 0 & 1 \end{pmatrix}. \tag{9.26}$$

This transform is simply added to the matrix product to move each B_i by applying $T_1T_2 \cdots T_i$.

A chain of 3D links is handled in the same way conceptually, but the algebra becomes more complicated. See Section 3.3 of [77] for more details. Figure 9.22

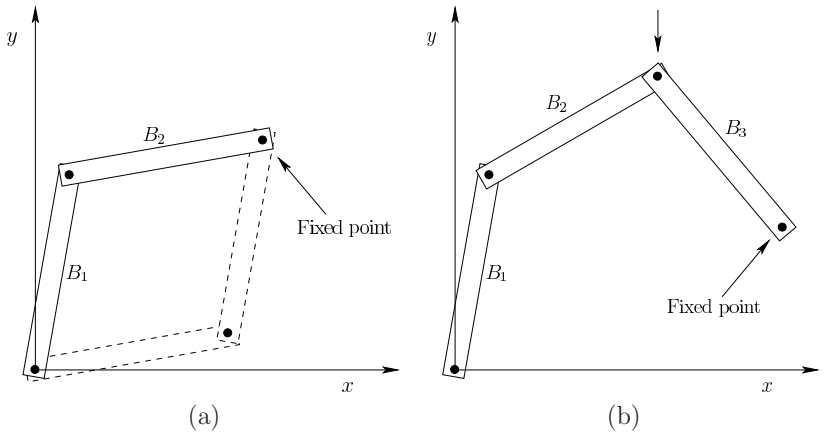


Figure 9.23: (a) The orientations of both links can be inferred from the position of the fixed point; however, there is a second solution if the angles are not restricted. (b) In the case of three links, a one-dimensional family of solutions exists when the end is fixed. This can be visualized by pushing down on the top joint, which would cause B_1 to rotate counter-clockwise. This is equivalent to the classical *four-bar mechanism*, which was used to drive the wheels of a steam engine (the fourth "link" is simply the fixed background).

shows six different kinds of joints that are obtained by allowing a pair of 3D links to slide against each other. Each link is assigned a convenient coordinate frame based on the joints. Each homogeneous transform T_i contains a mixture of constants and variables in which the variables correspond to the freedom allowed by the joint. The most common assignment scheme is called *Denavit-Hartenberg parameters* [55]. In some settings, it might be preferable to replace each T_i by a parameterized quaternion that rotates the body, followed by a simple addition that translates the body.

A tree of links may also be considered; a common example is a human torso serving as the root, with a head, two arms, and two legs being chains that extend from it. The human hand is another example. Coordinate frames in this case are often assigned using *Kleinfinger-Khalil parameters* [67].

Inverse kinematics Recall the PnP problem from Section 9.3, which involved calculating the pose of a body based on some observed constraints. A similar problem is to determine the joint parameters for a chain of bodies by observing constraints on the last body. A common example is to calculate the poses of the arm links by using only the pose of the hand. This is generally called the *inverse kinematics problem* (see [5] and Section 4.4 of [77]). As in the case of PnP, the number of solutions may be infinite, finite, one, or zero. Some 2D examples are

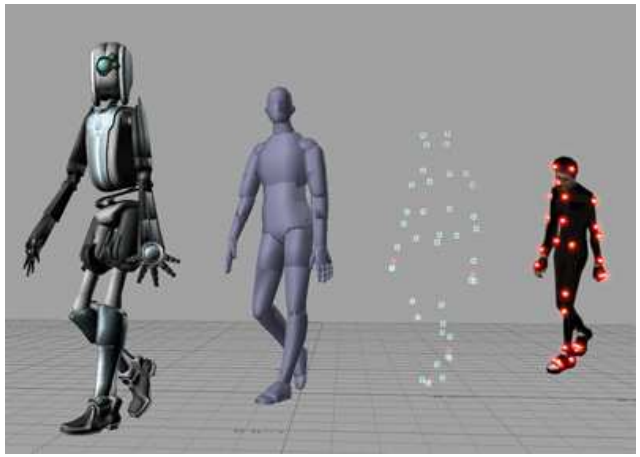
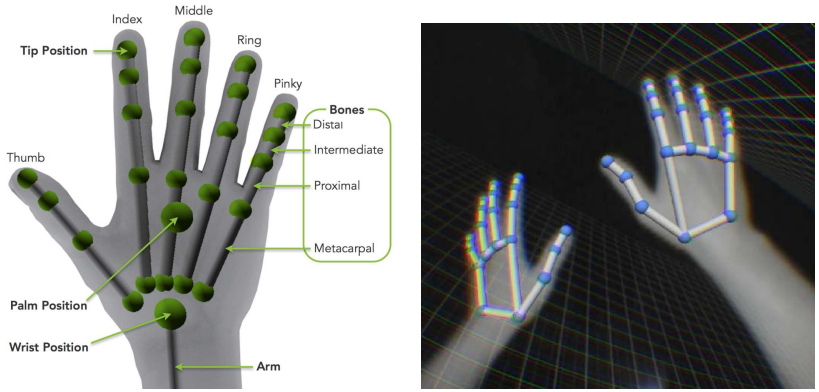


Figure 9.24: With a *motion capture* (*MOCAP*) system, artificial features are placed around the body of a human actor. The motions are extracted and matched to a kinematic model. Each rigid body in the model has an associated geometric model that is rendered to produce the final animated character. (Picture from Wikipedia user Hipocrite.)

shown in Figure 9.23. Generally, if the last link is constrained, the freedom of motion for the intermediate links increases as the number of links increases. The *Chebychev-Grübler-Kutzbach criterion* gives the number of DOFs, assuming the links are not in some special, singular configurations [6]. A common problem in animating video game characters is to maintain a kinematic constraint, such as the hand grasping a doorknob, even though the torso or door is moving. In this case, *iterative optimization* is often applied to perturb each joint parameter until the error is sufficiently reduced. The error would measure the distance between the hand and the doorknob in our example.

Motion capture systems Tracking systems for attached bodies use kinematic constraints to improve their accuracy. The most common application is tracking the human body, for which the skeleton is well-understood in terms of links and joints [169]. Such motion capture systems have been an important technology for the movie industry as the motions of real actors are brought into a virtual world for animation. Figure 9.24 illustrates the operation. Features, of the same kind as introduced in Section 9.3, are placed over the body and are visible to cameras mounted around the capture studio. The same options exist for visibility, with the most common approach over the past decade being to use cameras with surrounding IR LEDs and placing retroreflective markers on the actor.

To obtain a unique pose for each body part, it might seem that six features are needed (recall P6P from Section 9.3); however, many fewer are sufficient because



(a) (b)

Figure 9.25: (a) The hand model used by Leap Motion tracking. (b) The tracked model superimposed in an image of the actual hands.

of kinematic constraints. Additional features may nevertheless be used if the goal is to also capture skin motion as it moves along the skeleton. This is especially important for facial movement. Many new MOCAP technologies are currently under development. For example, a system developed by Noitom captures human body movement solely by placing IMUs on the body. Some systems capture motion by cameras alone, as in the case of Leap Motion (see Figure 9.25) for hand tracking, and systems by Microsoft and 8i for full-body tracking by extracting contours against a green screen. Solutions based on modern depth sensors may also become prevalent in the near future. One challenge is to make highly accurate and reliable systems for low cost and installation effort.

9.5 3D Scanning of Environments

Up until now, this chapter has described how to use sensors to track the motions of one or more rigid bodies. By contrast, this section describes how sensors are used to build geometric models of rigid bodies. These could be movable or stationary models, as introduced in Section 3.1. A movable model typically corresponds to an object that is be manipulated by the user, such as a sword, hammer, or coffee cup. These models are often built from a *3D scanner*, which images the object from many viewpoints in a controlled way. The object may be placed on a turntable that is surrounded by cameras and other sensors, or the sensors may move around while the object remains stationary; see Figure 9.26(a).

SLAM A 3D scanner is useful for smaller objects, with surrounding sensors facing inward. For larger objects and stationary models, the sensors are usually inside

(a) (b)

Figure 9.26: (a) The Afinia ES360 scanner, which produces a 3D model of an object while it spins on a turntable. (b) The FARO Focus3D X 330 is an outward-facing scanner for building accurate 3D models of large environments; it includes a GPS receiver to help fuse individual scans into a coherent map.

facing out; see Figure 9.26(b). A common example of a stationary model is the inside of a building. Scanning such models is becoming increasingly important for surveying and forensics. This is also the classical robotics problem of *mapping*, in which a robot carrying sensors builds a 2D or 3D representation of its world for the purposes of navigation and collision avoidance. Robots usually need to estimate their locations based on sensors, which is called the *localization* problem. Robot localization and tracking bodies for VR are fundamentally the same problems, with the main distinction being that known motion commands are given to robots, but the corresponding human intent is not directly given. Robots often need to solve mapping and location problems at the same time, which results in the *simultaneous localization and mapping* problem; the acronym *SLAM* is widely used. Due to the similarity of localization, mapping, and VR tracking problems, deep connections exist between robotics and VR. Therefore, many mathematical models, algorithms, and sensing technologies overlap.

Consider the possible uses of a large, stationary model for VR. It could be captured to provide a virtual world in which the user is placed at the current time or a later time. Image data could be combined with the 3D coordinates of the model, to produce a photorealistic model (recall Figure 2.14 from Section 2.2). This is achieved by texture mapping image patches onto the triangles of the model.

Live capture of the current location Rather than capturing a world in which to transport the user, sensors could alternatively be used to capture the physical world where the user is currently experiencing VR. This allows obstacles in the

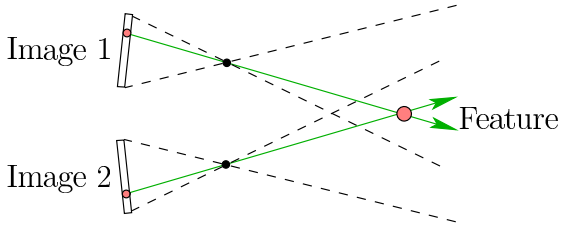


Figure 9.27: By using two cameras, *stereo vision* enables the location of a feature in the 3D world to be determined by intersecting the corresponding project ray from each camera. To accomplish this, the camera calibration parameters and relative poses must be known. Similarly, one camera could be replaced by a laser that illuminates the feature so that it is visible to the remaining camera. In either case, the principle is to intersect two visibility rays to obtain the result.

matched zone to be rendered in the virtual world, which might be useful for safety or to improve interactivity. For safety, the boundaries of the matched zone could be rendered to indicate that the user is about to reach the limit. Hazards such as a hot cup of coffee or a toddler walking across the matched zone could be indicated. Interactivity can be improved by bringing fixed objects from the physical world into the virtual world. For example, if the user is sitting in front of a desk, then the desk can be drawn in the virtual world. If she touches the virtual desk, she will feel the desk pushing back. This is an easy way to provide touch feedback in VR.

Are panoramas sufficient? Before embarking on the process of creating a large, detailed map of a surrounding 3D world, it is important to consider whether it is necessary. As mentioned in Section 7.5, panoramic images and videos are becoming increasingly simple to capture. In some applications, it might be sufficient to build an experience in which the user is transported between panoramas that were captured from many locations that are close to each other.

The main ingredients Building a 3D model from sensor data involves three important steps:

1. Extracting a 3D *point cloud* from a fixed location.
2. Combining point clouds from multiple locations.
3. Converting a point cloud into a mesh of triangles.

For the first step, a sensor is placed at a fixed position and orientation while 3D points are extracted. This could be accomplished in a number of ways. In theory, any of the depth cues from Section 6.1 can be applied to camera images to extract 3D points. Variations in focus, texture, and shading are commonly used in

computer vision as monocular cues. If two cameras are facing the same scene and their relative positions and orientations are known, then binocular cues are used to determine depth. By identifying the same natural feature in both images, the corresponding visibility rays from each image are intersected to identify a point in space; see Figure 9.27. As in Section 9.3, the choice between natural and artificial features exists. A single camera and an IR projector or laser scanner may be used in combination so that depth is extracted by identifying where the lit point appears in the image. This is the basis of the Microsoft Kinect sensor (recall Figure 2.10 from Section 2.1). The resulting collection of 3D points is often called a *point cloud*.

In the second step, the problem is to merge scans from multiple locations. If the relative position and orientation of the scanner between scans is known, then the problem is solved. In the case of the object scanner shown in Figure 9.26(a), this was achieved by rotating the object on a turntable so that the position remains fixed and the orientation is precisely known for each scan. Suppose the sensor is instead carried by a robot, such as a drone. The robot usually maintains its own estimate of its pose for purposes of collision avoidance and determining whether its task is achieved. This is also useful for determining the pose that corresponds to the time at which the scan was performed. Typically, the pose estimates are not accurate enough, which leads to an optimization problem in which the estimated pose is varied until the data between overlapping scans nicely aligns. The *estimation-maximization (EM) algorithm* is typically used in this case, which incrementally adjusts the pose in a way that yields the maximum likelihood explanation of the data in a statistical sense. If the sensor is carried by a human, then extra sensors may be included with the scanning device, as in the case of GPS for the scanner in Figure 9.26(b); otherwise, the problem of fusing data from multiple scans could become too difficult.

In the third stage, a large point cloud has been obtained and the problem is to generate a clean geometric model. Many difficulties exist. The point density may vary greatly, especially where two or more overlapping scans were made. In this case, some points may be discarded. Another problem is that outliers may exist, which correspond to isolated points that are far from their correct location. Methods are needed to detect and reject outliers. Yet another problem is that large holes or gaps in the data may exist. Once the data has been sufficiently cleaned, surfaces are typically fit to the data, from which triangular meshes are formed. Each of these problems is a research area in itself. To gain some familiarity, consider experimenting with the open-source *Point Cloud Library*, which was developed to handle the operations that arise in the second and third stages. Once a triangular mesh is obtained, texture mapping may also be performed if image data is also available. One of the greatest challenges for VR is that the resulting models often contain numerous flaws which are much more noticeable in VR than on a computer screen.

Further Reading

Need IMU calibration papers.

Mag calibration: [47, 71, 149].

Complementary filters over rotation group, with convergence proofs: [88].

Why you might as well use a complementary filter if your system model is just a double integrator: [60].

Fusion of IMU and Vision for Absolute Scale Estimation: Nutzi, Scaramuzza, Weiss, Siegwart.

Oculus VR blogs: [80, 78, 79]

Oculus Rift tracking: [81]

PnP with ambiguities analyzed: [128, 161]

Fast PnP, linear in number of points: [170]

Bundle adjustment: [148]

RANSAC for robust outlier rejection: [40]

Old but good VR tracking survey: [156].

Survey of human body tracking [171]

Eye tracking references: [36, 153]

Kinematics: [5, 7]; [29]

SLAM ([147]; Section 12.3.5 of LaValle 06)

Filtering or sensor fusion in the larger context can be characterized in terms of *information spaces* (Chapter 11 of [77]).

Chapter 10

Interaction

Chapter Status	Taken from *Virtual Reality*, S. M. LaValle
	This online chapter is not the final version! Check **http://vr.cs.uiuc.edu/** for information on the latest draft version. This draft was compiled on June 29, 2016.

How should users interact with the virtual world? How should they move about? How can they grab and place objects? How should they interact with representations of each other? How do they interact with files or the Internet? The following insight suggests many possible interfaces.

Universal Simulation Principle: Any interaction mechanism from the real world can be simulated in VR.

For example, the user might open a door by turning a knob and pulling. As another example, the user operate a virtual aircraft by sitting in a mock-up cockpit (as was shown in Figure 1.14). One could even simulate putting on a VR headset, leading to an experience that is comparable to a dream within a dream!

In spite of the universal simulation principle, recall from Section 1.1 that the goal is not necessarily realism. It is often preferable to make the interaction *better than reality*. Therefore, this chapter introduces *interaction mechanisms* that may not have a counterpart in the physical world.

Section 10.1 introduces a general motor learning and control concepts. The most important concept is *remapping*, in which a motion in the real world may be mapped into a substantially different motion in the virtual world. This enables many powerful interaction mechanisms. The task is to develop ones that are easy to learn, easy to use, effective for the task, and provide a comfortable user experience. Section 10.2 discusses how the user may move himself in the virtual world, while remaining fixed in the real world. Section 10.3 presents ways in which the user may interact with other objects in the virtual world. Section 10.4

discusses social interaction mechanisms, which allow users to interact directly with each other. Section 10.5 briefly considers some additional interaction mechanisms, such as editing text, designing 3D structures, and connecting to the Internet.

10.1 Motor Programs and Remapping

Motor programs Throughout our lives, we develop fine motor skills to accomplish many specific tasks, such as handwriting, tying shoelaces, throwing a ball, or riding a bicycle. These are often called *motor programs*, and are learned through repetitive trials, with gradual improvements in precision and ease as the amount of practice increases. Eventually, we produce the motions without even having to pay attention to them. Examples from the physical world include throwing a dart, swimming, doing a flip, and driving a car.

In the same way, most of us have learned how to use interfaces to computers, such as keyboards, mice, and game controllers. Some devices are easier to learn than others. For example, a mouse does not take long, but typing quickly on a keyboard takes years to master. What makes one skill harder to learn than another? This is not always easy to predict, as illustrated by the *backwards brain bicycle*, which was designed by Destin Sandlin by reversing the steering operation so that turning the handlebars left turns the front wheel to the right []. It took Sandlin six months learn how to ride it, and at the end he was unable to ride an ordinary bicycle. Thus, he unlearned how to ride a normal bicycle and the expense of learning the new one.

Design considerations In the development of interaction mechanisms for VR, the main considerations are:

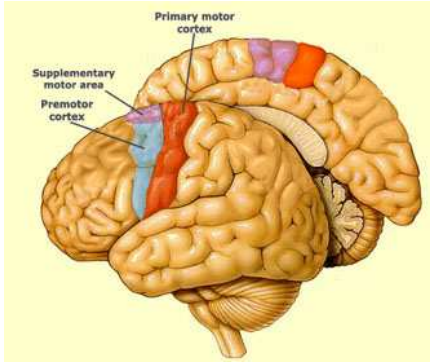
1. Effectiveness for the task in terms of achieving the required speed, accuracy, and motion range, if applicable.

2. Difficulty of learning the new motor programs; ideally, the user should not be expected to spend many months mastering a new mechanism.

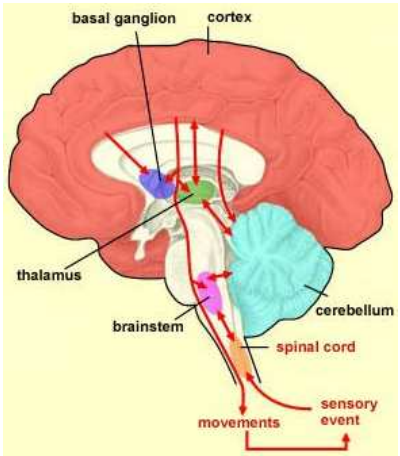
3. Ease of use in terms of cognitive load; in other words, the interaction mechanism should require little or no focused attention after some practice.

4. Overall comfort during use over extended periods; the user should not develop muscle fatigue, unless the task is to get some physical exercise.

To design and evaluate new interaction mechanisms, it is helpful to start by understanding the physiology and psychology of acquiring the motor skills and programs. Chapters 5 and 6 covered these for visual perception, which is the process of converting sensory *input* into a perceptual experience. We now consider the corresponding parts for generating *output* in the form of body motions in the physical world. In this case, the brain sends motor signals to the muscles, causing



(a)



(b)

Figure 10.1: (a) Part of the cerebral cortex is devoted to motion. (b) Many other parts interact with the cortex to produce and execute motions, including the thalamus, spinal cord, basal ganglion, brain stem, and cerebellum. (Figures provided by The Brain from Top to Bottom, McGill University.)

them to move, while at the same time incorporating sensory feedback by utilizing the perceptual processes.

The neurophysiology of movement First consider the neural hardware involved in learning, control, and execution of voluntary movements. As shown in Figure 10.1(a), parts of the cerebral cortex are devoted to motion. The *primary motor cortex* is the main source of neural signals that control movement, whereas the *premotor cortex* and *supplementary motor area* appear to be involved in the preparation and planning of movement. Many more parts are involved in motion and communicate through neural signals, as shown in Figure 10.1(b). The most interesting part is the *cerebellum*, meaning "little brain", which is located at the back of the skull. It seems to be a special processing unit that is mostly devoted to motion, but is also involved in functions such as attention and language. Damage to the cerebellum has been widely seen to affect fine motor control and learning of new motor programs. It has been estimated to contain around 101 billion neurons [4], which is far more than the entire cerebral cortex, which contains around 20 billion. Even though the cerebellum is much smaller, a large number is achieved through smaller, densely packed cells. In addition to coordinating fine movements, it appears to be the storage center for motor programs.

One of the most relevant uses of the cerebellum for VR is in learning *sensorimotor relationships*, which become encoded into a motor program. All body motions

(a)

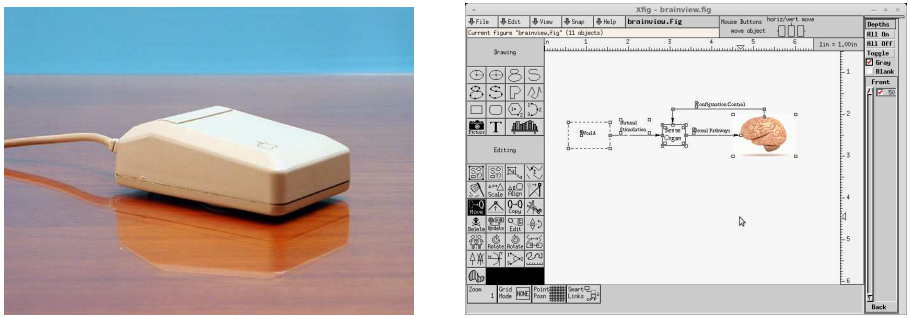
(b)

Figure 10.2: (a) Atari 2600 Paddle controller. (b) The Atari Breakout game, in which the bottom line segment is a virtual paddle that allows the ball to bounce to the top and eliminate bricks upon contacts.

involve some kind of sensory *feedback*. The most common example is *hand-eye coordination*; however, even if you move your arms with your eyes closed, proprioception provides information in the form of efference copies of the motor signals. Developing a tight connection between sensory and perceptual information and motor control signals is crucial to many tasks. This is also widely known in engineered systems, in which sensor-feedback and motor control are combined in applications such as robotics and aircraft stabilization; the subject that deals with this is called *control theory*. It is well-known that a *closed-loop* system is preferred in which sensor information provides feedback during execution, as opposed to *open-loop*, which specifies the motor signals as a function of time.

One of the most important factors is how long it takes to learn a motor program. As usual, there is great variation across humans. A key concept is *neuroplasticity*, which is the potential of the brain to reorganize its neural structures and form new pathways to adapt to new stimuli. Toddlers have a high level of neuroplasticity, which becomes greatly reduced through the process of *synaptic pruning*. This causes healthy adults to have about half as many synapses per neuron than a child of age two or three [50]. Unfortunately, the result is that adults have a harder time acquiring new skills such as learning a new language or learning how to use a complicated interface. In addition to the reduction of neuroplasticity with age, it also greatly varies among people of the same age.

Learning motor programs Now consider learning a motor program for a computer interface. A simple, classic example is the video game *Breakout*, which was developed by Atari in 1976. The player turns a knob, shown in Figure 10.2. This causes a line segment on the bottom of the screen to move horizontally. The Paddle contains a potentiometer that with calibration allows the knob orientation to be reliably estimated. The player sees the line segment position on the screen



(a) (b)

Figure 10.3: (a) The Apple Macintosh mouse. (b) As a mouse moves across the table, the virtual finger on the screen moves correspondingly, but is rotated by 90 degrees and travels over longer distances.

and quickly associates the knob orientations. The learning process therefore involves taking information from visual perception and the proprioception signals from turning the knob and determining the sensorimotor relationships. Skilled players could quickly turn the knob so that they could move the line segment much more quickly than one could move a small tray back and forth in the real world. Thus, we already have an example where the virtual world version allows better performance than in reality!

In the Breakout example, a one-dimensional *mapping* was learned between the knob orientation and the line segment position. Many alternative control schemes could be developed; however, they are likely to be more frustrating. If you find an emulator to try Breakout, it will most likely involve using keys on a keyboard to move the segment. In this case, the amount of time that a key is held down corresponds to the segment displacement. The segment velocity is set by the program, rather than the user. A reasonable alternative using modern hardware might be to move a finger back and forth over a touch screen while the segment appears directly above it. The finger would not be constrained enough due to extra DOFs and the rapid back and forth motions of the finger may lead to unnecessary fatigue, especially if the screen is large. Furthermore, there are conflicting goals in positioning the screen: Making it as visible as possible versus making it comfortable for rapid hand movement over a long period of time. In the case of the Paddle, the motion is accomplished by the fingers, which have high dexterity, while the forearm moves much less. The mapping provides an association between body movement and virtual object placement that achieves high accuracy, fast placement, and long-term comfort.

Figure 10.3 shows a more familiar example, which is the computer mouse. As the mouse is pushed around on a table, encoders determine the position, which is converted into a pointer position on the screen. The sensorimotor mapping

seems a bit more complex than in the Breakout example. Young children seem to immediately learn how to use the mouse, whereas older adults require some practice. The 2D position of the mouse is mapped to a 2D position on the screen, with two fundamental distortions. Th0Now suppose that "h" is instantly placed on the screen in a familiar font. Our visual perception system recognizes the "h" as being equivalent to the version on paper. Thus, typing the key results in the perception of "h". This is quite a comfortable, fast, and powerful operation. The amount of learning required seems justified by the value of the output.

Motor programs for VR The examples given so far already seem closely related to VR. A perceptual experience is controlled by body movement that is sensed through a hardware device. Using the universal simulation principle, any of these and more could be brought into a VR system. The physical interaction part might be identical (you could really be holding an Atari Paddle), or it could be simulated through another controller. Think about possible designs.

Using the tracking methods of Chapter 9, the position and orientation of body parts could be reliably estimated and brought into VR. For the case of head tracking, it is essential to maintain accurately determine the viewpoint with high accuracy and zero effective latency; otherwise, the VR experience is significantly degraded. This is essential because the perception of stationarity must be maintained for believability and comfort. The motion of the sense organ must be matched by a tracking system.

Remapping For the motions of other body parts, this perfect matching is not critical. Our neural systems can instead learn associations that are preferable in terms of comfort, in the same way as the Atari Paddle, mouse, and keyboard work in the real world. In many cases, we want to do *remapping*, which involves learning a sensorimotor mapping that produces different results in a virtual world than one would expect from the real world. The keyboard example above is one of the most common examples of remapping. The process of pushing a pencil across paper to produce a letter has been replaced by pressing a key. The term remapping is even used with keyboards to mean the assignment of one or more keys to another key.

Remapping is natural for VR. For example, rather than reaching out to grab a virtual door knob, one could press a button to open the door. For a simpler case, consider holding a controller for which the pose is tracked through space, as allowed by the HTC Vive VR system. A scaling parameter could be set so that one centimeter of hand displacement in the real world corresponds to two centimeters of displacement in the virtual world. This is similar to the scaling parameter for the mouse. Section 10.2 covers the remapping from natural walking in the real world to achieving the equivalent in a virtual world by using a controller. Section 10.3 covers object interaction methods, which are again achieved by remappings. You can expect to see many new remapping methods for VR in the coming years.

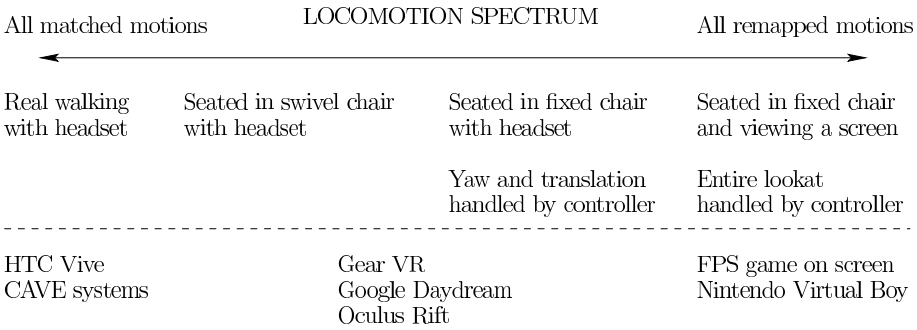


Figure 10.4: Moving from left to right, the amount of viewpoint mismatch between real and virtual motions increases.

10.2 Locomotion

Suppose that the virtual world covers a much larger area than the part of the real world that is tracked. In other words, the matched zone is small relative to the virtual world. In this case, some form of interaction mechanism is needed to move the user in the virtual world while she remains fixed within the tracked area in the real world. An interaction mechanism that moves the user in this way is called *locomotion*. It is as if the user is riding in a virtual vehicle that is steered through the virtual world.

Figure 10.4 shows a spectrum of common locomotion scenarios. At the left, the user walks around in an open space while wearing a headset. No locomotion is needed unless the virtual world is larger than the open space. This case involves no mismatch between real and virtual motions.

The two center cases correspond to a seated user wearing a headset. In these cases, an interaction mechanism is used to change the position of the matched zone in the virtual world. If the user is seated in a swivel chair, then he could change the direction he is facing (yaw orientation) by rotating the chair. This can be considered as orienting the user's torso in the virtual world. If the user is seated in a fixed chair, then the virtual torso orientation is typically changed using a controller, which results in more mismatch. The limiting case is on the right of Figure 10.4, in which there is not even head tracking. If the user is facing a screen, as in the case of a first-person shooter game on a screen, then a game controller is used to change the position and orientation of the user in the virtual world. This is the largest amount of mismatch because all changes in viewpoint are generated by the controller.

Redirected walking If the user is tracked through a very large space, such as a square region of at least 30 meters on each side, then it is possible to make her think she is walking in straight lines for kilometers while she is in fact walking in circles. This technique is called *redirected walking* [116]. Walking along a straight

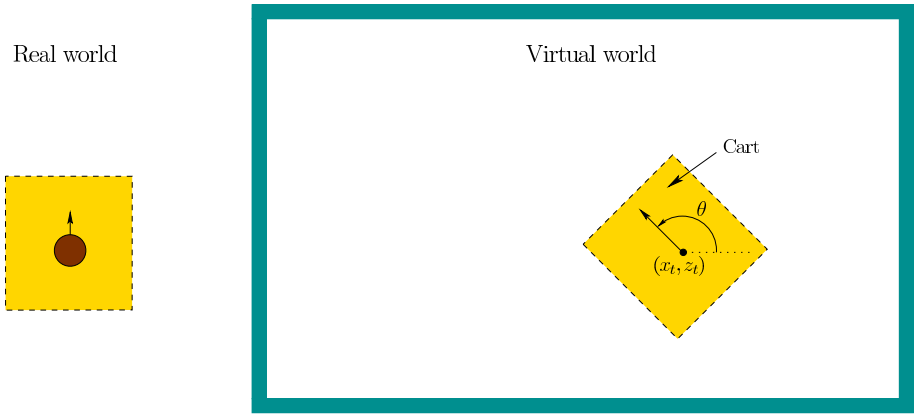


Figure 10.5: Locomotion along a horizontal terrain can be modeled as steering a cart through the virtual world. A top-down view is shown. The yellow region is the matched zone (recall Figure 2.15), in which the user's viewpoint is tracked. The values of x_t, z_t, and θ are changed by using a controller.

line over long distances without visual cues is virtually impossible for humans (and robots!) because in the real world it is impossible to achieve perfect symmetry. One direction will tend to dominate through an imbalance in motor strength and sensory signals, causing people to travel in circles.

Imagine a VR experience in which a virtual city contains long, straight streets. As the user walks down the street, the yaw direction of the viewpoint can be gradually varied. This represents a small amount of mismatch between the real and virtual worlds, and it causes the user to walk along circular arcs. The main trouble with this technique is that the user has free will and might decide to walk to the edge of the matched zone in the real world, even if he cannot directly perceive it. In this case, an unfortunate, disruptive warning might appear, suggesting that he must rotate to reset the yaw orientation.

Locomotion implementation Now consider the middle cases from Figure 10.4 of sitting down and wearing a headset. Locomotion can then be simply achieved by moving the viewpoint with a controller. It is helpful to think of the matched zone as a controllable cart that moves across the ground of the virtual environment; see Figure 10.5. First consider the simple case in which the ground is a horizontal plane. Let T_{track} denote the homogeneous transform that represents the tracked position and orientation of the cyclopean (center) eye in the physical world. The methods described in Section 9.2 could be used to provide T_{track} for the current time.

The position and orientation of the cart is determined by a controller. The

homogeneous matrix:

$$T_{cart} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & x_t \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & z_t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10.1}$$

encodes the position (x_t, z_t) and orientation θ of the cart (as a yaw rotation, borrowed from (3.18)). The height is set at $y_t = 0$ in (10.1) so that it does not change the height determined by tracking or other systems (recall from Section 9.2 that the height might be set artifically if the user is sitting in the real world, but standing in the virtual world).

The eye transform is obtained by chaining T_{track} and T_{cart} to obtain

$$T_{eye} = (T_{track} T_{cart})^{-1} = T_{cart}^{-1} T_{track}^{-1} \tag{10.2}$$

Recall from Section 3.4 that the eye transform is the *inverse* of the transform that places the geometric models. Therefore, (10.2) corresponds to changing the perspective due to the cart, followed by the perspective of the tracked head on the cart.

To move the viewpoint for a fixed direction θ, the x_t and z_t components are obtained by integrating a differential equation:

$$\begin{aligned} \dot{x}_t &= s\cos\theta \\ \dot{z}_t &= s\sin\theta. \end{aligned} \tag{10.3}$$

Integrating (10.3) over a time step Δt, the position update appears as

$$\begin{aligned} x_t[k+1] &= x_t[k] + \dot{x}_t \Delta t \\ z_t[k+1] &= z_t[k] + \dot{z}_t \Delta t. \end{aligned} \tag{10.4}$$

The variable s in (10.3) is the forward speed. The average human walking speed is about 1.4 meters per second. The virtual cart can be moved forward by pressing a button or key that sets $s = 1.4$. Another button can be used to assign $s = -1.4$, which would result in backward motion. If no key or button is held down, then $s = 0$, which causes the cart to remain stopped. An alternative control scheme is to use the two buttons to increase or decrease the speed, until some maximum limit is reached. In this case, motion is sustained without holding down a key.

Keys could also be used to provide lateral motion, in addition to forward/backward motion. This is called *strafing* in video games. It should be avoided, if possible, because it cases unnecessary lateral vection.

Issues with changing direction Now consider the orientation θ. To move in a different direction, θ needs to be reassigned. The assignment could be made based on the user's head yaw direction. This becomes convenient and comfortable when

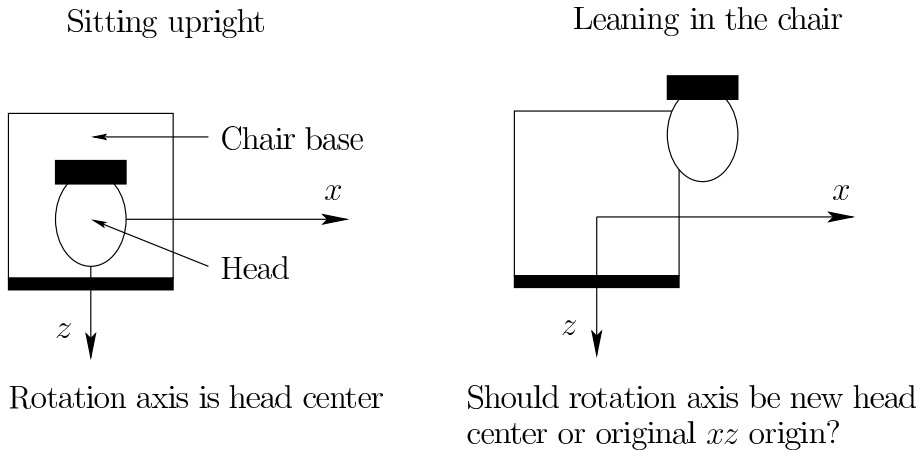


Figure 10.6: On the right the yaw rotation axis is centered on the head, for a user who is upright in the chair. On the left, the user is leaning over in the chair. Should the rotation axis remain fixed, or move with the user?

the user is sitting in a swivel chair and looking forward. By rotating the swivel chair, the direction can be set. (However, this could become a problem for a wired headset because the cable could wrap around the user.)

In a fixed chair, it may become frustrating to control θ because the comfortable head yaw range is limited to only 60 degrees in each direction (recall Figure 5.21). In this case, buttons can be used to change θ by small increments in clockwise or counterclockwise directions. Unfortunately, changing θ according to constant angular velocity causes yaw vection, which is nauseating to many people. Many users prefer an option in which they can tap a button to instantly yaw about 10 degrees each time. If the increments are too small, then vection appears again, and if the increments are too large, then users become confused about their orientation.

Another issue is where to locate the center of rotation, as shown in Figure 10.6. What happens when the user moves his head away from the center of the chair in the real world? Should the center of rotation be about the original head center or the new head center? If it is chosen as the original center, then the user will perceive a large translation as θ is changed. However, this would also happen in the real world if the user were leaning over while riding in a cart. If it is chosen as the new head center, then the amount of translation is less, but might not correspond as closely to reality.

For another variation, the car-like motion model (8.29) from Section 8.3.2 could be used so that the viewpoint cannot be rotated without translating. In other words, the avatar would have a minimum turning radius. In general, the viewpoint could changed by controlling any virtual vehicle model. Figure 1.1 from Chapter 1 showed an example in which the "vehicle" is a bird.

Vection reduction strategies The main problem with locomotion is vection, which leads to VR sickness. Recall from Section 8.4 that six different kinds of vection occur, one for each DOF. Furthermore, numerous factors were given that affect the sensitivity to vection. Reducing the intensity of these factors should reduce vection and, hopefully, VR sickness.

Several strategies for reducing vection-based VR sickness are:

1. If the field of view for the optical flow is reduced, then the vection is weakened. A common example is to make a cockpit or car interior that blocks most of the optical flow.

2. If the viewpoint is too close to the ground, then the magnitudes of velocity and acceleration vectors of moving features are higher. This is why you might feel as if you are traveling faster in a small car that is low to the ground in comparison to riding at the same speed in a truck or minivan.

3. Surprisingly, a larger mismatch for a short period of time may be preferable to a smaller mismatch over a long period of time; see Figure 10.7.

4. Having high spatial frequency will yield more features for the human vision system to track. Therefore, if the passing environment is smoother, with less detail, then vection should be reduced. Consider the case of traveling up a staircase. If the steps are clearly visible so that they appear as moving horizontal stripes, then the user may quickly come nauseated by the strong vertical vection signal.

5. Reducing contrast, such as making the world seem hazy or foggy while accelerating, may help.

6. Providing other sensory cues such as blowing wind or moving audio sources might provide stronger evidence of motion. Including vestibular stimulation in the form of a rumble or vibration may also help lower the confidence of the vestibular signal. Even using head tilts to induce changes in virtual-world motion may help because it would cause distracting vestibular signals.

7. If the *world* is supposed to be moving, rather than the user, then making it clear through cues or special instructions can help.

8. Providing specific tasks, such as firing a laser as flying insects, may provide enough distraction from the vestibular conflict. If the user is instead focused entirely on the motion, then she might become sick more quickly.

9. The adverse effects of vection may decrease through repeated practice. People who regularly play FPS games in front of a large screen already seem to have reduced sensitivity to vection in VR. Requiring users to practice before sickness is reduced might not be a wise strategy for companies hoping to introduce new products. Imagine trying some new food that makes you

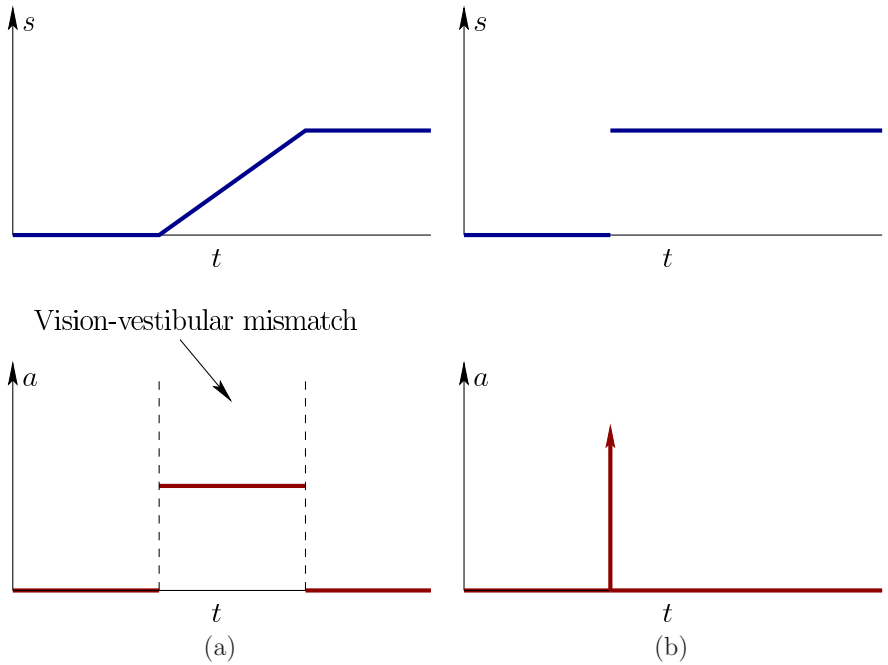


Figure 10.7: (a) Applying constant acceleration over a time interval to bring the avatar from being fix up to a speed limit. The upper plot shows the speed over time. The lower plot shows the acceleration. The interval of time over which there is nonzero acceleration corresponds to a mismatch with the vestibular sense. (b) In this case, an acceleration impulse is appled, resulting in the desired speed limit being immediately achieved. In this case, the mismatch occurs over a time interval that is effectively zero length. In practice, the perceived speed changes in a single pair of consecutive frames. Surprisingly, case (b) is much more comfortable than (a). It seems the brain prefers an outlier for a very short time inverval, as supposed to a smaller, sustained mismatch over a longer time interval (such as 5 seconds).

nauseated after the first 20 times of eating it, but then gradually becomes more acceptable. Who would keep trying it?

A final suggestion is to avoid locomotion wherever possible! Try to design experiences that do not critically depend on it.

Non-planar locomotion Now consider more complicated locomotion cases. If the user is walking over a terrain, then the y component can be simply increased or decreased to reflect the change in altitude. This may seem realistic, but keep in mind that it increases the amount of mismatch between the real and virtual worlds because vertical vection is combined with forward vection.

In the case of moving through a 3D medium, all six forms of vection from Section 8.4 become enabled. Common settings include a virtual spacecraft, aircraft, or scuba diver. Yaw, pitch, and roll vection can be easily generated. For example, imagine flying a virtual spacecraft. By rolling the craft, roll vection can be caused as the stars spin around in a circular pattern. If a developer must make a craft move in these ways, then the prior suggestions for reducing vection intensity should be followed. Furthermore, careful experimentation with human subjects should be performed to determine which forms of vection are worse in the particular application; see Chapter 12. To avoid singularities, for systems in which all 3 DOFs of rotational motion are possible, the transformations are best maintained in terms of quaternions (recall the issues from Section 3.3).

Adding special effects that move the viewpoint will cause further difficulty with vection. For example, making an avatar jump up and down will cause vertical vection. It is also a bad idea to account for swaying head motions while walking because of the increased mismatch. Imagine a far worse case of looking out through the eyes of an avatar that performance gymnastics. The view of the world may become unbearable during multiple flips.

Specialized hardware Many kinds of hardware have been developed to support locomotion. One of the oldest examples is to create an entire cockpit for aircraft flight simulation (recall Figure 1.14). Figure 10.8(a) shows an *omnidirectional treadmill* that enables walking in any direction and over any distance. Exercise machines, such as a stationary bicycle have been connected to VR systems so that the user can pedal and steer to guide herself through a large virtual world, as shown in Figure 10.8(b). Figure 1.1 showed a mechanical platform sytsem for virtual flying like a bird. Various locomotion systems have also been designed around omnidirectional treadmills, but it has been difficult to make them affordable and comfortable.

Teleportation The locomotion methods so far have mainly focused on reproducing experiences that are familiar in the real world, which provide instances of the universal simulation principle. In VR, however, we could also move in ways that are physical implausible. The most common is *teleportation*, which it works

(a) (b)

Figure 10.8: (a) An omnidirectional treadmill used in a CAVE sytsem by the US Army for trainining. (b) A home-brew bicycle riding system connected to a VR headset, developed by Paul Dyan.

like a transporter in the TV series Star Trek. The user is immediately transported to another location.

How is the desired location determined? One simple mechanism is a *virtual laser pointer* (or *3D mouse*), which is accomplished by the user holding a controller that is similar in shape to a laser pointer in the real world. A smart phone could even be used. The user rotates the controller to move a laser dot in the virtual world. This requires performing a ray casting operation (recall from Section 7.1) to find the nearest visible triangle, along the ray that corresponds to the laser light.

To select a location where the user would prefer to stand, she could simply point the virtual laser and press a key to be instantly teleported. Places that are not visible can be selected by using a pop-up map, or even performing a text-based search (voice commands could be used instead of typing). One method, called *world in miniature*, involves showing the user a virtual small-scale version of the environment [143]. This is effectively a 3D map.

Wayfinding The cognitive problem of learning a spatial representation and using it to navigate is called *wayfinding*. This is a higher-level process than the low-level locomotion mechanism, but the two are closely related. One trouble with locomotion systems that are not familiar in the real world is that users might not learn the spatial arrangement of the world around them. Would your brain still form place cells for an environment in the real world if you were able to teleport from place to place? We widely observe this phenomenon with people who learn to navigate a city using only GPS or taxi services, rather than doing their own wayfinding.

The teleportation mechanism reduces vection, and therefore VR sickness; however, it may come at the cost of reduced learning of the spatial arrangement of the environment. When performing teleportation, it is important not to change

the yaw orientation of the viewpoint; otherwise, the user may become eve more disoriented. He might not understand where he is now positioned and oriented in the virtual world relative to the previous location.

Note that the universal simulation principle can once again be employed to borrow any effective navigation tools from the real world. If virtual buildings and cities are laid out in ways that are common in the real world, then they should be easier to navigate. Signs and landmarks can even be placed into the virtual world to help with navigation. In the real world, signs often tell us the locations of exits, the names of streets, or the boundary of a district. Landmarks such as tall buildings, windmills, or towers provide visual cues that are effective for navigation over long distances. Many of these ideas are discussed in Chapter 7 of [20].

10.3 Manipulation

We interact with objects in the real world for many reasons. You might eat a bowl of soup by moving a spoon between the bowl and your mouth. You might pick up a rock and throw it as far as possible. You might put on a pair of pants. These examples and many more fall under the topic of *manipulation*. In the real world, manipulation involves complex sensorimotor relationships which, through evolution and experience, enable us to manipulate objects under a wide variety of settings. The variation of objects includes differences in size, weight, friction, flexibility, temperature, fragility, and so on. Somehow our bodies can handle that. Getting robots to perform the manipulation in the ways that humans do has been a long and frustrating road, with only limited success.

Because of manipulation complexity in the real world, it is an ideal candidate for applying the remapping concepts from Section 10.1 to make manipulation as simple as possible in VR. The virtual world does not have to follow the complicated physics of manipulation. It is instead preferable to make operations such as selecting, grasping, manipulating, carrying, and placing an object as fast and easy as possible. Furthermore, extensive reaching or other forms of muscle strain should be avoided, unless the VR experience is designed to provide exercise.

Avoid gorilla arms One of the most common misconceptions among the public is that the interface used by Tom Cruise in the movie Minority Report is desirable; see Figure 10.9. In fact, it quickly leads to the well-known problem of *gorilla arms*, in which the user quickly feels fatigue from extended arms. How long can you hold your arms directly in front of yourself without becoming fatigued?

Selection One of the simplest ways to select an object in the virtual world is with the virtual laser pointer, which was described in Section 10.2. Several variations may help improve the selection process. For example, the user might instead hold a virtual flashlight that illuminates potential selections. The field of view of the flashlight could be adjustable [43]. A virtual mirror could be placed

Figure 10.9: Tom Cruise moving windows around on a holographic display in the 2002 movie Minority Report. It is a great-looking interaction mechanism for Hollywood, but it is terribly tiring in reality. The user would quickly experience *gorilla arms*.

so that a selection could be made around a corner. Chapter 5 of [20] offers many other suggestions.

With a pointer, the user simply illuminates the object of interest and presses a button. If the goal is to retrieve the object, then it can be immediately placed in the user's virtual hand or inventory. If the goal is to manipulate the object in a standard, repetitive way, then pressing the button could cause a virtual motor program to be executed. This could be used, for example, to turn a doorknob, thereby opening a door. In uses such as this, developers might want to set a limit on the depth of the laser pointer, so that the user must be standing close enough to enable the interaction. It might seem inappropriate, for example, to turn doorknobs from across the room!

If the object is hard to see, then the selection process may be complicated. It might be behind the user's head, which might require uncomfortable turning. The object could be so small or far away that it occupies only a few pixels on the screen, making it difficult to precisely select it. The problem gets significantly worse if there is substantial clutter around the object of interest, particularly if other selectable objects are nearby. Finally, the object may be partially or totally occluded from view.

Manipulation If the user carries an object over a long distance, then it is not necessary for her to squeeze or clutch the controller; this would yield unnecessary fatigue. In some cases, the user might be expected to carefully inspect the object while having it in possession. For example, he might want to move it around in his hand to determine its 3D structure. The object orientation could be set to follow exactly the 3D orientation of a controller that the user holds. The user could

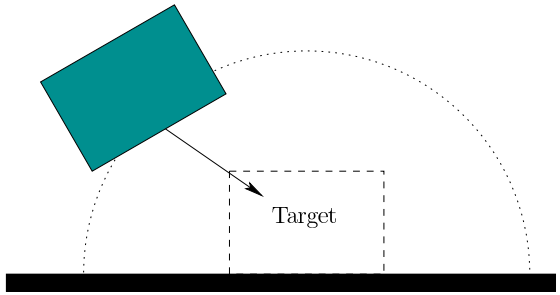


Figure 10.10: To make life easier on the user, a *basin of attraction* can be defined around an object so that when the basin in entered, the dropped object is attracted directly to the target pose.

even hold a real object in hand that is tracked by external cameras, but have a different appearance in the virtual world. This enables familiar force feedback to the user, a concept that is revisited in Section 13.1. Note that an object could even be manipulated directly in its original place in the virtual world, without bringing it close to the user's virtual body [19]. In this case, the virtual hand is brought to the object, while the physical hand remains in place. Having a longer arm than normal can also be simulated [112], to retrieve and place objects over greater distances.

Placement Now consider ungrasping the object and placing it into the world. An easy case for the user is to press a button and have the object simply fall into the right place. This is accomplished by a *basin of attraction* which is an attractive potential defined in a neighborhood of the target pose (position and orientation); see Figure 10.10. The minimum of the potential function is at the target. After the object is released, the object falls into the target pose by moving so that the potential is reduced to its minimum. This behavior is seen in many 2D drawing programs so that the endpoints of line segments conveniently meet. An example of convenient object placement is in the 2011 Minecraft sandbox game by Markus Persson (Notch), in which building blocks simply fall into place. Children have built millions of virtual worlds in this way.

Alternatively, the user may be be required to delicately place the object. Perhaps the application involves stacking and balancing objects as high as possible. In this case, the precision requirements would be very high, placing a burden on both the controller tracking system and the user.

Remapping Now consider the power of remapping, as described in Section 10.1. The simplest case is the use of the button to select, grasp, and place objects. Instead of a button, continuous motions could be generated by the user and tracked by systems. Examples include turning a knob, moving a slider bar, moving a finger

(a) (b)

Figure 10.11: (a) A pair of hand-held controllers that came with the HTC Vive headset in 2016; the device includes side buttons, a trigger, and a touch pad for the thumb. (b) A user trying the controllers (prototype version).

over a touch screen, and moving a free-floating body through space. Recall that one of the most important aspects of remapping is easy learnability. Reducing the number of degrees of freedom that are remapped will generally ease the learning process. To avoid gorilla arms and related problems, a scaling factor could be imposed on the tracked device so that a small amount of position change in the controller corresponds to a large motion in the virtual world. Note that this might have an adverse effect on precision in the virtual world. In some settings orientation scaling might also be desirable. In this case, the 3D angular velocity $(\omega_x, \omega_y, \omega_z)$ could be scaled by a factor to induce more rotation in the virtual world than in the real world.

Current systems The development of interaction mechanisms for manipulation remains one of the greatest challenges for VR. Current generation consumer VR headsets either leverage existing game controllers, as in the bundling of the XBox 360 controller with the Oculus Rift in 2016, or introduce systems that assume large hand motions are the norm, as in the HTC Vive headset controller, as shown in Figure 10.11. Controllers that have users moving their hands through space seem not too far from the Minority Report interaction mechanism shown in Figure 10.9. Others are developing gesturing systems that involve no hardware in the hands, as in the Leap Motion system that was shown in Figure 9.25 from Section 9.4. These are perhaps updated versions of the vision of "goggles and gloves" that was popular in the 1990s (recall Figure 1.26(c) from Section 1.3). Rapid evolution of methods and technologies for manipulation can be expected in the coming years, with increasing emphasis on user comfort and ease of use.

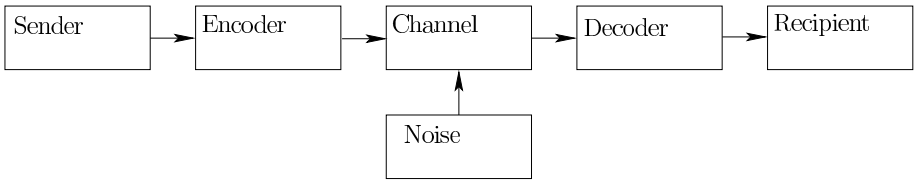


Figure 10.12: The classical Shannon-Weaver model of communication (from 1948). The sender provides a message to the encoder, which transmits the message through a channel corrupted by noise. At the other end, a decoder converts the message into a suitable format for the receiver. This model serves as the basis of communication theory in engineering.

10.4 Social Interaction

Communication and social interaction are vast subjects that extend well outside of the scope of this book. Furthermore, social interaction in VR remains in a stage of infancy, with substantial experimentation and rethinking of paradigms occuring. Nevertheless, connecting humans together is one of the greatest potentials for VR technology. Although it might seem isolating to put displays between ourselves and the world around us, we can also be brought closer together through successful interaction mechanisms. This section highlights several interesting issues with regard to social interaction, rather than provide a complete review.

Beyond Shannon-Weaver communication An important factor is how many people will be interacting through the medium. Start with a pair of people. One of the most powerful mathematical models ever developed is the *Shannon-Weaver model of communication*, which for decades has been the basis of design for communication systems in engineering; see Figure 10.12. The model involves a *sender* and a *recipient*. The communication system *encodes* a message from the sender, which is then sent over a noisy *channel*. At the other end, the system *decodes* the message and it arrives to the recipient. The recipient could give *feedback* to indicate whether the message has been received intact. This communication model gave rise to the field of *information theory*, which enabled a well-defined notion of *bandwidth* for a communication channel and revealed the limits of data compression.

This model is powerful in that it mathematically quantifies human interaction, but it is also inadequate for covering the kinds of interactions that are possible in VR. By once again following the universal simulation principle, any kind of human interaction that exists in the real world could be brought into VR. The Shannon-Weaver model is inspired by interaction mechanisms such as the 19th century telegraph or 20th century *handheld receiver* (or *walkie-talkie*). In these cases, the humans are completely isolated from each other, and the technology provides a burst of information that is similar to writing a letter. We have gone

Figure 10.13: A collection of starter avatars offered by Second Life.

from text to audio to video communication, and could extend even further by incorporating displays for other senses, such as touch and smell. There are also so many opportunities to use synthetic models, possibly in combination with actual captured information from cameras and microphones. Simple gestures and mannerisms can provide subtle but important components of interaction but are not nicely captured by the classical communication model.

In spite of its shortcomings for VR, it is important to keep in mind that the Shannon-Weaver model provides powerful analysis of bandwidth and latency for computer networks and systems, which ultimately support any form of social interaction. Therefore, it has far reaching implications on what can or cannot be accomplished in a VR system. This occurs because all "communication" is converted into streams of bits that are sent through cables or network connections. One key problem is to ensure that the targeted social interaction VR experience is comfortable, convincing, and reliably supported over the computer network.

From avatars to visual capture How should others see you in VR? This is one of the most intriguing questions because it depends on both the social context and on the technological limitations. A clear spectrum of possibilities exists. At one extreme, a user may represent himself through an *avatar*, which is a 3D representation that might not correspond at all to his visible, audible, and behavioral characteristics; see Figure 10.13. At the other extreme, a user might be captured using imaging technology and reproduced in the virtual world with a highly accurate volumetric representation; see Figure 10.14. In this case, it may seem as if the person were teleported directly from the real world to the virtual world. Many other possibilities exist along this spectrum, and it is worth

Figure 10.14: Holographic communication research from Microsoft in 2016. A volumetric representation of a person is extracted in real time and superimposed in the world, as seen through augmented reality glasses (Hololens).

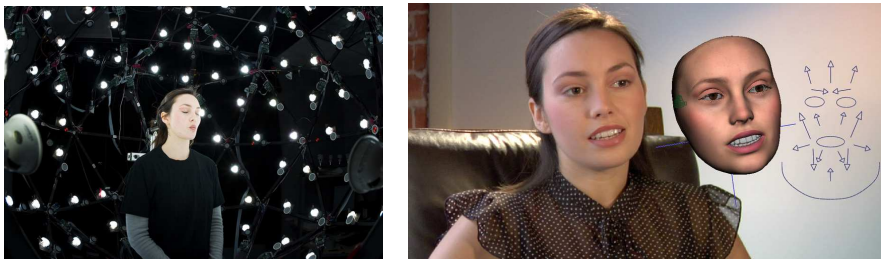
considering the tradeoffs.

One major appeal of an avatar is anonymity, which offers the chance to play a different role or exhibit different personality traits in a social setting. In a phenomenon called the *Proteus effect*, it has been observed that a person's behavior changes based on the virtual characteristics of the avatar, which is similar to the way in which people have been known to behave differently when wearing a uniform or costume [163]. The user might want to live a fantasy, or try to see the world from a different perspective. For example, people might develop a sense of empathy if they are able to experience the world from an avatar that appears to be different in terms of race, gender, height, weight, age, and so on.

Users may also want to experiment with other forms of embodiment. For example, a group of children might want to inhabit the bodies of animals while talking and moving about. Imagine if you could have people perceive you as if you as an alien, an insect, an automobile, or even as a talking block of cheese. People were delightfully surprised in 1986 when Pixar brought a desk lamp to life in the animated short Luxo Jr. Hollywood movies over the past decades have been filled with animated characters, and we have the opportunity to embody some of them while inhabiting a virtual world!

Now consider moving toward physical realism. Based on the current technology, three major kinds of similarity can be independently considered:

1. **Visual appearance:** How close does the avatar seem to the actual person in terms of visible characteristics?



(a) (b)

Figure 10.15: The Digital Emily project from 2009: (a) A real person is imaged. (b) Geometric models are animated along with sophisticated rendering techniques to produce realistic facial movement.

2. **Auditory appearance:** How much does the sound coming from the avatar match the voice, language, and speech patterns of the person?

3. **Behavioral appearance:** How closely do the avatar's motions match the body language, gait, facial expressions, and other motions of the person?

The first kind of similarity could start to match the person making a kinematic model in the virtual world (recall Section 9.4) that corresponds in size and mobility to the actual person. Other simple matching such as hair color, skin tone, and eye color could be performed. To further improve realism, texture mapping could be used to map skin and clothes onto the avatar. For example, a picture of the user's face could be texture mapped onto the avatar face. Highly accurate matching might also be made by constructing synthetic models, or combining information from both imaging and sythetic sources. Some of the best synthetic matching performed to date has been by researchers at the USC Institute for Creative Technologies; see Figure 10.15. A frustrating problem, as mentioned in Section 1.1, is the uncanny valley. People often describe computer-generated animation that tends toward human realism as seeing zombies or talking cadavers. Thus, being far from perfectly matched is usually much better than "almost" matched in terms of visual appearance.

For the auditory part, users of Second Life and similar systems have preferred text messaging. This interaction is treated as if they were talking aloud, in the sense that text messages can only be seen by avatars that would have been close enough to hear it at the same distance in the real world. Texting helps to ensure anonymity. Recording and reproducing voice is simple in VR, making it much simpler to match auditory appearance than visual appearance. One must take care to render the audio with proper localization, so that it appears to others to be coming from the mouth of the avatar; see Chapter 11. If desired, anonymity can

Figure 10.16: Oculus Social Alpha, which was an application for Samsung Gear VR. Multiple users could meet in a virtual world and socialize. In this case, they are watching a movie together in a theater. Their head movements are provided using head tracking data. They are also able to talk to each other with localized audio.

be easily preserved in spite of audio recording by using real-time voice-changing software (such as MorphVOX or Voxal Voice Changer); this might be preferred to texting in some settings.

Finally, note that the behavioral experience could be matched perfectly, while the avatar has a completely different visual appearance. This is the main motivation for motion capture systems, in which the movements of a real actor are recorded and then used to animate an avatar in a motion picture. Note that movie production is usually a long, off-line process. Accurate, real-time performance that perfectly matches the visual and behavioral appearance of a person is currently unattainable in low-cost VR systems. Furthermore, capturing the user's face is difficult if part of it is covered by a headset.

On the other hand, current tracking systems can be leveraged to provide accurately matched behavioral appearance in some instances. For example, head tracking can be directly linked to the avatar head so that others can know where the head is turned. Users can also understand head nods or gestures, such as "yes" or "no". Figure 10.16 shows a simple VR experience in which friends can watch a movie together while being represented by avatar heads that are tracked (they can also talk to each other). In some systems, eye tracking could also be used so that users can also see where the the avatar is looking; however, in some cases, this might enter back into the uncanny valley. If the hands are tracked, which could be done using controllers such as those shown in Figure 10.11, then they can also

be brought into the virtual world.

From one-on-one to societies Now consider social interaction on different scales. The vast majority of one-on-one interaction that we have in the real world is with people we know. Likewise, it is the same when interacting through technology, whether through text messaging, phone calls, or video chat. Most of our interaction though technology is targeted in that there is a specific purpose to the engagement. This suggests that VR can be used to take a video chat to the next level, where two people feel like they are face-to-face in a virtual world, or even in a panoramic capture of the real world. Note, however, that in the real world, we may casually interact simply by being in close proximity while engaged in other activities, rather than having a targeted engagement.

One important aspect of one-on-one communication is whether the relationship between the two people is *symmetrical* or *complementary* (from Paul Watzlawick's Axioms of Communication). In a symmetrical relatonship the two people are of equal status, whereas in a complementary relationship one person is in a superior position, as in the case of a boss and employee or a parent and a child. This greatly affects the style of interaction, particularly in a targeted activity.

Now consider interactions within a small group of people in the real world. Perhaps a family or coworkers are sharing a meal together. Perhaps children are together on a playground. Perhaps friends and family have gathered for a holiday or birthday celebration. To make VR versions of such interactions, it seems once again to be best suited for a targeted activity, such as gathering for a party. Perhaps you are the one who could not attend in person, but will instead "hang out" with the group through some VR interface. Perhaps there is a meeting, and a few people need to attend remotely, which is currently handled by *teleconferencing*, in which voice and video are transmitted over the network. The common scenario that is closest to VR is schoolchildren meeting in a networked video game, with some social interaction occuring while they play. They might form teams and interact through text messaging or voice while playing.

As the number of people increases to over a dozen, the case of a complemenary relationship leads to a presentation or interview. Some examples are a teacher lecturing to a class of students, and a politician speaking in front of a group of reporters. In these interactions, a leader has been clearly assigned to communicate with the group. These settings could be reproduced in VR by allowing people to attend through panoramic video capture. Alternatively, the entire event could take place in a virtual world. In the case of a symmetrical relationship, people might mingle at a large reception, and carry on conversations in small groups. This could also be reproduced in VR.

In the limiting case, an online community may emerge, which could connect millions of users. Several examples were given in Section 1.3, including MMORPGs and Second Life. People may have casual interactions by bumping into each other while spending a significant amount of time living or working in a networked virtual world. One issue, which exists in any online community, is membership. Are they

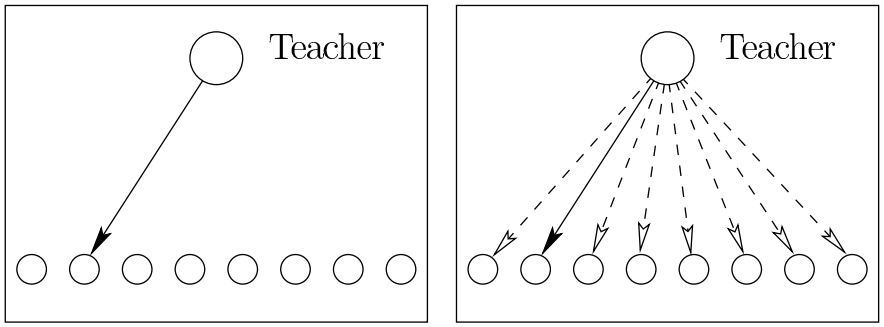


Figure 10.17: (a) A top-down depiction of an ordinary classrom is shown, in which a teacher can look drectly at one student. (b) In a VR classroom, the teacher could be looking at each student simultaneously, at least from the perspective of each student.

open to everyone, or only a closed group?

Transformed social interaction Two common themes in this book have been that VR can produce experiences that are better than reality, and that our perceptual systems adapt to new stimuli. It is therefore natural to wonder how social interaction can be altered or improved through VR. The notion of *transformed social interaction* has been introduced Jeremy Bailenson [11], to make precisely this point. A thought-provoking example is shown in Figure 10.17. In a virtual world, a teacher could look at every student simultaneously, directly in the eyes, while lecturing to the class. This is physically impossible in the real world, but it is easy to make in VR because each student could see a different version of the virtual world. Of course, the students might reason that the teacher could not possibly be paying attention to *all* of them, but the chance that she *might* be watching could have a significant effect on learning outcomes. The classroom could also appear to have a small number of students, while in reality thousands of students are in attendance. How many more mechanisms for social interaction can be introduced that are impossible to acheive in the real world? How quickly will our brains adapt to them? In what settings would be prefer such interaction to meeting in the real world? The future should bring about many exciting new mechanisms.

10.5 Additional Interaction Mechanisms

This chapter has covered three families of interaction mechanisms: locomotion, manipulation, and social. These families emerged from decades of research and

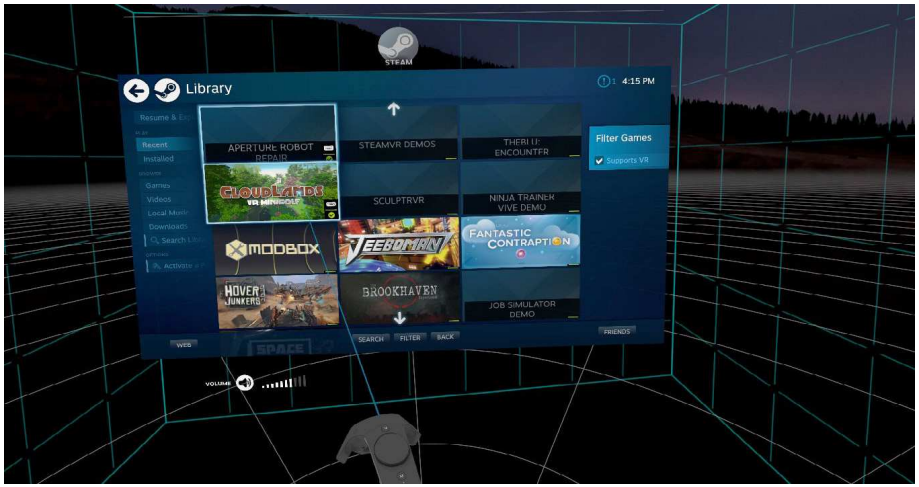


Figure 10.18: The Valve Steam game app store when viewed in the HTC Vive headset.

development, but do not completely cover every kind of interaction. Many systems demand a custom interaction mechanism be constructed that does not fall into the three families. Furthermore, with the widespread current use of low-cost VR systems, we expect that new families will emerge. A few examples of other interaction mechanisms and associated challenges are presented here.

Interaction with information and media The content of the Internet can be brought into VR in numerous ways by following the universal simulation principle. Figure 1.6 from Section 1.2 showed a movie screen in a virtual movie theater. In this case, simple interaction may be needed to pause or change the movie. As a more complex example, a web browser could appear on a public display in the virtual world or on any other device that is familiar to users in the real world. Alternatively, a virtual screen may float directly in front of the user, while a stable, familiar background is provided; see Figure 10.18.

For decades, people have interacted with the their computers and web browsers using two input devices, one for typing and the other for pointing. In the case of a PC, this has taken the form of a keyboard and mouse. With modern smartphones, people are expected to type on small touch screens, or use alternatives such as voice or swipe-to-type. They use their fingers to point by touching, and additionally zoom with a pair of fingers.

Text entry and editing The typing options on a smartphone are sufficient for entering search terms or typing a brief message, but they are woefully inadequate for writing a novel. For professionals who current sit in front of keyboards to write

reports, computer programs, newspaper articles, and so on, what kind of interfaces are needed to entice them to work in VR?

One option is to track a real keyboard and mouse, making them visible VR. Tracking of fingertips may also be needed to provide visual feedback. This enables a system to be developed that magically transforms the desk and surrounding environment into anything. Much like the use of a background image on a desktop system, the relaxing panoramic image or video could envelop the user while she works. For the actual work part, rather than having one screen in front of the user, a number of screens or windows could appear all around and at different depths.

It is easy to borrow interface concepts from existing desktop windowing systems, but much research remains to design and evaluate completely novel interfaces for improved productivity and comfort while writing. What could word processing look like in VR? What could an integrated development environment (IDE) for writing and debugging software look like? If the keyboard and mouse are replaced by other interfaces, then the user might not even need to sit at a desk to work. One challenge would be to get users to learn a method that offers text entry speeds that are comparable to a using keyboard, but enables them to work more comfortably.

3D design and visualization What are the professional benefits to being able to inhabit a 3D virtual world? In addition to video games, several other fields have motivated the development of computer graphics. Prior to *computer-aided design* (*CAD*), architects and engineers spent many hours with pencil and paper to painstakingly draw accurate lines on paper. The computer has proved to be an indispensable tool for design. Data visualization has been a key use of computers over the past years. Examples are medical, scientific, and market data. With all of these uses, we are still forced to view designs and data sets by manipulating 2D projections on screens.

VR offers the ability to interact with and view 3D versions of a design or data set. This could be from the outside looking in, perhaps at the design of a new kitchen utensil. It could also be from the inside looking out, perhaps at the design of a new kitchen. If the perceptual concepts from Chapter 6 are carefully addressed, then the difference between the designed object or environment and the real one may be less than ever before. Viewing a design in VR can be considered as a kind of *virtual prototyping*, before a physical prototype is constructed. This enables rapid, low-cost advances in product development cycles.

A fundamental challenge to achieving VR-based design and visualization is the interaction mechanism. What will allow an architect, artist, game developer, movie set builder, or engineer to comfortably build 3D worlds over long periods of time? What tools will allow people to manipulate high-dimensional data sets as they project onto a 3D world?

The future Many more forms of interaction can be imagined, even by just applying the universal simulation principle. Video games have already provided many ideas for interaction via a standard game controller. Beyond that, the Nintendo

Wii remote has been especially effective in making virtual versions of sports activities such as bowling a ball or swinging a tennis racket. What new interaction mechanisms will be comfortable and effective for VR? If more displays are presented to other senses, then even more possibilities emerge. For example, could you give someone a meaningful hug on the other side of the world if they are wearing a suit that applies the appropriate forces to the body?

Further Reading

Human Motor Control book: [122]

For more on locomotion and wayfinding: Chapters 6 and 7 of [20], and Spatial Orientation, Wayfinding, and Representation, 2015, Handbook of Virtual Environments, 2nd. Ed.

Go-go interaction: [112]

Maybe some grasping work from robotics?

More social references:

Ugly duckling by day, super model by night: The influence of body image on the use of virtual worlds, EP Becerra, MA Stutts

Virtual worldspast, present, and future: New directions in social computing PR Messinger, E Stroulia, K Lyons, M Bone 2009

Me, myself and I: The role of interactional context on self-presentation through avatars. A Vasalou, AN Joinson - Computers in Human Behavior, 2009 - Elsevier

Chapter 11

Audio

Chapter Status	Taken from *Virtual Reality*, S. M. LaValle
	This online chapter is not the final version! Check **http://vr.cs.uiuc.edu/** for information on the latest draft version. This draft was compiled on June 29, 2016.

Hearing is an important sense for VR and has been unfortunately neglected up until this chapter. Developers of VR systems tend to focus mainly on the vision part because it is our strongest sense; however, the audio component of VR is powerful and the technology exists to bring high fidelity audio experiences into VR. In the real world, audio is crucial to art, entertainment, and oral communication. As mentioned in Section 2.1, audio recording and reproduction can be considered as a VR experience by itself, with both a CAVE-like version (surround sound) and a headset version (wearing headphones). When combined consistently with the visual component, audio helps provide a compelling and comfortable VR experience.

Each section of this chapter is the auditory (or audio) complement to one of Chapters 4 through 7. The progression again goes from physics to physiology, and then from perception to rendering. Section 11.1 explains the physics of sound in terms of waves, propagation, and frequency analysis. Section 11.2 describes the parts of the human ear and their function. This naturally leads to auditory perception, which is the subject of Section 11.3. Section 11.4 concludes by presenting auditory rendering, which can produce sounds synthetically from models or reproduce captured sounds. When reading these sections, it is important to keep in mind the visual counterpart of each subject. The similarities make it easier to quickly understand and the differences lead to unusual engineering solutions.

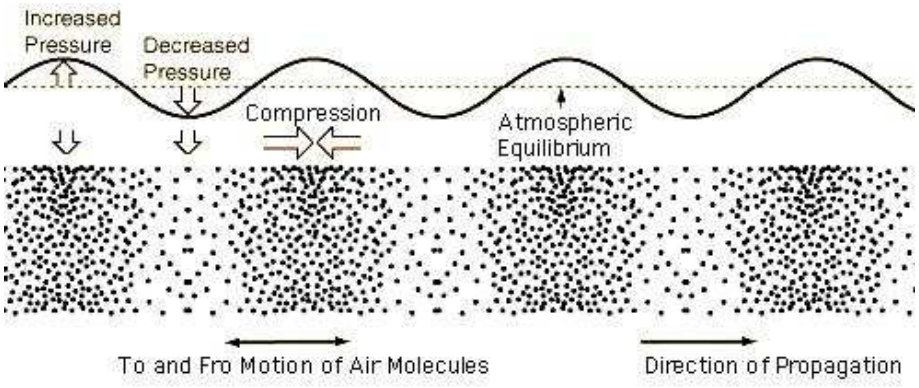


Figure 11.1: Sound is a longitudinal wave of compression and rarefaction of air molecules. The case of a pure tone is shown here, which leads to a sinusoidal pressure function. (Figure from Pond Science Institute.)

11.1 Physics of Sound

This section parallels many concepts from Chapter 4, which covered the basic physics of light. Sound wave propagation is similar in many ways to light, but with some key differences that have major perceptual and engineering consequences. Whereas light is a *transverse wave*, which oscillates in a direction perpendicular to its propagation, sound is a *longitudinal wave*, which oscillates in a direction parallel to its propagation. Figure 11.1 shows an example of this for a parallel wavefront.

Sound corresponds to vibration in a medium, which is usually air, but could also be water, or any other gases, liquids, or solids. There is no sound in a vacuum, which is unlike light propagation. For sound, the molecules in the medium displace, causing variations in pressure that range from a *compression* extreme to a decompressed, *rarefaction* extreme. At a fixed point in space, the pressure varies as a function of time. Most importantly, this could be the pressure variation on a human eardrum, which is converted into a perceptual experience. The sound pressure level frequently reported in *decibels* (abbreviated as *dB*), which is defined as

$$N_{db} = 20 * \log_{10}(p_e/p_r), \tag{11.1}$$

in which p_e is the pressure level of the peak compression and p_r is a reference pressure level, which is usually taken as 2×10^{-7} newtons / square meter.

Sound waves are typically produced by vibrating solid materials, especially as they collide or interact with each other. A simple example is striking a large bell, which causes it to vibrate for many seconds. Materials may also be forced into sound vibration by sufficient air flow, as in the case of a flute. Human bodies are designed to produce sound by using lungs to force air through the vocal cords,

which causes them to vibrate. This enables talking, singing, screaming, and so on.

Sound sources and attenuation As in the case of light, we can consider rays, for which each *sound ray* is perpendicular to the sound propagation wavefront. A point sound source can be considered, which produces eminating rays with equal power in all directions. This also results power reduction at a quadratic rate as a function of distance from the source. Such a point source is useful for modeling, but cannot be easily achieved in the real world. Planar wavefronts can be achieved by vibrating a large, flat plate, which results in the acoustic equivalent of collimated light. An important distinction, however, is the *attenuation* of sound as it propagates through a medium. Due to energy lost in the vibration of molecules, the sound intensity increases by a constant factor (or fixed percentage) for every unit of distance from the planar source; this is an example of *exponential decay*.

Propagation speed Sound waves propagate at 343.2 meters per second through air at 20° C (68° F). For comparison, light propagation is about 874,000 times faster. We have planes and cars that can surpass the speed of sound, but are nowhere near traveling at the speed of light. This is perhaps the most important difference between sound and light for making VR systems. For example, human senses and engineered sensors easily measure differences in arrival times of sound waves, leading to stronger emphasis on temporal information.

Frequency and wavelength As in Chapter 4.1, the decomposition of waves into frequency components becomes important. For sound, the frequency is the number of compressions per second and is called *pitch*. The range is generally considered to be from 20 Hz to 20,000 Hz, which is based on human hearing, much in the same way that the frequency range for light is based on human vision. Vibrations above 20,000 Hz are called *ultrasound*, and are audible to some animals. Vibrations below 20 Hz are called *infrasound*.

Using (4.1) from Section 4.1 and the propagation speed $s = 343.2$, the wavelength of a sound wave can also be determined. At $20Hz$ the wavelength is $\lambda = 343.2/20 = 17.1$m. At 20,000 Hz, it becomes $\lambda = 17.1$mm. They are the sizes of objects in our world. This causes the waves to inferfere with objects in a complicated way that is difficult to model for making synthetic sounds in VR. By comparison, light waves are tiny, ranging from 400 nm to 700 nm.

Doppler effect The sound pressure variations described above were for a fixed receiving point. If the point is moving away from the source, then the wavefronts will arrive at a reduced frequency. For example, if the receiver moves at 42.2m/s away from the source, then the waves would seem to be traveling at only $343.2 - 42.2 = 300$ meters per second. The received frequency shifts due to the relative motion between the source and receiver. This is known as the *Doppler effect*, and

the frequency as measured at the receiver can be calculated as

$$f_r = \left(\frac{s + v_r}{s + v_s} \right) f_s, \tag{11.2}$$

in which s is the speed in the medium, v_r is the velocity of the receiver, v_s is the velocity of the source, and f_s is the frequency of the source. In our example, $s = 343.2$, $v_r = -43.2$, and $v_s = 0$. The result is that a sound source with frequency $f_s = 1000$Hz would be perceived by the receiver as having frequency $f_r \approx 876.7$. This is the reason why a siren seems to change pitch as a police car passes by. The Doppler effect also applies to light, but the effect is negligible in normal VR contexts (unless developers want to experiment with virtual time dilation, space travel, and so on).

Reflection and transmission As with light, wave propagation is strongly effected by propagation through media. Imagine a sound wave hitting an interior wall as someone yells from inside of a room. It may be helpful to think about a ray of sound approaching the wall. Due to reflection, much of the sound will bounce as if the wall were an acoustic mirror. However, some of the sound energy will penetrate the wall. Sounds propagates more quickly through most solid materials, resulting in a bending of the ray as it penetrates. This is refraction. Some of the sound escapes the far side of the wall and propagates through the air in an adjacent room, resulting in transmission. Thus, someone in the adjacent room can hear yelling. The total amount of energy contained in the sound waves before it hits the wall is split by reflection and transmission, with additional loss due to attenuation.

Diffraction Wavefronts can also bend around corners, which is called *diffraction*; see Figure 11.2. This would enable someone to hear a sound that is around the corner of a building, without relying on any reflection or transmission. More diffraction occurs for longer wavelengths; thus, a lower-pitched sound bends around corners more easily. This also explains why we are more concerned about acoustic diffraction in a room than light diffraction, although the latter is often important for lenses (recall the Fresnel lens drawback of Section 7.3).

Fourier analysis Spectral decompositions were important for characterizing light sources and reflections in Section 4.1. In the case of sound, they are even more important. A sinusoidal wave, as shown in Figure 11.3(a), corresponds to a *pure tone*, which has a single associated frequency; this is analogous to a color from the light spectrum. A more complex waveform, such the sound of a piano note, can be constructed from a combination of various pure tones. Figure 11.3(b) to 11.3(d) provides a simple example. This principle is derived from *Fourier analysis*, which enables any periodic function to be decomposed into sinusoids (pure tones in our case) by simply adding them up. Each pure tone has a particular *frequency*,

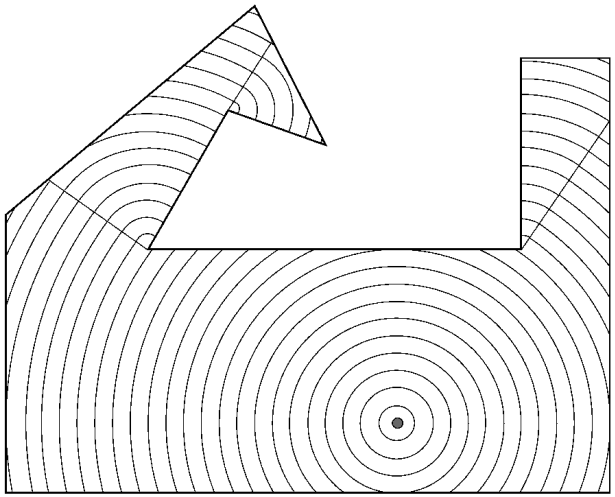


Figure 11.2: Waves can even bend around corners, due to *diffraction*. A top-down view of a room is shown. At each of the three interior corners, the propagating wavefront expands around it.

amplitude or scaling factor, and a possible timing for its peak, which is called its *phase*. By simply adding up a finite number of pure tones, virtually any useful waveform can be closely approximated. The higher-frequency, lower-amplitude sinusoids are often called *higher-order harmonics*; the largest amplitude wave is called the *fundamental frequency*. The plot of amplitude and phase as a function of frequency is obtained by applying the *Fourier transform*, which will be briefly covered in Section 11.4.

Where are the lenses? At this point, the most obvious omission in comparison to Chapter 4 is the acoustic equivalent of lenses. As stated above, refraction occurs for sound. Why is it that human ears do not focus sounds onto a spatial image in the same way as the eyes? One problem is the long wavelengths in comparison to light. Recall from Section 5.1 that the photoreceptor density in the fovea is close to the wavelength of visible light. It is likely that an "ear fovea" would have to be several meters across or more, which would makes our heads too large. Another problem is that low-frequency sound waves interact with objects in the world in a more complicated way. Thus, rather than forming an image, our ears instead work by performing Fourier analysis to sift out the structure of sound waves in terms of sinusoids of various frequencies, amplitudes, and phases. Each ear is more like a single-pixel camera operating at tens of thousands of "frames per second", rather than capturing a large image at a slower frame rate. The interesting information for sound is distributed time, whereas it is mostly distributed over space for vision.

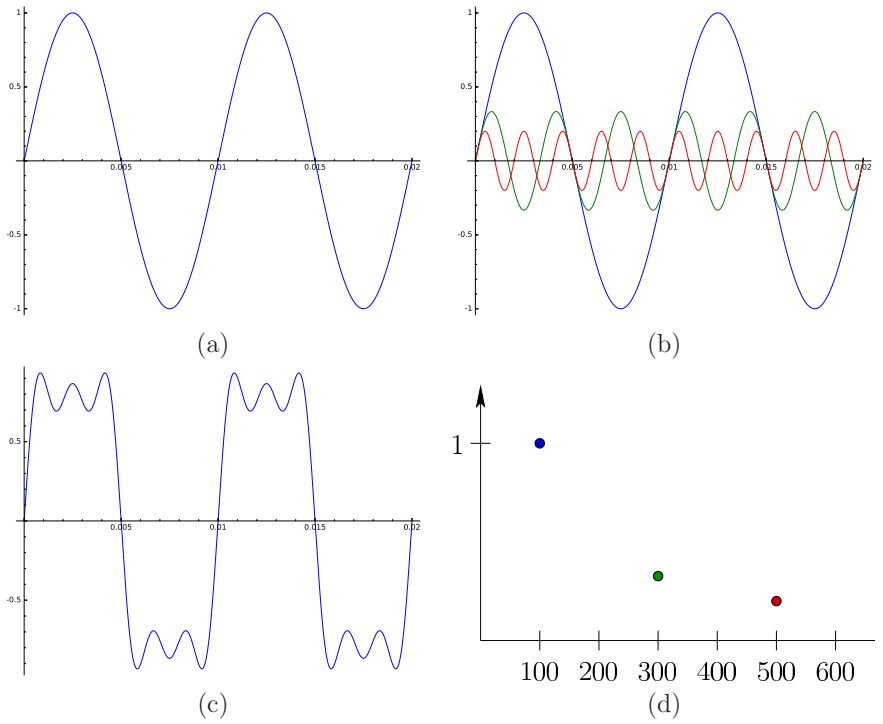


Figure 11.3: (a) A pure tone (sinusoid) of unit amplitude and frequency 100 Hz. (b) Three pure tones; in addition to the original blue, the green sinusoid has amplitude 1/3 and frequency 300 Hz, and the red one has amplitude 1/5 and frequency 500 Hz. (c) Directly adding the three pure tones approximates a square-like waveform. (d) In the frequency spectrum, there are three non-zero points, one for each pure tone.

11.2 The Physiology of Human Hearing

Our ears convert sound pressure waves into neural impulses, which ultimately lead to a perceptual experience. The anatomy of the human ear is shown in Figure 11.4. The ear is divided into outer, middle, and inner parts, based on the flow of waves. Recall from Section 5.3 the complications of eye movements. Although cats and some other animals can rotate their ears, humans cannot, which simplifies this part of the VR engineering problems.

Outer ear The floppy part of the ear that protrudes from the human head is called the *pinna*. It mainly serves as a funnel for collecting sound waves and guiding them into the *ear canal*. It has the effect of amplifying sounds in the 1500

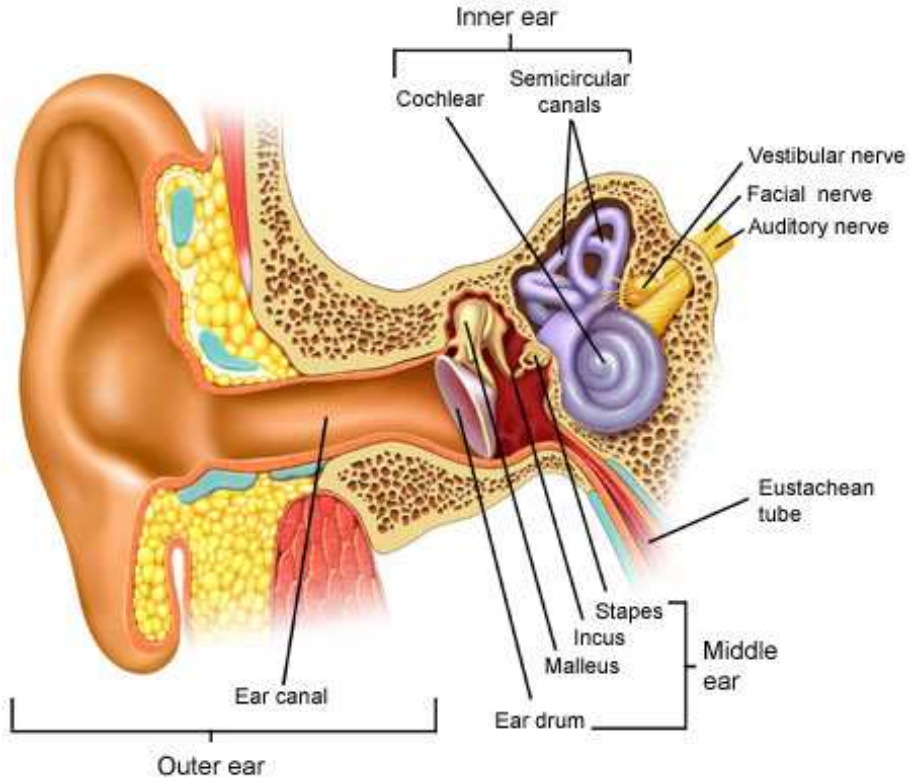


Figure 11.4: The physiology of the human auditory system.

to 7500Hz frequency range [165]. It also performs subtle filtering of the sound, causing some variation in the high-frequency range that depends on the incoming direction of the sound source. This provides a powerful cue regarding the direction of a sound source.

After traveling down the ear canal, the sound waves cause the *eardrum* to vibrate. The eardrum is a cone-shaped membrane that separates the outer ear from the middle ear. Its covers only 55mm^2 of area. It this were a camera, it would have a resolution of one pixel at this point because no additional spatial information exists other than what can be inferred from the vibrations.

Middle ear The main function of the middle ear is to convert vibrating air molecules in the outer ear into vibrating liquid in the inner ear. This is accomplished by bones that connect the eardrum to the inner ear. The air and the liquid of the inner ear have differing *impedance*, which is the resistance to vibration. The bones are called the maleus (hammer), incus (anvil), and stapes (stirrup), and

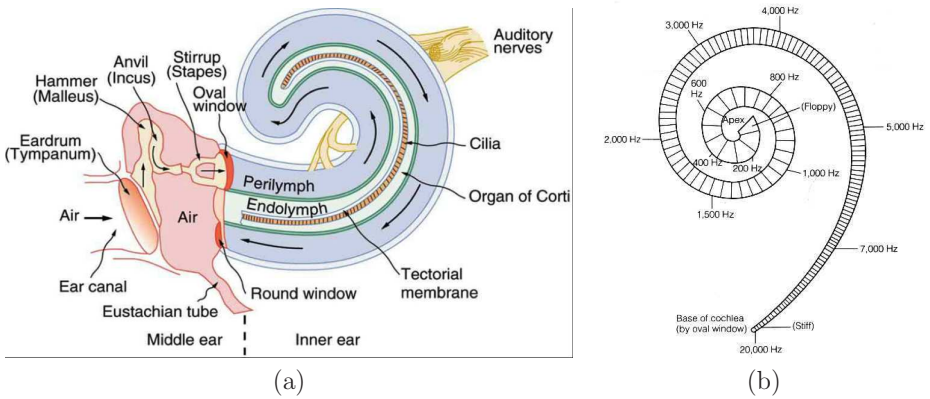


Figure 11.5: The operation of the cochlea: (a) The perilymph transmits waves that are forced by the oval window through a tube that extends the length of the cochlea and back again, to the round window. (b) Because of varying thickness and stiffness, the central spine (basilar membrane) is sensitive to particular frequencies of vibration; this causes the mechanoreceptors, and ultimately auditory perception, to be frequency sensitive.

they connected in series via muscles and ligaments that allow relative movement. The purpose of the bones is to match the impedance so that the pressure waves are transmitted to the inner ear with as little power loss as possible. This avoids the tendency of a higher impedance material to reflect the sound away. An example of this is voices reflecting over the surface of a lake, rather than being transmitted into the water.

Inner ear The inner ear contains both the vestibular organs, which were covered in Section 8.2, and the *cochlea*, which is the sense organ for hearing. The cochlea converts sound energy into neural impulses via mechanoreceptors. This is accomplished in a beautiful way that performs a spectral decomposition in the process so that the neural impulses encode amplitudes and phases of frequency components.

Figure 11.5 illustrates its operation. As seen in Figure 11.5(a), eardrum vibration is converted into oscillations of the *oval window* at the base of the cochlea. A tube that contains a liquid called *perilymph* runs from the oval window to the *round window* at the other end. The *basilar membrane* is a structure that runs through the center of the cochlea, which roughly doubles the length of the tube containing perilymph. The first part of the tube is called the *scala vestibuli*, and the second part is called the *scala tympani*. As the oval window vibrates, waves travel down the tube, which causes the basilar membrane to displace. The membrane is thin and stiff near the base (near the oval and round windows) and gradually becomes

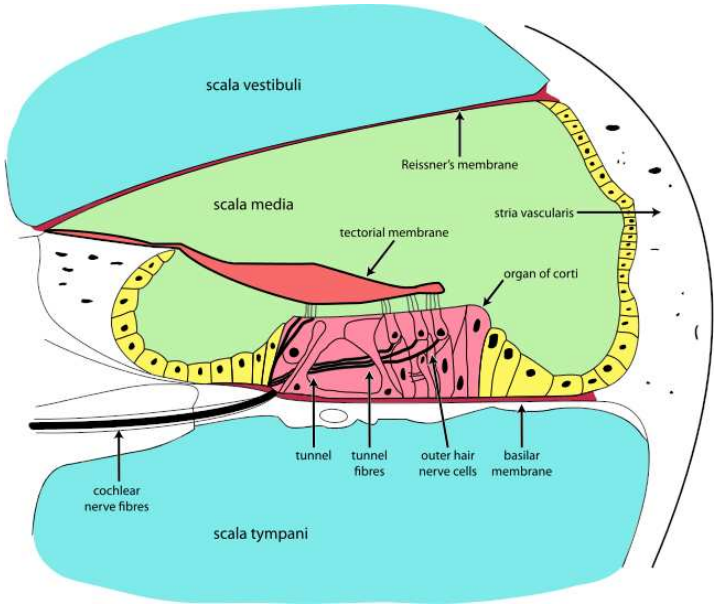


Figure 11.6: A cross section of the *organ of Corti*. The basilar and tectorial membranes move relative to each other, causing the hairs in the mechanoreceptors to bend. (Figure from multiple Wikipedia users.)

soft and floppy at the furthest away point, called the *apex*; see Figure 11.5(b). This causes each point on the membrane to vibrate only over a particular, narrow range of frequencies.

Mechanoreceptors The basilar membrane is surrounded by a larger and complicated structure called the *organ of Corti*, which additionally contains mechanoreceptors that are similar to those shown in Section 8.2. See Figure 11.6. The mechanoreceptors convert displacements of hairs into neural impulses. The hairs are displaced as the basilar membrane vibrates because the ends of some are attached to the *tectorial membrane*. The relative motions of the basilar and tectorial membranes causes a shearing action that moves the hairs. Each ear contains around 20,000 mechanoreceptors, which is considerably lower than the 100 million photoreceptors in the eye.

Spectral decomposition By exploiting the frequency-based sensitivity of the basilar membrane, the brain effectively has access to a spectral decomposition of the incoming sound waves. It is similar to, but not exactly the same as, the Fourier decomposition which discussed in Section 11.1. Several differences are mentioned in Chapter 4 of [92]. If pure tones at two different frequencies are presented to the

Figure 11.7: Due to the *precedence effect*, an auditory illusion occurs if the head is placed between stereo speakers so that one is much closer than the other. If they output the same sound at the same time, then the person perceives the sound arriving from the closer speaker, rather than perceiving an echo.

ear, the basilar membrane produces a third tone, which is sometimes audible [68]. Also, the neural impulses that result from mechoreceptor output are not linearly proportional to the frequency amplitude. Furthermore, the detection one of tone may cause detections of nearby tones (in terms of frequency) to be inhibited [126], much like lateral inhibition in horizontal cells (recall from Section 5.2). Section 11.4.1 will clarify how these differences make it more complex in terms of filtering.

Auditory pathways The neural pulses are routed from the left and right cochleae up to the highest level, which is the *primary auditory cortex* in the brain. As usual, hierarchical processing occurs as the signals are combined through neural structures. This enables multiple frequencies and phase shifts to be analyzed. An early structure called the *superior olive* receives signals from both ears so that differences in amplitude and phase can be processed. This will become important in Section 11.3 for determining the location of an audio source. At the highest level, the primary auditory cortex is mapped out *tonotopically* (locations are based on frequency), must in the same way as topographic mapping of the visual cortex.

11.3 Auditory Perception

Now that we have seen the hardware for hearing, the next part is to understand how we perceive sound. In the visual case, we saw that perceptual experiences are often surprising because they are based on adaptation, missing data, assumptions filled in by neural structures, and many other factors. The same is true for auditory experiences. Furthermore, *auditory illusions* exist in the same way as optical illusions. The McGurk effect from Section 6.4 was an example that used vision to induce incorrect auditory perception.

Precedence effect A more common auditory illusion is the *precedence effect*, in which only one sound is perceived if two nearly identical sounds arrive at slightly different times; see Figure 11.7. Sounds often reflect from surfaces, causing *reverberation*, which is the delayed arrival at the ears of many "copies" of the sound

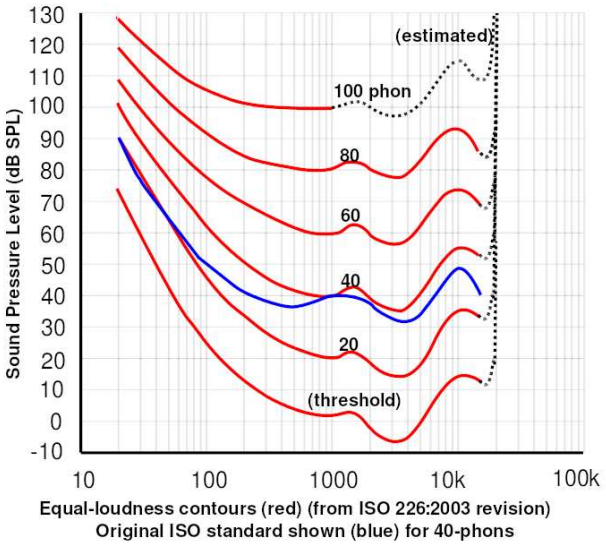


Figure 11.8: Contours of equal loudness perception as a function of frequency.

due to the different propagation paths that were taken from reflections, transmissions, and diffraction. Rather than hearing a jumble, people perceive a single sound. This is based on the first arrival, which usually has the largest amplitude. An echo is perceived if the timing difference is larger than the *echo threshold* (in one study it ranged from 3 to 61ms [162]). Other auditory illusions involve incorrect localization (*Franssen effect* and *Glissando illusion* [35]), illusory continuity of tones [154], and forever increasing tones (*Shepard tone illusion* [130]).

Psychoacoustics and loudness perception The area of psychophysics, which was introduced in Section 2.3, becomes specialized to *psychoacoustics* for the case of auditory perception. Stevens' law of perceived stimulus magnitude and Weber's law of just noticeable differences (JNDs) appear throughout the subject. For example, the exponent for Stevens law (recall (2.1)), for perceived loudness of a 3000 Hz pure tone is $x = 0.67$ [142]. This roughly means that if a sound increases to a much higher pressure level, we perceive it as only a bit louder. A more complicated example from psychacoustics is shown in Figure 11.8, which are contours that correspond to equal loudness perception as a function of frequency. In other words, as the frequency varies, at what levels are the sounds perceived to be the same loudness? This requires careful design of experiments with human subjects, a problem that is common throughout VR development as well; see Section 12.3.

Pitch perception When considering perception, the frequency of a sound wave is referred to as *pitch*. Perceptual psychologists have studied the ability of people to detect a targeted pitch in spite of confusion from sounds consisting of other wavelengths and phases. One fundamental observation is that the auditory perception system performs *critical band masking* to effectively block out waves that have frequencies outside of a particular range of interest. Another well-studied problem is the perception of differences in pitch (or frequency). In other words, for a pure tone at 1000 Hz, could you distinguish it from a tone at 1010 Hz? This is an example of JND. It turns out that for frequencies below 1000 Hz, humans can detect a change of frequency that is less than 1 Hz. The discrimination ability decreases as the frequency increases. At 10,000 Hz, the JND is about 100 Hz. In terms of percentages, this means that pitch perception is better than a 0.1% difference at low frequencies, but increases to 1.0% for higher frequencies.

Also regarding pitch perception, surprising auditory illusion occurs when the fundamental frequency is removed from a complex waveform. Recall from Figure 11.3 that a square wave can be approximately represented by adding sinusoids of smaller and smaller amplitudes, but higher frequencies. It turns out that people perceive the tone of the fundamental frequency, even when it is removed, and only the higher-order harmonics remain; several theories for this are summarized in Chapter 5 of [92].

Localization One of the main areas of psychoacoustics is *localization*, which means estimating the location of a sound source by hearing it. This is crucial for many VR experiences. For example, if people are socializing, then their voices should seem to come from the mouths of representations of people in VR. In other words, the auditory and visual cues should match. Any kind of sound effect, such as a car or zombie approaching, should also have matched cues.

The JND concept is applied for localization to obtain the *minimum audible angle* (*MAA*), which is the minimum amount of angular variation that can be detected by a human listener. A spherical coordinate system is usually used for localization, in which the listener's head is at the origin; see Figure 11.9. The angle in the horizontal plane between the forward direction and the source is called the *azimuth*, which extends from -180 to 180 degrees. The angle corresponding to deviation of the source from the horizontal plane is called the *elevation*, which extends from -90 to 90 degrees. The third coordinate is the *radius* or *distance* from the origin (head center) to the source. The MAA depends on both frequency and the direction of the source. Figure 11.10 shows a plot of the MAA as a function of frequency, at several values for azimuth. The amount of variation is surprising. At some frequencies and locations, the MAA is down to 1 degree; however, at other combinations, localization is extremely bad.

Monaural cues Auditory localization is analogous to depth and scale perception for vision, which was covered in Section 6.1. Since humans have a pair of ears, localization cues can be divided into ones that use a single ear and others that

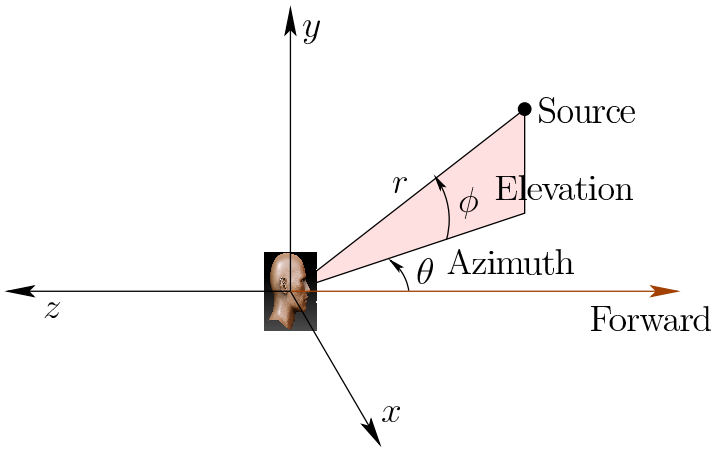


Figure 11.9: Spherical coordinates are used for the source point coordinates in auditory localization. Suppose the head is centered on the origin and facing in the $-z$ direction. The *azimuth* θ is the angle with respect to the forward direction after projecting the source into the xz plane. The *elevation* ϕ is the interior angle formed by a vertical triangle that connects the origin to the source and to the projection of the source into the plane. The radius r is the distance from the origin to the source.

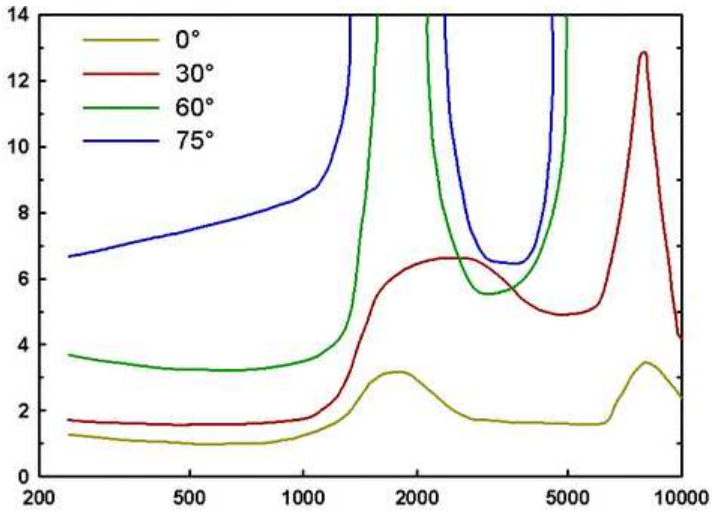


Figure 11.10: Plots of the *minimum audible angle* (*MAA*) as a function of frequency. Each plot corresponds to a different azimuth angle.

require both ears. This is analogous to monocular and binocular cues for vision. A *monaural cue* relies on sounds reaching a single ear to constrain the set of possible sound sources. Several monaural cues will now be given [166]:

1. The pinna is shaped asymmetrically so that incoming sound is distorted in a way that depends on the direction from which it arrives, especially the elevation. Although we are not consciously aware of this distortion, our auditory complex uses it for localization.

2. The amplitude of a sound decreases quadratically with distance. If it is a familiar sound, then its distance can be estimated from the perceived amplitude. Familiarity affects the power of this cue in the same way that familiarity with an object allows depth and scale perception to be separated.

3. For distant sounds, a distortion of the frequency spectrum occurs because higher-frequency components attenuate more quickly that low-frequency components. For example, distant thunder is perceived as a deep rumble, but nearby thunder includes a higher-pitched popping sound.

4. Finally, a powerful monaural cue is provided by the reverberations entering the ear as the sounds bounce around; this is especially strong in a room. Even though the precedence effect prevents us perceiving these reverberations, the brain nevertheless uses the information for localization. This cue alone is called *echolocation*, which is used naturally by some animals, including bats. It is also used by some people by making clicking sounds or other sharp noises; this allows *acoustic wayfinding* for blind people.

Binaural cues If both ears become involved, then a *binaural cue* for localization results. The simplest case is the *interaural level difference* (*ILD*), which is the difference in sound magnitude as heard by each ear. For example, one ear may be facing a sound source, while the other is in the *acoustic shadow* (the shadow caused by an object in front of a sound source is similar the shadow from a light source). The first ear would receive a much stronger vibration than the other.

Another binaural cue is *interaural time difference* (*ITD*), which is closely related to the TDOA sensing approach described in Section 9.3. The distance between the two ears is approximately 21.5cm, which results in different arrival times of the sound from a source source. Note that sound travels 21.5cm in about 0.6ms, which means that surprisingly small differences are used for localization.

Suppose that the brain measures the difference in arrival times as 0.3ms. What is the set of possible possible places where the source could have originated? This can be solved by setting up algebraic equations, which results in a conical surface known as a hyperboloid. If it is not known which sound came first, then the set of possible places is a hyperboloid of two disjoint sheets. Since the brain knows which one came first, the two sheets are narrowed down to one hyperbolid sheet, which is called the *cone of confusion*; see Figure 11.11 (in most cases, it approximately

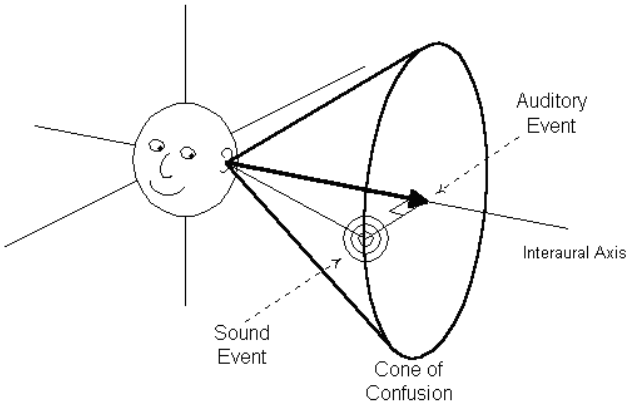


Figure 11.11: The *cone of confusion* is the set of locatons where a point source might lie after using the ITD binaural cue. It is technically a hyberboloid, but approximately looks like a cone.

looks like a cone, even though it is hyperboloid). Uncertainty within this cone can be partly resolved through by using the distortions of the pinna.

The power of motion More importantly, humans resolve much ambiguity by simply moving their heads. Just as head movement allows the powerful vision depth cue of parallax, it also provides better auditory localization. In fact, *auditory parallax* even provides another cue as nearby audio sources change their azimuth and elevation faster than distant ones. With regard to ITD, imagine having a different cone of confusion for every head pose, all within a short time. By integrating other senses, the relative head poses can be estimated, which roughly allows for an intersection of multiple cones of confusion to be made, until the sound source is precisely pinpointed. Finally, recall that the motion of a source relative to the receiver causes the Doppler effect. As in the case of vision, the issue of perceived self motion versus the motion of objects emerges based on the auditory input.

11.4 Auditory Rendering

We now arrive at the problem of producing sounds for the virtual world, and sending them to aural displays (speakers) so that the user perceives them as they were designed for the VR experience. They should be consistent with visual cues and with past auditory experiences in the real world. Whether recorded sounds, synthetic sounds, or a combination, the virtual pressure waves and their rendering to speakers should sufficiently fool the user's brain.

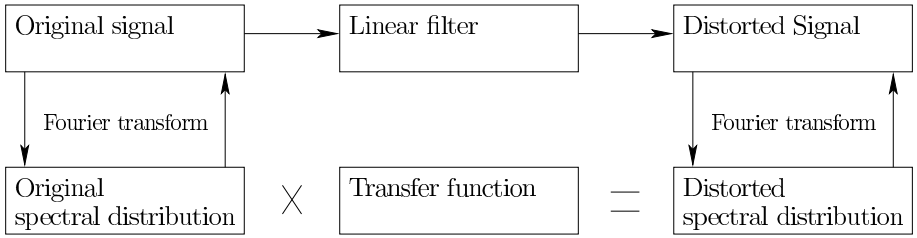


Figure 11.12: An overview of a linear filter and its relationship to Fourier analysis. The top row of blocks corresponds to the time domain, whereas the bottom row is the frequency (or spectral) domain.

11.4.1 Basic signal processing

The importance of frequency components in sound waves should be clear by now. This remains true for the engineering problem of synthesizing sounds for VR, which falls under the area of *signal processing*. A brief overview is given here; see [8, 87] for further reading. As the core of this subject is the characterization or design of *filters* that transform or distort signals. In our case the signals are sound waves that could be fully synthesized, captured using microphones, or some combination. (Recall that both synthetic and captured models exist for the visual case as well.)

Figure 11.12 shows the overall scheme, which will be presented over this section. The original signal appears in the upper left. First, follow the path from left to right. The signal enters a black box labeled *linear filter* and becomes distorted, as shown in the right. What is a linear filter? Some background concepts are needed before returning to that question.

Sampling rates Signal processing formulations exist for both *continuous-time*, which makes nice formulations and mathematical proofs, and *discrete-time*, which has an uglier appearance, but corresponds directly to the way computers process signals. Because of its practical value, we will focus on the discrete-time case.

Start with a signal as a function of time, with values represented as $x(t)$. Using digital processing, it will be sampled at regular time intervals. Let Δt be the sampling interval. The *sampling rate* or (*sampling frequency*) roughly $1/\Delta t$ Hz. For example, with 1000 Hz sampling frequency, Δt is one millisecond. According to the *Nyquist-Shannon sampling theorem*, the sampling rate should be at least two times the highest frequency component in the signal. Since the highest frequency component for audio is 20,000 Hz, this suggests that the sampling rate should be at least 40,000 Hz. By no coincidence, the sampling rate of CDs and DVDs are 44,100 Hz and 48,000 Hz, respectively.

By sampling the signal, an array of values is produced.[1] At 1000 Hz, the array

[1]The values are also discretized, and are represented using floating-point numbers. This level of discretization will be ignored.

would contain a thousand values for every second. Using an index variable k, we can refer to the kth sample as $x[k]$, which corresponds to $x(k\Delta t)$. Arbitrarily, the first sample is $x[0] = x(0)$.

Linear filters In the context of signal processing, a *filter* is a transformation that maps one signal to another. Each signal is a function of time, and the filter is like a black box that receives the one signal as input, and produces another as output. If x represents an entire signal (over all times), then let $F(x)$ represent the resulting signal after running it through the filter.

A *linear filter* is a special a special kind of filter that satisfies two algebraic properties. The first algebraic property is *additivity*, which means that if two signals are added and sent through the filter, the result should be the same as if they were each sent through the filter independently, and then the resulting transformed signals were added. Using notation, this is $F(x + x') = F(x) + F(x')$ for any two signals x and x'. For example, if two different sounds are sent into the filter, the result should be the same whether they are combined before or after the filtering. This concept will become useful as multiple sinusoids are sent through the filter.

The second algebraic property is *homogeneity*, which means that if the signal is scaled by a constant factor before being sent though the filter, the result would be the same as if it were scaled by the same factor afterwards. Using notation, this means that $cF(x) = F(cx)$ for every constant c and signal x. For example, this means that if we double the sound amplitude, the output sound from the filter doubles its amplitude as well.

A linear filter generally takes the form

$$y[k] = c_0x[k] + c_1x[k-1] + c_2x[k-2] + c_3x[k-3] + \cdots + c_nx[k-n], \quad (11.3)$$

in which each c_i is a constant, and $n+1$ is the number of samples involved in the filter. One may consider the case in which n tends to infinity, but it will not be pursued here. Not surprisingly, (11.3) is a linear equation. This particular form is a *causal filter* because the samples on the left occur no later than the sample $y[k]$. A non-causal filter would require dependency on future samples, which is reasonable for a recorded signal, but not for live sampling (the future is unpredictable!).

Here are some examples of linear filters (special cases of (11.3)). This one takes a moving average of the last three samples:

$$y[k] = \frac{1}{3}x[k] + \frac{1}{3}x[k-1] + \frac{1}{3}x[k-2]. \quad (11.4)$$

Alternatively, this is an example of *exponential smoothing* (also called *exponentially weighted moving average*):

$$y[k] = \frac{1}{2}x[k] + \frac{1}{4}x[k-1] + \frac{1}{8}x[k-2] + \frac{1}{16}x[k-3]. \quad (11.5)$$

Finite impulse response An important and useful result is that the behavior of a linear filter can be fully characterized in terms of its *finite impulse response* (*FIR*). The filter in (11.3) is often called an *FIR filter*. A *finite impulse* is a signal for which $x[0] = 1$ and $x[k] = 0$ for all $k > 0$. Any other signal can be expressed as a linear combination of time-shifted finite impulses. If a finite impulse is shifted, for example $x[2] = 1$, with $k[x] = 0$ for all other $k \neq 2$, then a linear filter produces the same result, but it is just delayed two steps later. A finite impulse can be rescaled due to filter linearity, with the output simply being rescaled. The results of sending scaled and shifted impulses through the filter are also obtained directly due to linearity.

Nonlinear filters Any causal filter that does not follow the form (11.3) is called a *nonlinear filter*. Recall from Section 11.2, that the operation of the human auditory system is almost a linear filter, but exhibits characteristics that make it into a nonlinear filter. Linear filters are preferred because of their close connection to spectral analysis, or frequency components, of the signal. Even if the human auditory system contains some nonlinear behavior, analysis based on linear filters is nevertheless valuable.

Returning to Fourier analysis Now consider the bottom part of Figure 11.12. The operation of a linear filter is easy to understand and compute in the *frequency domain*. This is the function obtained by performing the Fourier transform on the signal, which provides an amplitude for every combination of frequency and phase. This transform was briefly introduced in Section 11.1 and illustrated in Figure 11.3. Formally, it is defined for discrete-time systems as

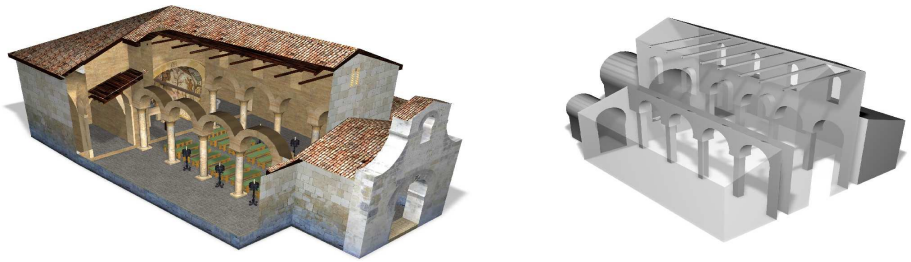
$$X(f) = \sum_{k=-\infty}^{\infty} x[k]e^{-i2\pi fk}, \quad (11.6)$$

in which $X(f)$ is the resulting spectral distribution, which is a function of the frequency f. The exponent involves $i = \sqrt{-1}$ and is related to sinusoids through Euler's formula:

$$e^{-i2\pi fk} = \cos(-2\pi fk) + i\sin(-2\pi fk). \quad (11.7)$$

Unit complex numbers are used as an algebraic trick to represent the phase. The *inverse Fourier transform* is similar in form and converts the spectral distribution back into the time domain. These calculations are quickly performed in practice by using the *Fast Fourier Transform* (*FFT*) [8, 87].

Transfer function In some cases, a linear filter is designed by expressing how it modifies the spectral distribution. It could amplify some frequencies, while suppressing others. In this case, the filter is defined in terms of a *transfer function*, which is applied as follows: 1) transforming the original signal using the Fourier transform, 2) multiplying the result by the transfer function to obtain the distorted



(a) (b)

Figure 11.13: An audio model is much simpler. (From Pelzer, Aspock, Schroder, and Vorlander, 2014, [111])

spectral distribution, and then 3) applying the inverse Fourier transform to obtain the result as a function of time. The transfer function can be calculated from the linear filter by applying the discrete Laplace transform (called *z-transform*) to the finite impulse response [8, 87].

11.4.2 Acoustic modeling

The geometric modeling concepts from Section **??** apply to the auditory side of VR, in addition to the visual side. In fact, the same models could be used for both. Walls that reflect light in the virtual world also reflect sound waves. Therefore, both could be represented by the same triangular mesh. This is fine in theory, but fine levels of detail or spatial resolution do not matter as much for audio. Due to high visual acuity, geometric models designed for visual rendering may have a high level of detail. Recall from Section 5.4 that humans can distinguish 60 stripes or more per degree of viewing angle. In the case of sound waves, small structures are essentially invisible to sound. One recommendation is that the acoustic model needs to have a spatial resolution of only 0.5m [150]. Figure 11.13 shows an example. Thus, any small corrugations, door knobs, or other fine structures can be simplified away. It remains an open challenge to automatically convert a 3D model designed for visual rendering into one optimized for auditory rendering.

Now consider a sound source in the virtual environment. This could, for example, be a "magical" point that emits sound waves or a vibrating planar surface. The equivalent of white light is called *white noise*, which in theory contains equal weight of all frequencies in the audible spectrum. Pure static from an old analog TV is an approximate example of this. In practical settings, the sound of interest has a high concentration among specific frequencies, rather than being uniformly distributed.

How does the sound interact with the surface? This is analogous to the shading problem from Section 7.1. In the case of light, diffuse and specular reflections occur with a dependency on color. In the case of sound, the same two possibilities

exist, again with a dependency on the wavelength (or equivalently, the frequency). For a large, smooth, flat surface, a specular reflection of sound waves occurs, with the outgoing angle being equal to the incoming angle. The reflected sound usually has a different amplitude and phase. The amplitude may be decreased by a constant factor due to absorption of sound into the material. The factor usually depends on the wavelength (or frequency). The back of [150] contains coefficients of absorption, given with different frequencies, for many common materials.

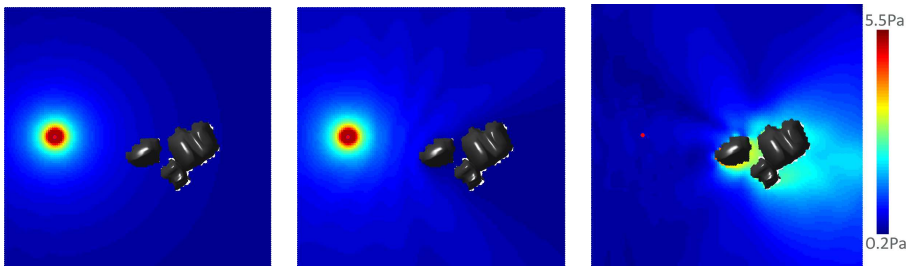
In the case of smaller objects, or surfaces with repeated structures, such as bricks or corrugations, the sound waves may scatter in a way that is difficult to characterize. This is similar to diffuse reflection of light, but the scattering pattern for sound may be hard to model and calculate. One unfortunate problem is that the scattering behavior depends on the wavelength. If the wavelength is much smaller or much larger than the size of the structure (entire object or corrugation), then the sound waves will mainly reflect. If the wavelength is close to the structure size, then significant, complicated scattering may occur.

At the extreme end of modeling burdens, a *bidirectional scattering distribution function* (*BSDF*) could be constructed. The BSDF could be estimated from equivalent materials in the real world by a combination of speaker placed in different locations and a microphone array to measure the scattering in a particular direction. This might work well for flat materials that are large with respect to the wavelength, but it will still not handle the vast variety of complicated structures and patterns that can appear on a surface.

Capturing sound Sounds could also be captured in the real world using microphones and then brought into the physical world. For example, the matched zone might contain microphones that become speakers at the equivalent poses in the real world. As in the case of video capture, making a system that fully captures the sound field is challenging. Simple but effective techniques based on interpolation of sounds captured by multiple microphones are discussed in [113].

11.4.3 Auralization

Propagation of sound in the virtual world As in visual rendering, there are two main ways to handle the propagation of waves. The most expensive way is based on simulating the physics as accurately as possible, which involves computing numerical solutions to partial differential equations that precisely model wave propagation. The cheaper way is to shoot visibility rays and characterize the dominant interactions between sound sources, surfaces, and ears. The choice between the two methods also depends on the particular setting; some systems involve both kinds of computations [94, 150]. If the waves are large relative to the objects in the environment, then numerical methods are preferred. In other words, the frequencies are low and the geometric models have a high level of detail. At higher frequencies or with larger, simpler models, visibility-based methods are preferable.



(a) (b) (c)

Figure 11.14: Computed results for sound propagation by numerically solving the Helmzoltz wave equation (taken from [94]): (a) The pressure magnitude before obstacle interaction is considered. (b) The pressure after taking into account scattering. (c) The scattering component, which is the pressure from (b) minus the pressure from (a).

Numerical wave propagation The *Helmholtz wave equation* expresses constraints at every point in $\mathbb{R}^3$ in terms of partial derivatives of the pressure function. Its frequency-dependent form is

$$\nabla^2 p + \frac{\omega^2}{s^2} p = 0, \tag{11.8}$$

in which p is the sound pressure, ∇^2 is the Laplacian operator from calculus, and ω is related to the frequency f as $\omega = 2\pi f$.

Closed-form solutions to (11.8) do not exist, except in trivial cases. Therefore, numerical computations are performed by iteratively updating values over the space; a brief survey of methods in the context of auditory rendering appears in [94]. The wave equation is defined over the obstacle-free portion of the virtual world. The edge of this space becomes complicated, leading to *boundary conditions*. One or more parts of the boundary correspond to sound sources, which can be considered as vibrating objects or obstacles that force energy into the world. At these locations, the 0 in (11.8) is replaced by a *forcing function*. At the other boundaries, the wave may undergo some combination of absorption, reflection, scattering, and diffraction. These are extremely difficult to model; see [118] for details. In some rendering applications, these boundary interactions may simplified and handled with simple *Dirichlet boundary conditions* and *Neumann boundary conditions* [164]. If the virtual world is unbounded, then an additional *Sommerfield radiation condition* is needed. For detailed models and equations for sound propagation in a variety of settings, see [118]. An example of a numerically computed sound field is shown in Figure 11.14.

Visibility-based wave propagation The alternative to numerical computations, which gradually propagate the pressure numbers through the space, is

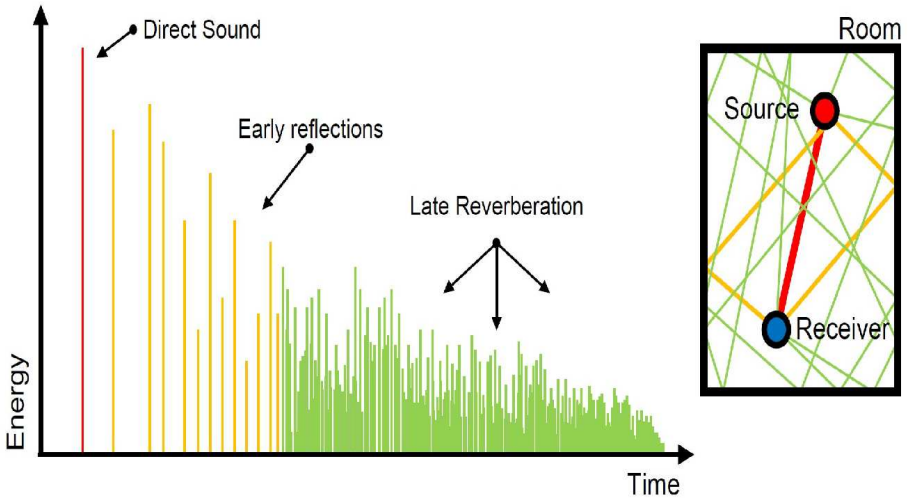


Figure 11.15: Reverberations. (From Pelzer, Aspock, Schroder, and Vorlander, 2014, [111])

visibility-based methods, which consider the paths of sound rays that emanate from the source and bounce between obstacles. The methods involve determining ray intersections with the geometric model primitives, which is analogous to ray tracing operations from Section 7.1.

It is insightful to look at the impulse response of a sound source in a virtual world. If the environment is considered as a linear filter, then the impulse response provides a complete characterization for any other sound signal [95, 111, 115]. Figure 11.15 shows the simple case of the impulse response for reflections in a rectangular room. Visibility-based methods are particularly good at simulating the reverberations, which are important to reproduce for perceptual reasons. More generally, visibility-based methods may consider rays that undergo reflection, absorption, scattering, and diffraction. Due to the high computational cost of characterizing all rays, *stochastic ray tracing* offers a practical alternative by randomly sampling rays and their interactions with materials [150]; this falls under the general family of Monte Carlo methods, which are used, for example, to approximate solutions to high-dimensional integration and optimization problems.

Entering the ear Sound that is generated in the virtual world must be transmitted to each ear in the physical world. It is as if a virtual microphone positioned in the virtual world captures the simulated sound waves. These are then converted into audio output through a speaker that is positioned in front of the ear. Recall from Section 11.3 that humans are able to localize sound sources from auditory cues. How would this occur for VR if all of the sound emanates from a fixed speaker? The ILD and ITD cues could be simulated by ensuring that each ear re-

ceives the appropriate sound magnitude and phase to that differences in amplitude and timing are correct. This implies that the physical head must be reproduced at some level of detail in the virtual world so that these differences are correctly calculated. For example, the distance between the ears and size of the head may become important.

HRTFs This solution would still be insufficient to resolve ambiguity within the cone of confusion. Recall from Section 11.3 that the pinna shape distorts sounds in a direction-dependent way. To fully take into account the pinna and other parts of the head that may distort the incoming sound, the solution is to develop a *head-related transfer function* (*HRTF*). The idea is to treat this distortion as a linear filter, which can be characterized in terms of its transfer function (recall Figure 11.12). This is accomplished by placing a human subject into an anechoic chamber and placing sound sources at different locations in the space surrounding the head. At each location, an impulse is generated on a speaker, and the impulse response is recorded with a small microphone placed inside of the ear canal of a human or dummy. The locations are selected by incrementally varying the distance, azimuth, and elevation; recall the coordinates for localization from Figure 11.10. In many cases, a *far-field approximation* may be appropriate, in which case a large value is fixed for the distance. This results in an HRTF that depends on only the azimuth and elevation.

It is, of course, impractical to build an HRTF for every user. There is significant motivation to use a single HRTF that represents the "average listener"; however, the difficulty is that it might not be sufficient in some applications because it is not designed for individual users (see Section 6.3.2 of [150]). One compromise might be to offer a small selection of HRTFs to users, to account for variation among the population, but they may be incapable of picking the one most suitable for their particular pinnae and head. Another issue is that the transfer function may depend on factors that frequently change, such as wearing a hat, putting on a jacket with a hood or large collar, or getting a haircut. Recall that adaptation occurs throughout human perception and nearly all aspects of VR. If people adapt to frequent changes in the vicinity of their heads in the real world, then perhaps they also adapt to an HRTF that is not perfect. Significant research issues remain on this topic.

Tracking issues The final challenge is to ensure that the physical and virtual ears align in the matched zone. If the user turns her head, then the sound should be adjusted accordingly. If the sound emanates from a fixed source, then it should be perceived as fixed while turning the head. This is another example of the perception of stationarity. Accordingly, tracking of the ear pose (position and orientation) is needed to determine the appropriate "viewpoint". This is equivalent to head tracking with simple position and orientation offsets for the right and left ears. As for vision, there are two choices. The head orientation alone may be tracked, with the full pose of each ear determined by a head model (recall Figure

9.8). Alternatively, the full head pose may be tracked, directly providing the pose of each ear through offset transforms. To optimize performance, user-specific parameters can provide a perfect match: The distance along the z axis from the eyes to the ears and the distance between ears. The later is analogous to the IPD, the distance between pupils for the case of vision.

Further Reading

Computational methods for linear acoustics: [146].

Auditory display overview: [151]

Anatomy, physiology, psychoacoustics, perception: [102, 165]. Also, [92], chapters 4 and 5.

Visibility-based auralization: [44].

Rendering in rooms: [134]

Book on audio localization: [13]

Thresholds before echoes are heard: [120, 162].

Monoural cues [166]

Good overview of cone of confusion and technicalities: [132]

Room acoustics: [111]

Spatial Impulse Response Rendering: [95, 115]

Convenient way to place virtual sound sources: [113].

directional audio coding: [114]

Free book on acoustics (PDE heavy): [118]

Chapter 12

Evaluating VR Systems

Chapter Status	Taken from *Virtual Reality*, S. M. LaValle
	This online chapter is not the final version! Check **http://vr.cs.uiuc.edu/** for information on the latest draft version. This draft was compiled on June 29, 2016.

Need to fit Fitts's law. Also, Card, English, and Burr (1978)

R.W. Soukoreff, I.S. MacKenzie / Int. J. Human-Computer Studies 61 (2004) 751789

HCI models, theories, and frameworks: Toward a multidisciplinary science JM Carroll - 2003

12.1 Perceptual Training

(perceptual learning)

Examples: Blind people can hear better, and deaf people can see better. Doctors and medical technicians train to read nonintuitive medical scans. Detectives train to look for clues at a crime scene. Question: What should VR engineers and developers train for?

12.2 Comfort and VR Sickness

Mention how vection and VR sickness is NOT the same as motion sickness or seasickness. Or space sickness. Make references to this from 10.1 and 8.4.

Remind about optical distortion problems.

Fusion of sensory cues: When it is harder, then more fatigue.

Adaptation.

Survey: [66]; Motion Sickness Sympyomatology and Origins, B. D. Lawson, 2015, Handbook of Virtual Environments, 2nd Ed.

Strobing (4Hz):

See work by: Tom Stoffregen, Visually induced motion sickness predicted by postural instability

Vection!

Simulator Sickness in Virtual Environments - Defense ... EM Kolasinski - 1995

Effects of Field of View on Performance with Head-Mounted Displays, PhD thesis, University of North Carolina at Chapel Hill, by K. W. Arthur - 2000

12.3 Design of Experiments

See Wolfe et al book Sensation and Perception

method of limits (page 9)

method of adjustment

Foundations of Behavioral Research (4th edition), by Kerlinger, F. N. and Lee, H. B.

Experimental Design (4rd Edition) by Kirk, R. E.

12.4 Best Practices

12.5 The Development Cycle

Chapter 13

Frontiers

Chapter Status	Taken from *Virtual Reality*, S. M. LaValle
	This online chapter is not the final version! Check **http://vr.cs.uiuc.edu/** for information on the latest draft version. This draft was compiled on June 29, 2016.

13.1 Touch

Haptic feedback

Note: Can be faked by reproducing objects from the real world in the virtual world, within the matched zone.

Vibrations, air pressure, force feedback with robot

Holding real objects. Tangible interface.

13.2 Taste and Smell

Olfactory Interfaces, D. L. Jones et al., 2015, Handbook of Virtual Environments, 2nd. Ed.

13.3 Robotic Interfaces

Telepresence

Remote control vs. higher levels of autonomy (motion planning)

13.4 Brain-Machine Interfaces

Related: Cochlear and retinal implants

David Eagleman, creating new senses

Control via reading neural signals

Ultimately: The brain in a vat, Harman. Descartes Evil Demon. Nozick Experience Machine.

Bibliography

[1] M. Abrash. Raster scan displays: More than meets the eye. Blog post. Retrieved from http://blogs.valvesoftware.com/abrash/raster-scan-displays-more-than-meets-the-eye/, January 2013. Last retrieved on Jan 10, 2016.

[2] K. Akeley, S. J. Watt, A. Reza Girschick, and M. S. Banks. A stereo display prototype with multiple focal distances. *ACM Transactions on Graphics*, 23(3), 2004.

[3] T. Akenine-Möller, E. Haines, and N. Hoffman. *Real-Time Rendering*. CRC Press, Boca Raton, FL, 2008.

[4] B. B. Andersen, L. Korbo, and B. Pakkenberg. A quantitative study of the human cerebellum with unbiased stereological techniques. *Journal of Comparative Neurology*, 326(4):549–560, 1992.

[5] J. Angeles. *Spatial Kinematic Chains. Analysis, Synthesis, and Optimisation*. Springer-Verlag, Berlin, 1982.

[6] J. Angeles. *Rotational Kinematics*. Springer-Verlag, Berlin, 1989.

[7] J. Angeles. *Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms*. Springer-Verlag, Berlin, 2003.

[8] A. Antoniou. *Digital Signal Processing: Signals, Systems, and Filters*. McGraw-Hill Education, Columbus, OH, 2005.

[9] K. J. Astrom and R. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, Princeton, NJ, 2008.

[10] F. A. Azevedo, L. R. Carvalho, L. T. Grinberg, J. M. Farfel, R. E. Ferretti, R. E. Leite, W. Jacob Filho, R. Lent, and S. Herculano-Houzel. Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *Journal of Computational Neurology*, 513:532–541, 2009.

[11] J. N. Bailenson, A. C. Beall, J. Loomis, J. Blascovich, and M. Turk. Transformed social interaction: Decoupling representation from behavior and form in collaborative virtual environments. *PRESENCE: Teleoperators and Virtual Environments*, 13(4):428–441, 2004.

[12] J. Birn. *Digital Lighting and Rendering, 3rd Ed.* New Riders, San Francisco, CA, 2013.

[13] J. Blauert. *Spatial Hearing: Psychophysics of Human Sound Localization*. MIT Press, Boston, MA, 1996.

[14] J. F. Blinn. Models of light reflection for computer synthesized pictures. In *Proceedings Annual Conference on Computer Graphics and Interactive Techniques*, 1977.

[15] I. Bogost and N. Monfort. *Racing the Beam: The Atari Video Computer System*. MIT Press, Cambridge, MA, 2009.

[16] W. M. Boothby. *An Introduction to Differentiable Manifolds and Riemannian Geometry. Revised 2nd Ed.* Academic, New York, 2003.

[17] D. Bordwell and K. Thompson. *Fild History: An Introduction, 3rd Ed.* McGraw-Hill, New York, NY, 2010.

[18] J. K. Bowmaker and H. J. A. Dartnall. Visual pigment of rods and cones in a human retina. *Journal of Physiology*, 298:501–511, 1980.

[19] D. Bowman and L. Hodges. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In *Proceedings ACM Symposium on Interactive 3D Graphics*, pages 35–38, 1997.

[20] D. A. Bowman, E. Kruijff, J. J. LaViola, and I. Poupyrev. *3D User Interfaces*. Addison-Wesley, Boston, MA, 2005.

[21] K. Brown. Silent films: What was the right speed? *Sight and Sound*, 49(3):164–167, 1980.

[22] M. Brown and D. G. Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73, 2007.

[23] D. C. Burr, M. C. Morrone, and L. M. Vaina. Large receptive fields for optic flow detection in humans. *Vision Research*, 38(12):1731–1743, 1998.

[24] E. Catmull. *A subdivision algorithm for computer display of curved surfaces*. PhD thesis, University of Utah, 1974.

[25] A. Y. Chang. A survey of geometric data structures for ray tracing. Technical Report TR-CIS-2001-06, Brooklyn Polytechnic University, 2001.

[26] G. Chen, J. A. King, N. Burgess, and J. O'Keefe. How vision and movement combine in the hippocampal place code. *Proceedings of the National Academy of Science USA*, 110(1):378–383, 2013.

[27] C. K. Chui and G. Chen. *Kalman Filtering*. Springer-Verlag, Berlin, 1991.

[28] E. Cline. *Ready Player One*. Random House, 2011.

[29] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms*. Springer-Verlag, Berlin, 1992.

[30] C. Cruz-Neira, D. J. SAndin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart. The CAVE: Audio visual experience automatic virtual environment. *Communications of the ACM*, 35(6):64–72, 1992.

[31] C. A. Curcio, K. R. Sloan, R. E. Kalina, and A. E. Hendrickson. Human photoreceptor topography. *Journal of Comparative Neurobiology*, 292:497–523, 1990.

[32] R. Darwin. New experiments on the ocular spectra of light and colours. *Philosophical Transactions of the Royal Society of London*, 76:313–348, 1786.

[33] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications, 2nd Ed.* Springer-Verlag, Berlin, 2000.

[34] J. L. Demer, J. Goldberg, H. A. Jenkins, and F. I. Porter. Vestibulo-ocular reflex during magnified vision: Adaptation to reduce visual-vestibular conflict. *Aviation, Space, and Environmental Medicine*, 58(9 Pt 2):A175–A179, 1987.

[35] D. Deutsch, T. Hamaoui, and T. Henthorn. The glissando illusion and handedness. *Neuropsychologia*, 45:2981–2988, 2007.

[36] A. T. Duchowski. *Eye Tracking Methodology: Theory and Practice, 2nd Ed.* Springer-Verlag, Berlin, 2007.

[37] R. Engbert and K. Mergenthaler. Mircosaccades are triggered by low retinal image slip. *Proceedings of the National Academy of Sciences of the United States of America*, 103(18):7192–7197, 2008.

[38] C. J. Erkelens. Coordination of smooth pursuit and saccades. *Vision Research*, 46(1–2):163–170, 2006.

[39] J. Favre, B. M. Jolles, O. Siegrist, and K. Aminian. Quaternion-based fusion of gyroscopes and accelerometers to improve 3D angle measurement. *Electronics Letters*, 32(11):612–614, 2006.

[40] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[41] R. C. Fitzpatrick and B. L. Day. Probing the human vestibular system with galvanic stimulation. *Journal of Applied Physiology*, 96(6):2301–2316, 2004.

[42] R. C. Fitzpatrick, J. Marsden, S. R. Lord, and B. L. Day. Galvanic vestibular stimulation evokes sensations of body rotation. *NeuroReport*, 13(18):2379–2383, 2002.

[43] A. K. Forsberg, K. Herndon, and R. Zelznik. Aperture based selection for immersive virtual environments. In *Proceedings ACM Symposium on User Interface Software and Technology*, pages 95–96, 1996.

[44] T. Funkhouser, I. Carlbom, G. Elko, G. Pingali, M. Sondhi, and J. West. A beam tracing approach to acoustic modeling for interactive virtual environments. In *Proceedings ACM Annual Conference on Computer Graphics and Interactive Techniques*, pages 21–32, 1998.

[45] J. Gallier. *Curves and Surfaces in Geometric Modeling*. Morgan Kaufmann, San Francisco, CA, 2000.

[46] G. M. Gauthier and D. A. Robinson. Adaptation of the human vestibuloocular reflex to magnifying lenses. *Brain Research*, 92(2):331–335, 1975.

[47] D. Gebre-Egziabher, G. Elkaim, J. David Powell, and B. Parkinson. Calibration of strapdown magnetometers in magnetic field domain. *Journal of Aerospace Engineering*, 19(2):87–102, 2006.

[48] W. Gibson. *Neuromancer*. Ace Books, 1984.

[49] W. C. Gogel. An analysis of perceptions from changes in optical size. *Perception and Psychophysics*, 60(5):805–820, 1998.

[50] A. Gopnik, A. N. Meltzoff, and P. K. Kuhl. *The Scientist in the Crib: What Early Learning Tells Us About the Mind*. HarperCollins, New York, NY, 2000.

[51] S. Gottschalk, M. C. Lin, and D. Manocha. Obbtree: A hierarchical structure for rapid interference detection. In *Proceedings ACM SIGGRAPH*, 1996.

[52] J. Gregory. *Game Engine Architecture, 2nd Ed.* CRC Press, Boca Raton, FL, 2014.

[53] B. Guentner, M. Finch, S. Drucker, D. Tan, and J. Snyder. Foveated 3D graphics. Technical report, Microsoft Research, 2012. Available at http://research.microsoft.com/.

[54] P. Guigue and O. Devillers. Fast and robust triangle-triangle overlap test using orientation predicates. *Journal of Graphics Tools*, 8(1):25–32, 2003.

[55] R. S. Hartenberg and J. Denavit. A kinematic notation for lower pair mechanisms based on matrices. *Journal of Applied Mechanics*, 77:215–221, 1955.

[56] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision, 2nd Ed.* Cambridge University Press, Cambridge, U.K., 2004.

[57] C. D. Harvey, F. Collman, D. A. Dombeck, and D. W. Tank. Intracellular dynamics of hippocampal place cells during virtual navigation. *Nature*, 461:941–946, 2009.

[58] E. G. Heckenmueller. Stabilization of the retinal image: A review of method, effects, and theory. *Psychological Bulletin*, 63:157–169, 1965.

[59] J. Heikkilä and O. Silvén. A four-step camera calibration procedure with implicit image correction. In *Proc. Computer Vision and Pattern Recognition*, pages 1106–1112, 1997.

[60] W. T. Higgins. A comparison of complementary and Kalman filtering. *IEEE Transactions on Aerospace and Electronic Systems*, 11(3):321–325, 1975.

[61] J. G. Hocking and G. S. Young. *Topology*. Dover, New York, 1988.

[62] C. M. Hoffmann. *Geometric and Solid Modeling*. Morgan Kaufmann, San Francisco, CA, 1989.

[63] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy. Visual odometry and mapping for autonomous flight using an RGB-D camera. In *Proceedings International Symposium on Robotics Research*, 2011.

[64] B. W. Jones. Apple retina display. Blog post. Available at http://prometheus.med.utah.edu/∼bwjones/2010/06/apple-retina-display/, June 2010. Last retrieved on Jan 10, 2015.

[65] R. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME, Journal of Basic Engineering*, 82:35–45, 1960.

[66] B. Keshavarz, H. Hecht, and B. D. Lawson. Visually induced motion sickness: Causes, characteristics, and countermeasures. In K. S. Hale and K. M. Stanney, editors, *Handbook of Virtual Environments, 2nd Edition*. CRC Press, Boca Raton, FL, 2015.

[67] W. Khalil and J. F. Kleinfinger. A new geometric notation for open and closed-loop robots. In *Proceedings IEEE International Conference on Robotics & Automation*, volume 3, pages 1174–1179, 1986.

[68] D. O. Kim, C. E. Molnar, and J. W. Matthews. Cochlear mechanics: Nonlinear behaviour in two-tone responses as reflected in cochlear-new-fibre responses and in ear-canal sound pressure. *Journal of the Acoustical Society of America*, 67(5):1704–1721, 1980.

[69] H. Kingma and M. Janssen. Biophysics of the vestibular system. In A. M. Bronstein, editor, *Oxford Textbook of Vertigo and Imbalance*. Oxford University Press, Oxford, UK, 2013.

[70] C. L. Kinsey. *Topology of Surfaces*. Springer-Verlag, Berlin, 1993.

[71] C. Konvalin. Compensating for tilt, hard-iron, and soft-iron effects. Available at http://www.sensorsmag.com/sensors/motion-velocity-displacement/compensating-tilt-hard-iron-and-soft-iron-effects-6475, December 2009. Last retrieved on May 30, 2016.

[72] P. R. Kumar and P. Varaiya. *Stochastic Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1986.

[73] R. Lafer-Sousa, K. L. Hermann, and B. R. Conway. Striking individual differences in color perception uncovered by the dress photograph. *Current Biology*, 25(13):R545–R546, 2015.

[74] M. F. Land and S.-E. Nilsson. *Animal Eyes*. Oxford University Press, Oxford, UK, 2002.

[75] D. Lanman and D. Luebke. Near-eye light field displays. *ACM Transactions on Graphics*, 32(6), 2013.

[76] J. Lanman, E. Bizzi, and J. Allum. The coordination of eye and head movement during smooth pursuit. *Brain Research*, 153(1):39–53, 1978.

[77] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[78] S. M. LaValle. Help! My cockpit is drifting away. Oculus blog post. Retrieved from https://developer.oculus.com/blog/magnetometer/, December 2013. Last retrieved on Jan 10, 2016.

[79] S. M. LaValle. The latent power of prediction. Oculus blog post. Retrieved from https://developer.oculus.com/blog/the-latent-power-of-prediction/, July 2013. Last retrieved on Jan 10, 2016.

[80] S. M. LaValle. Sensor fusion: Keeping it simple. Oculus blog post. Retrieved from https://developer.oculus.com/blog/sensor-fusion-keeping-it-simple/, May 2013. Last retrieved on Jan 10, 2016.

[81] S. M. LaValle, A. Yershova, M. Katsev, and M. Antonov. Head tracking for the Oculus Rift. In *Proc. IEEE International Conference on Robotics and Automation*, pages 187–194, 2014.

[82] R. J. Leigh and D. S. Zee. *The Neurology of Eye Movements, 5th Ed.* Oxford University Press, 2015.

[83] M. C. Lin and J. F. Canny. Efficient algorithms for incremental distance computation. In *Proceedings IEEE International Conference on Robotics & Automation*, 1991.

[84] M. C. Lin and D. Manocha. Collision and proximity queries. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry, 2nd Ed.*, pages 787–807. Chapman and Hall/CRC Press, New York, 2004.

[85] S. Liversedge, I. Gilchrist, and S. Everling (eds). *Oxford Handbook of Eye Movements*. Oxford University Press, 2011.

[86] G. D. Love, D. M. Hoffman, P. J. H. Hands, J. Gao, A. K. Kirby, and M. S. Banks. High-speed switchable lens enables the development of a volumetric stereoscopic display. *Optics Express*, 17(18):15716–15725, 2009.

[87] R. G. Lyons. *Understanding Digital Signal Processing, 3rd Ed.* Prentice-Hall, Englewood Cliffs, NJ, 2010.

[88] R. Mahoney, T. Hamel, and J.-M. Pfimlin. Nonlinear complementary filters on the special orthogonal group. *IEEE Transactions on Automatic Control*, 53(5):1203–1218, 2008.

[89] A. Maimone, D. Lanman, K. Rathinavel, K. Keller, D. Luebke, and H. Fuchs. Pinlight displays: Wide field of view augmented-reality eyeglasses using defocused point light sources. *ACM Transactions on Graphics*, 33(4), 2014.

[90] W. R. Mark, L. McMillan, and G. Bishop. Post-rendering 3D warping. In *Proceedings of the Symposium on Interactive 3D Graphics*, pages 7–16, 1997.

[91] S. Marschner and P. Shirley. *Fundamentals of Computer Graphics, 4th Ed.* CRC Press, Boca Raton, FL, 2015.

[92] G. Mather. *Foundations of Sensation and Perception*. Psychology Press, Hove, UK, 2008.

[93] H. McGurk and J. MacDonald. Hearing lips and seeing voices. *Nature*, 264:746–748, 1976.

[94] R. Mehra, N. Raghuvanshi, L. Antani, A. Chandak, S. Curtis, and D. Manocha. Wave-based sound propagation in large open scenes using an equivalent source formulation. *ACM Transactions on Graphics*, 32(2), 2013.

[95] J. Merimaa and V. Pulkki. Spatial impulse response rendering i: Analysis and synthesis. *Journal of the Audio Engineering Society*, 53(12):1115–1127, 2005.

[96] A. Mikami, W. T. Newsome, and R. H. Wurtz. Motion selectivity in macaque visual cortex. II. Spatiotemporal range of directional interactions in MT and V1. *Journal of Neurophysiology*, 55:1328–1339, 1986.

[97] M. Mine and G. Bishop. Just-in-time pixels. Technical Report TR93-005, University of North Carolina, Chapel Hill, NC, 1993.

[98] B. Mirtich. V-Clip: Fast and robust polyhedral collision detection. Technical Report TR97-05, Mitsubishi Electronics Research Laboratory, 1997.

[99] B. Mirtich. Efficient algorithms for two-phase collision detection. In K. Gupta and A.P. del Pobil, editors, *Practical Motion Planning in Robotics: Current Approaches and Future Directions*, pages 203–223. Wiley, New York, 1998.

[100] T. Möller. A fast triangle-triangle intersection test. *Journal of Graphics Tools*, 2(2):25–30, 1997.

[101] T. Möller and N. Trumbore. Fast, minimum storage ray/triangle intersection. *Journal of Graphics Tools*, 2(1):21–28, 1997.

[102] B. Moore. *An Introduction to the Psychology of Hearing, 6th Ed.* Brill, Somerville, MA, 2012.

[103] H. S. Mortensen, B. Pakkenberg, M. Dam, R. Dietz, C. Sonne, B. Mikkelsen, and N. Eriksen. Quantitative relationships in delphinid neocortex. *Frontiers in Neuroanatomy*, 8, 2014.

[104] M. E. Mortenson. *Geometric Modeling, 2nd Ed.* Wiley, New York, 1997.

[105] E. I. Moser, E. Kropff, and M.-B. Moser. Place cells, grid cells, and the brain's spatial representation system. *Annual Reviews of Neuroscience*, 31:69–89, 2008.

[106] D. E. Muller and F. P. Preparata. Finding the intersection of two convex polyhedra. *Theoretical Computer Science*, 7:217–236, 1978.

[107] F. Nicodemus. Directional reflectance and emissivity of an opaque surface. *Applied Optics*, 4(7):767–775, 1965.

[108] J. Ninio. *The Science of Illusions*. Cornell University Press, Ithaca, NY, 2001.

[109] J. O'Keefe and J. Dosytovsky. The hippocampus as a spatial map. preliminary evidence from unit activity in the freely-moving rat. *Brain Research*, 34(1):171–175, 1971.

[110] G. Osterberg. Topography of the layer of rods and cones in the human retina. *Acta Ophthalmologica, Supplement*, 6:1–103, 1935.

[111] Sönke Pelzer, Lukas Aspöck, Dirk Schröder, and Michael Vorländer. Integrating real-time room acoustics simulation into a cad modeling software to enhance the architectural design process. *Buildings*, 2:1103–1138, 2014.

[112] I. Poupyrev, M. Billinghust, S. Weghorst, and T. Ichikawa. The go-go interaction technique: non-linear mapping for direct manipulation in VR. In *Proceedings ACM Symposium on User Interface Software and Technology*, pages 79–80, 1996.

[113] V. Pulkki. Virtual sound source positioning using vector base amplitude panning. *Journal of the Audio Engineering Society*, 45(6):456–466, 1997.

[114] V. Pulkki. Virtual sound source positioning using vector base amplitude panning. *Journal of the Audio Engineering Society*, 55(6):503–516, 2007.

[115] V. Pulkki and J. Merimaa. Spatial impulse response rendering ii: Reproduction of diffuse sound and listening tests. *Journal of the Audio Engineering Society*, 54(1/2):3–20, 2006.

[116] S. Razzaque, Z. Kohn, and M C. Whitton. Redirected walking. In *Proceedings of Eurographics*, pages 289–294, 2001.

[117] W. Reichardt. Autocorrelation, a principle for the evaluation of sensory information by the central nervous system. In W. Rosenblith, editor, *Sensory Communication*. MIT Press, Cambridge, MA, 1961.

[118] S. W. Rienstra and A. Hirschberg. *An Introduction to Acoustics*. Endhoven University of Technology, 2016. Available at http://www.win.tue.nl/∼sjoerdr/papers/boek.pdf.

[119] C. P. Robert. *The Bayesian Choice, 2nd. Ed.* Springer-Verlag, Berlin, 2001.

[120] P. Robinson, A. Walther, C. Faller, and J. Braasch. Echo thresholds for reflections from acoustically diffusive architectural surfaces. *Journal of the Acoustical Society of America*, 134(4):2755–2764, 2013.

[121] M. Rolfs. Microsaccades: Small steps on a long way. *Psychological Bulletin*, 49(20):2415–2441, 2009.

[122] D. Rosenbaum. *Human Motor Control, 2nd Ed.* Elsevier, Amsterdam, 2009.

[123] S. Ross. *A First Course in Probability, 9th Ed.* Pearson, New York, NY, 2012.

[124] G. Roth and U. Dicke. Evolution of the brain and intelligence. *Trends in Cognitive Sciences*, 9:250–257, 2005.

[125] W. Rushton. Effect of humming on vision. *Nature*, 216:1173–1175, 2009.

[126] M. B. Sachs and N. Y. S. Kiang. Two-tone inhibition in auditory nerve fibres. *Journal of the Acoustical Society of America*, 43:1120–1128, 1968.

[127] J. F. Schouten. Subjective stroboscopy and a model of visual movement detectors. In I. Wathen-Dunn, editor, *Models for the perception of speech and visual form*. MIT Press, Cambridge, MA, 1967.

[128] G. Schweighofer and A. Pinz. Robust pose estimation from a planar target. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2024–2030, 2006.

[129] M. Shelhamer, D. A. Robinson, and H. S. Tan. Context-specific adaptation of the gain of the vestibulo-ocular reflex in humans. *Journal of Vestibular Research: Equilibrium and Orientation*, 2(1):89–96, 1992.

[130] R. N. Shepard. Circularity in judgements of relative pitch. *Journal of the Acoustical Society of America*, 36(12):2346–2453, 1964.

[131] T. Shibata, J. Kim, D. M. Hoffman, and M. S. Banks. The zone of comfort: predicting visual discomfort with stereo displays. *Journal of Vision*, 11(8):1–29, 2011.

[132] B. G. Shinn-Cunningham, S. Santarelli, and N. Kopco. Tori of confusion: Binaural localization cues for sources within reach of a listener. *Journal of the Acoustical Society of America*, 107(3):1627–1636, 2002.

[133] P. Signell. Predicting and specifying the perceived colors of reflective objects. Technical Report MISN-0-270, Michigan State University, East Lansing, MI, 2000. Available at http://www.physnet.org/.

[134] S. Siltanen, T. Lokki, S. Kiminki, and L. Savioja. The room acoustic rendering equation. *Journal of the Acoustical Society of America*, 122(3):1624–1635, 2007.

[135] G. Smith and D. A. Atchison. *The Eye and Visual Optical Instruments*. Cambridge University Press, Cambridge, U.K., 1997.

[136] W. J. Smith. *Modern Optical Engineering, 4th Ed.* SPIE Press, 2008.

[137] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3D. *ACM Transactions on Graphics*, 25(3):835–846, 2006.

[138] B. R. Sorensen, M. Donath, G.-B. Yanf, and R. C. Starr. The minnesota scanner: A prototype sensor for three-dimensional tracking of moving body segments. *IEEE Transactions on Robotics*, 5(4):499–509, 1989.

[139] M. W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. Wiley, New York, 2005.

[140] R. M. Steinman, Z. Pizlo, and F. J. Pizlo. Phi is not beta, and why Wertheimer's discovery launched the Gestalt revolution. *Vision Research*, 40(17):2257–2264, 2000.

[141] N. Stephenson. *Snow Crash*. Bantam Books, 1996.

[142] S. S. Stevenson. On the psychophysical law. *Psychological Review*, 64(3):153–181, 1957.

[143] R. Stoakley, M. J. Conway, and R. Pausch. Virtual reality on a WIM: interative worlds in minature. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 265–272, 1995.

[144] R. Szeliski. Image alignment and stitching: A tutorial. Technical Report MSR-TR-2004-92, Microsoft Research, 2004. Available at http://research.microsoft.com/.

[145] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag, Berlin, 2010.

[146] L. L. Thompson and P. M. Pinsky. Acoustics. *Encyclopedia of Computational Mechanics*, 2(22), 2004.

[147] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.

[148] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzbiggon. Bundle adjustment - a modern synthesis. In *Proceedings IEEE International Workshop on Vision Algorithms*, pages 298–372, 1999.

[149] J. F. Vasconcelos, G. Elkaim, C. Silvestre, P. Oliveira, and B. Cardeira. Geometric approach to strapdown magnetometer calibration in sensor frame. *Transactions on Aerospace and Electronic Systems*, 47(2):1293–1306, 2011.

[150] M. Vorländer. *Auralization*. Springer-Verlag, Berlin, 2010.

[151] M. Vorländer and B. Shinn-Cunningham. Virtual auditory displays. In K. S. Hale and K. M. Stanney, editors, *Handbook of Virtual Environments, 2nd Edition*. CRC Press, Boca Raton, FL, 2015.

[152] B. A. Wandell. *Foundations of Vision*. Sinauer Associates, 1995. Available at https://foundationsofvision.stanford.edu/.

[153] X. Wang and B. Winslow. Eye tracking in virtual environments. In K. S. Hale and K. M. Stanney, editors, *Handbook of Virtual Environments, 2nd Edition*. CRC Press, Boca Raton, FL, 2015.

[154] R. M. Warren, J. M. Wrightson, and J. Puretz. Illusory continuity of tonal and infratonal periodic sounds. *Journal of the Acoustical Society of America*, 84(4):1338–1142, 1964.

[155] D. S. Watkins. *Fundamentals of Matrix Computations*. Wiley, New York, 2002.

[156] G. Welch and E. Foxlin. Motion tracking: no silver bullet, but a respectable arsenal. *IEEE Computer Graphics and Applications*, 22(6):24–28, 2002.

[157] R. B. Welch and B. J. Mohler. Adapting to virtual environments. In K. S. Hale and K. M. Stanney, editors, *Handbook of Virtual Environments, 2nd Edition*. CRC Press, Boca Raton, FL, 2015.

[158] M. Wertheimer. Experimentelle Studien über das Sehen von Bewegung (Experimental Studies on the Perception of Motion). *Zeitschrift für Psychologie*, 61:161–265, 1912.

[159] A. F. Wright, C. F. Chakarova, M. M. Abd El-Aziz, and S. S. Bhattacharya. Photoreceptor degeneration: genetic and mechanistic dissection of a complex trait. *Nature Reviews Genetics*, 11:273–284, 2010.

[160] F. E. Wright. *The Methods of Petrographic-microscopic Research*. Carnegie Institution of Washington, 1911.

[161] Y. Wu and Z. Hu. PnP problem revisited. *Journal of Mathematical Imaging and Vision*, 24(1):131–141, 2006.

[162] X. Yang and W. Grantham. Effects of center frequency and bandwidth on echo threshold and buildup of echo suppression. *Journal of the Acoustical Society of America*, 95(5):2917, 1994.

[163] N. Yee and J. Bailenson. The Proteus effect: The effect of transformed self-representation on behavior. *Human Communication Research*, 33:271–290, 2007.

[164] H. Yeh, R. Mehra, Z. Ren, L. Antani, M. C. Lin, and D. Manocha. Wave-ray coupling for interactive sound propagation in large complex scenes. *ACM Transactions on Graphics*, 32(6), 2013.

[165] W. A. Yost. *Fundamentals of Hearing: An Introduction, 5th Ed.* Emerald Group, Somerville, MA, 2006.

[166] P. Zahorik. Assessing auditory distance perception using virtual acoustics. *Journal of the Acoustical Society of America*, 111(4):1832–1846, 2002.

[167] V. M. Zatsiorsky. *Kinematics of Human Motion*. Human Kinetics, Champaign, IL, 1997.

[168] V. M. Zatsiorsky. *Kinetics of Human Motion*. Human Kinetics, Champaign, IL, 2002.

[169] V. M. Zatsiorsky and B. I. Prilutsky. *Biomechanics of Skeletal Muscles*. Human Kinetics, Champaign, IL, 2012.

[170] Y. Zheng, Y. Kuang, S. Sugimoto, and K. Aström. Revisiting the pnp problem: A fast, general and optimal solution. In *Proceedings IEEE International Conference on Computer Vision*, pages 2344–2351, 2013.

[171] H. Zhou and H. Hu. Human motion tracking for rehabilitation - A survey. *Biomedical Signal Processing and Control*, 3(1):1–18, 2007.

Index