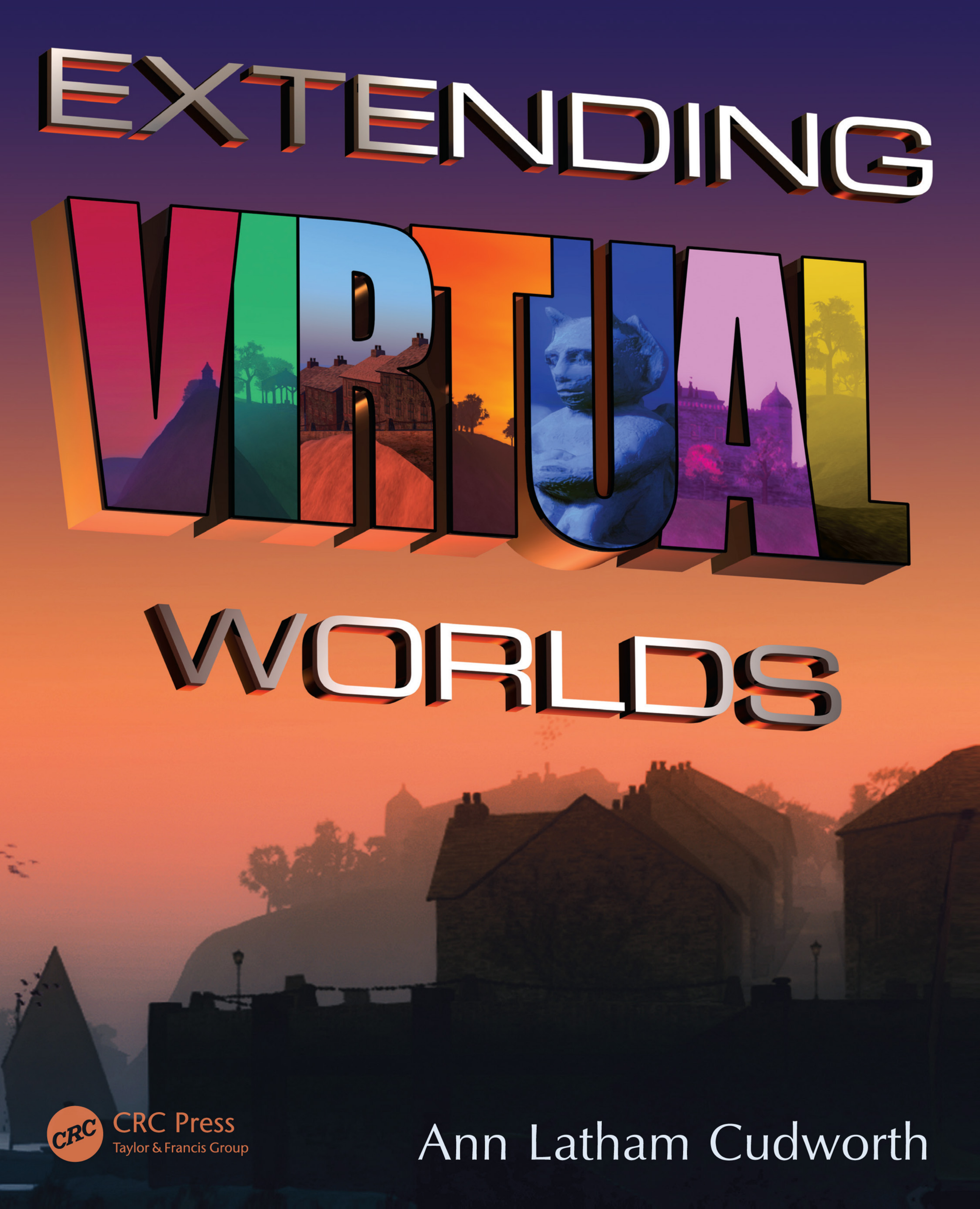
EXTENDING VIRTUAL WORLDS



CRC Press
Taylor & Francis Group

Ann Latham Cudworth

EXTENDING VIRTUAL WORLDS

This page intentionally left blank

EXTENDING VIRTUAL WORLDS

Advanced Design for Virtual Environments

Ann Latham Cudworth

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Version Date: 20151218

International Standard Book Number-13: 978-1-4822-6117-2 (eBook - PDF)

Contents

This page intentionally left blank

Preface

Almost 25 years ago, my first home computer arrived from Gateway 2000 in a black and white piebald box. I used it to run AutoCad12 and 3DStudio so I could make 3D models for my work as a set designer for film and television. Four years later, I came across an article in *Wired* magazine that changed the direction of my life and career. For the first time, I was exposed to the idea of creating television scenery as a 3D virtual environment. Emboldened by my intense curiosity and deep necessity to get some hands-on experience with this system, I contacted the virtual set company mentioned in the magazine and offered my services as a designer. They brought me out to California, trained me, and when I returned, I shared the information with my peers in New York. For the next 18 months or so, it was all the rage. I won an Emmy Award for a virtual set design in 1994. Eventually, interest in using these big, expensive computer systems for broadcast television waned. A few years later, real-time compositing systems entered the market, Internet bandwidth increased, and media websites got interested in producing videos with virtual backgrounds. We moved onto the creation of "2½D" virtual sets. These complex 3D environments with sophisticated lighting were rendered as flat images from three or four camera angles and composited as layers into the background frame in real-time while we recorded the talent and switched from camera to camera. I continued to design and build models for virtual scenery this way for the next 10 years. Then one day in 2008, when I was at home, sick with the flu, I stepped into Second Life for the first time. Suddenly, all my previous training came together in a venue that supported the scale I liked to work in. The experience of being in a socially connected collaborative environment changed my whole outlook on how designers could work and what they could utilize their talents for. With some friends, we formed a build group called Alchemy Sims and started to develop virtual environments, which became more and more like storytelling interactive games. Eventually, I decided to write about my experiences and the knowledge gained, and that is how this book and its predecessor *Virtual World Design* came into being.

THE PURPOSE OF THIS BOOK

I wrote this book for two reasons: to extend and enrich the understanding of design concepts for virtual environments that are introduced in my first book, and to provide content and methodologies for designing a game-based sim in a virtual world. This book will deepen your understanding of the process of game design and how that can be applied to creating a game-based virtual environment. Included with this book are the 3D models needed for building your own four-region game-based environment and projects that show you how to use it. I believe that we are all sharing in the birth of a new experiential storytelling genre. This will require that we create new structures and new frameworks that enable the visitor to understand our message. We are all in this together, the 3D content makers, the code writers, the storytellers, and the architects of virtual space. Each one has a significant contribution to make, and this book was written to support that effort.

INSIGHTS AND KNOWLEDGE GAINED WHILE WRITING THIS BOOK

I wrote this book because I wanted to learn more about building game-based environments and how they can be used in virtual worlds. Some of that process included learning about how the underpinnings of virtual worlds—the physics engine, the level of detail computations, and more complex scripts—enable us to create these kinds of environments. Processing all the references about these topics was like taking a drink from a fire hose at times, sometimes I got specific and useful design-based knowledge to pass on and sometimes there was just too much information to be included within these pages. I have tried to put forth the useful and memorable information a designer will need as they set out to build a game-based sim, as well as the sources I found most valuable.

The additional material necessary for the projects described in this book can be downloaded from the author's website: http://www.anncudworthprojects.com/.

Acknowledgments

My research and sources came from everywhere. The topic of virtual worlds is so new that a significant portion of the information still resides in people's memories and current experiences. Along with sources from technical papers, and reputable blogs, I also included "Spotlight" sections throughout the book that are firsthand accounts from experts in the field who are working to create new experiences in virtual worlds.

I extend special thanks to

Bruce Damer for his advice and comments on the history of virtual worlds in Chapter 5
David Luebke for his advice on Chapter 7
Deb Thomas for her spotlight contribution concerning games, training, and social media in Chapter 3
Fred Beckhusen for providing so much useful content and advice on the Outworldz website
Jacki Morie for her spotlight contribution concerning art, virtual reality, and education in Chapter 14
Jamie Thalman for his spotlight contribution regarding production design in virtual reality films in Chapter 13
Jopsy Pendragon (The Vehicle Lab) for his assistance with Chapter 8
Kiana Writer and MadPea Productions for their contributions to this book and game-based sims in Second Life
Scott Lawrence for his contribution to Chapter 7
Professor Vicki Robinson for her spotlight section in Chapter 8
Will Burns for his advice and comments overall

Special thanks to

Michael Fulwiler for his illustrative talents that bring clarity and interest to the figures in this book
Michael Thome for his work on the LSL scripts that are distributed with the projects in this book
Tim Widger for his excellent work on the 3D content for the projects included with this book

Also included in this list of acknowledgments is the support of my peers in the Metaverse: Arrehn Oberlander and the Atlas Chen, Azriel Xue, Ben Lindquist, Callum Diavolo, Chris Collins, Cynthia Calongne, Debs Regent, Eddie Mathieson, Grayson Stebbins, Henry Segerman, Ilan Tochner, Justin Clark-Casey, Maya Paris, MetaHarper Cabal, Morgan Phelan, Norton Burns, PatriciaAnne Daviau, Philip Rosedale, Radu Privantu, Ran Hinrichs, Reynaldo Vargas, Stone Librande, and Vicki Brandenburg to name but a few of the many, many people who have helped me along the way.

This page intentionally left blank

Introduction

Major concepts in this book include understanding and accessing creativity in design, methods for making game design documentation, working with advanced scripting and 3D models, utilizing advanced materials on these models and vehicles, and world building for game-based virtual environments. Following is a chapter-by-chapter synopsis.

CHAPTER 1: CONCEPTS IN ADVANCED DESIGN FOR GAME-BASED VIRTUAL ENVIRONMENTS

In this chapter you will learn about some big picture concepts that are intertwined with the process of designing game-based virtual worlds. Some of the topics discussed are transrealism design, designing for emergent game play, the reality–virtuality continuum, and understanding the vizome, as well as universal access and safety.

CHAPTER 2: ABOUT THE VIEWERS AND CONTENT CREATED FOR THIS BOOK

This chapter contains an overview of the "nuts and bolts" of virtual world viewers, and the necessary computer gear for accessing the virtual environments. It includes such topics as choosing a client viewer for 2D screen and head-mounted display (HMD) of your OpenSim environment, how to use the Upload menu, and where to get the content provided with this book.

CHAPTER 3: THE VIZOME, DEVELOPING CREATIVITY, AND VIRTUAL ENVIRONMENTAL DESIGN

As a designer, you will work with ideas big and small, and on many levels simultaneously, as you explore the underlying interconnections of a virtual environment, which we shall call the vizome. Key concepts discussed in this chapter are making the space for creativity in design, developing a creation system, and social interactivity in design.

CHAPTER 4: PLANNING AND PROTOTYPING THE DESIGN OF A GAME-BASED VIRTUAL ENVIRONMENT

Games in a virtual world have their own set of unique challenges. Knowing about game design, and how to plan for game-based sim requirements, is a good set of tools to have in your design repertoire. Key ideas involving designing for gameplay discussed in this chapter are about the structure of games, the kinds of players, and the process of game development. Included in this chapter is a project about creating four kinds of game development documents.

CHAPTER 5: DESIGNING FOR SOCIAL INTERACTION IN VIRTUAL SPACES

To be a virtual world designer requires an understanding of how social media works and how to build support for it into your 3D designs. Concepts we will examine in this chapter concern virtual social spaces, fostering communication, and the design of social spaces within a virtual world. Also included is a project about building the network of related content for social media.

CHAPTER 6: DEVELOPING THE DESIGNER–SCRIPTER COLLABORATION

Any game-based virtual environment will contain a number of scripts written to provide player interactivity, vehicular mobility, scenic changes, and other dynamic elements. The alliance between the designer and the scripter is explored in this chapter, with such topics as organizational structure of the team, understanding a common vocabulary, and working together effectively. At the end of this chapter is a collaborative game for the designer and scripter to play together, called Script Roulette.

CHAPTER 7: DESIGNING WITH LEVEL OF DETAIL (LOD)

Although we are several years away from totally realistic 3D worlds running in real-time on our displays, the capacity to achieve that goal increases with each new generation of graphics cards. In this chapter, we will explore such topics as level of detail (LOD) fundamentals for the designer, LOD optimization, and creative methodologies that leverage the use of LOD. At the end of this chapter is a project about creating content at various levels of detail by using photogrammetry.

CHAPTER 8: DESIGNING WITH VIRTUAL PHYSICS IN MIND

As a designer you should know how to work with the physics of a virtual world. The concepts discussed in this chapter are about physics engines and how they influence the design of objects, vehicles, and terrain. The project in this chapter concerns the scripting of a vehicle, a fairy boat, for the game-based sim.

CHAPTER 9: LANDSCAPE AND TERRAIN DESIGN IN VIRTUAL ENVIRONMENTS

To be a designer of virtual worlds and the landscapes they contain, you must understand the relationship between the terrain and your subjective "point of view." Various concepts included in this chapter are ways to approach terrain design, storytelling with terrain design, and methodologies for creating a terrain. A project about creating a four-region contiguous terrain is included in this chapter and is used for building the game-based environment included in the content of this book.

CHAPTER 10: DESIGN CONSIDERATIONS AND MESH USAGE IN VIRTUAL ENVIRONMENTS

This chapter includes various aspects about mesh usage in a virtual environment including choosing mesh-making software, what rigged meshes are, and how to obtain premade meshes. Included at the end of this chapter is a project about how to approach a large design challenge, how to utilize plans and diagrams for team coordination, and how to use the component mesh content provided with this book to build a game-based environment.

CHAPTER 11: DESIGNING WITH ADVANCED MATERIALS AND ANIMATED GRAPHIC/TEXTURES

The concepts included in this chapter relate to graphic/textures creation, and their application to inworld content in the Texture (diffuse), Bumpiness (normal), and Shininess (specular) channels contained in the Build/Edit/Textures menu. Also included is information about UV unwrapping/mapping and usage of animated graphics. In the project at the end of this chapter, you will work with a mesh fairy bridge, creating and applying advanced materials such as Bumpiness (normal), Shininess (specular), and animated graphic textures.

CHAPTER 12: WORLD BUILDING AND DESIGN WITH PROCEDURAL TECHNIQUES

Included in this chapter are key concepts about world building such as the use of a creation myth, and utilization of procedural techniques and scripts for world creation. The project included in this chapter involves the creation of a scripted procedural fairy tree.

CHAPTER 13: DESIGNING VIRTUAL ENVIRONMENTS FOR HEAD-MOUNTED DISPLAYS

In this chapter we shall consider various aspects of the stereoscopic virtual environment in a head-mounted display (HMD) and how these aspects influence the design choices that this completely immersive environment presents to us. Key concepts discussed in this chapter are the fundamental components of an HMD, the five dimensions of design for virtual environments, and the inherent design challenges in a virtual reality HMD space. The project included in this chapter is about developing and building an HMD/VR narrative environment for storytelling.

CHAPTER 14: THE MULTIPLE DIMENSIONS OF GAME-BASED VIRTUAL ENVIRONMENTS

In this chapter we will discuss some aspects of how game-based virtual environments might be developed for your teaching and training needs. Suggestions for developing games with game jams and competitions, and how virtual worlds can be used for prototyping are also included.

A NOTE ABOUT THE TERMINOLOGY USED IN THIS BOOK

I have made an effort to be consistent with my terminology, so that it coincides with the Client Viewer terms found in the on-screen menus, since that is the interface used by the most people. Please note: Sometimes the word *graphics* is used along with *texture* or as *graphic/texture*. I do this to indicate that once a graphic created in a 2D program has been uploaded inworld, it may be used in the Build/Edit/Texture menu in the Texture (diffuse), Bumpiness (normal), or Shininess (specular) layers or channels, and many people refer to graphics as textures at that point. Readers who are more experienced with 3D modeling programs may use the word *maps* to describe normal and specular textures, and that term is used in some of the text when referencing these outside programs and links. In general, I tried to use *graphics* as much as possible, since it is a universal term for computer-generated images, and *textures* when these images are in the virtual world applied to virtual objects.

NOTICE ABOUT THE CONTENT INCLUDED WITH THIS BOOK

Knowledge and best practice methodologies in this field are constantly changing. As new research and experiences broaden our understanding, changes in research methods or professional practices may become necessary. Practitioners and researchers must always rely on their own experience and knowledge in evaluating and using any information or methods described in this book. In using such information or methods they should be aware of their own safety and the safety of others, including parties for whom they have a professional responsibility. The publisher, author, contributors, or editors do not assume liability for injury and/or damage to persons or property from the use of any methods, content, projects, or ideas contained in this book. Neither the author nor the content creators for this book will accept responsibility for the contraction of any viruses, nor damage to a user's computer from any of the freeware or other software mentioned in this book. Although the software mentioned has been used safely and successfully by the author, it is to be used at the reader's own risk.

This page intentionally left blank

Author

Ann Latham Cudworth is a two-time Emmy Award winner who designs virtual and physical scenery for network television. Her goal is to inspire people to engage with the message embodied in her designs. A transplanted Bostonian thriving in New York City, she has shared her knowledge as a teacher of design and visualization at New York University, and at workshops and conferences worldwide for 16 years. Her weapons of choice are SketchUp, 3ds Max, and Photoshop. When not in a virtual world, she can be found listening to a Red Sox baseball game or painting watercolors. More information about her numerous projects is available on her LinkedIn page at https://www.linkedin.com/in/anncudworth.

This page intentionally left blank

Contributors

Michael Fulwiler (illustrator) is a native Californian. He transplanted to New York City to choreograph experimental dance theater. While exploring motion studies using computers, he also began to work as a freelance designer, which he continues to this day. His own work is currently focused on portraits and the figure, working conceptually using design programs, and then bringing that work to life using a variety of natural media.

"Stimulate the imagination with concept, form, and color" (http://www.mpfulwiler.com).

Michael Thome (Vex Streeter in the Metaverse) is a Boston-area native with an educational background in artificial intelligence. He has worked as a research computer scientist/software engineer for more than 25 years. When he discovered Second Life, he naturally gravitated to scripting as a focus of his explorations in virtual world content creation. This led to collaboration with several friends on *Scripting Your World: The Official Guide to Second Life Scripting* and moonlighting for the teaching staff of the University of Washington online program in virtual worlds. While his day job has only rarely been related to virtual worlds, he continues to be engaged in Second Life and Open Sim: using the process of coding to bring art to life, to enhance immersion into virtual spaces, and to add more fun and surprise to the otherwise ordinary. Thome contributed the scripts used in the projects for Chapters 8 and 12, as well as an enormous amount of advice to the author about scripting and the general workings of OpenSim.

Tim Widger (3D content creator), after a chance meeting in Second Life, has been collaborating with Ann Cudworth and building 3D models for the virtual environments they have created in Second Life and OpenSim (OpenSimulator) since 2008. He originates from Plymouth, England, and currently lives in Cornwall. He has a background in engineering and now works in social care. He is a keen guitarist, motorcyclist, and science fiction fan. He developed the 3D model content, based on concept images from Ann Cudworth, for the projects in Chapters 10, 11, and 13 using the 3D software Blender.

This page intentionally left blank

1 Concepts in Advanced Design for Game-Based Virtual Environments

Design is the method of putting form and content together. Design, just as art, has multiple definitions; there is no single definition. Design can be art. Design can be aesthetics. Design is so simple, that's why it is so complicated.

Paul Rand

1.1 OVERVIEW OF CONCEPTS IN ADVANCED DESIGN FOR GAME-BASED VIRTUAL ENVIRONMENTS

You pause for a minute, just before you unlock the final part of the riddle. It's been a long, frustrating, exhilarating search through town and country, human and fairy lands, to fit all the parts of this rebus together. When you finally get this last bit, the mystery of Castle Teague will be solved. But you don't want it to end; the hunt has been such fun …

Does that sound familiar? You may have had that experience in a virtual world. It is the feeling of involvement with the environment, immersion into the story of the place, and the excitement of discovering what the designer created for you. There is a big virtual world out there, and we want you to design for it. In this chapter you will learn about some big picture concepts that are intertwined with the process of designing game-based virtual worlds. We will examine some aspects of media-based and interactive components that you can apply to the design of your virtual game-based environment. Knowing about these will provide you with a general overview on the kinds of challenges and decisions that you, the virtual world designer, will deal with every day. These concept intros will also provide you with a general point of view on the specific topics we explore in greater detail later this book, so sit back and let them just wash over you; we will dig into details later. If you put these concepts into one-phrase sound bites, you would get a list like this:

- Design for virtual environments is transrealism design.
- Inherent in a virtual world is the potential to design for emergent game play.
- Design for virtual environments spans the reality–virtuality continuum.
- Design for virtual worlds is deeply interconnected with the vizome.
- Designing properly for virtual worlds includes universal access and safety.
- Code can be used as a design tool.
- The design process is a voyage of discovery.

1.1.1 Opening Your Understanding of Design

Your understanding of design, at its very foundation, is created by your brain from the input of your five senses. In a virtual environment, you may make 3D models "by the numbers," but creative design comes from what looks and feels best to you. Perception is what counts. You may have a good design by the numbers, but be unable to sell it to clients, because of their preconceived notions, perceptions, or inability to get a feel for it. Try this exercise the next time you are in an interesting space. Put the smartphone in your pocket, turn off the music, and tune into the space around you. Look and think about the structures, masses, and surface textures you see. Listen and think about how the space sounds and how the surfaces in the space affect that sound. Feel the ambient temperature of the space, the sensations of touch and pressure that the space creates on your skin and eardrums. Smell the air and think about how the materials and activities in the space add to the ambiance. Taste with your eyes and your tongue, looking for colors or shapes in the space that look good enough to eat and make your mouth water. A good virtual environment design brings all this information across to the visitor. For instance, the creepy underwater environment of the *BioShock* game (developed by Irrational Games, published by 2K Games, August 2007) hits you on all those sensory levels. Paul Hellquist (designer) and Scott Sinclair (artist) paint the walls with the color of dread and decorate the spaces with the patterns of fear. In the less slimy category, the production design of *Tron: Legacy* (Walt Disney Productions, 2010) by Darren Gilford, creates an environment that is part virtual, part real. Gilford's designs create a vast plane of virtual existence that suppresses our humanity with overwhelming scale and unearthly light, the place feels as chilly as the core of a data center. *BioShock*'s underwater tunnels and *Tron: Legacy*'s digital platforms are both good examples of environment design based on sensory observations. This design is instinctual; you know it when you see, hear, feel, smell, and taste it.

1.1.2 Letting Go of the 2D Screen

The 2D screen is a crutch. While having a clear orthographic view of an object or scene can help you compare relative sizes and make the manipulation of 2D menu items more accessible, ultimately, you are working with a distortion of real 3D space. Imagine if you had to walk around all day with a pane of glass in front of you, clicking on icons to eat your breakfast or take the elevator. Now imagine that you can break that glass and step into the world of your 3D digital environment as easily as you walk out of your front door. As we near the first quarter of the 21st century, the artifice of the 2D interface is fading away. Head-mounted display (HMD) devices like Oculus Rift (Oculus VR), Project Morpheus (Sony), and Samsung VR, all under rapid development, will allow you to enter and interact with the virtual world in a 3D space. Apple has patented a 3D gesture-hover-based control for its tablet, eventually allowing you to "pull" a 3D form off the 2D screen. This is one of the many devices that will promote the use of augmented reality (AR), and mobile devices that will break you away from the desktop screen and allow you access to the virtual world from anywhere in the real one [1]. The time has come to think of how your designing process can be enhanced by the display device, and how the real and virtual worlds can coexist in virtual 3D space.

1.1.3 Qualities of a Good Designer

There are five qualities that all good designers share. They all (1) try and fail often in order to improve their designs, (2) listen in order to broaden their understanding of human life, (3) constantly read to feed their hunger for knowledge, (4) uphold standards of quality for design and professional behavior, and (5) can lead a team while maintaining their own personal thinking space. There are many creative design paths that intertwine in virtual worlds and the platforms that support them. The world of virtual environment design is like a great undiscovered continent; there are many entry points and several ways to get lost. Developing and practicing these five qualities will give you the tools to find your way, and forge a path for you and your design group.

1.2 TRANSREALISTIC DESIGN, CONSTRUCTIVE IMAGINATION, AND VIRTUAL ENVIRONMENTS

Like most designers, you probably want to create deeper levels of meaning in your virtual environment designs. To accomplish this, you must understand that designing with layers will create a deeper impact in the visitor's comprehension of your design and allow them to discover underlying messages in your virtual environments. Your designs should convey to the visitor that your virtual environment should not be taken at face value; that it has many faces, many depths, and many stories waiting for them to explore and contribute to. In literature, there is a style known as transrealism. Transrealism is a mash-up of the pure escapism of science fiction ("In a galaxy far, far away") and natural observations from the real world ("At midnight, three of us stood in the middle of Times Square watching *Star Wars* on the giant screen") [2]. An example of transrealistic writing from these scenarios could be

> At midnight, three of us stood in the middle of Times Square watching *Star Wars* on the giant screen. Perhaps it was the graphic overload of the surrounding signage, perhaps it was the extra tequila shots we did, but something had changed. The storefronts, windows, and even the street surface, slick with a coating of light rain, were all reflecting images from other places, alien places, back to our eyes. Beside us, on the windows of McDonald's we could see the green sands of N'inara, and across the street reflected on the shiny side of a black limo, we saw glimpses of the lake filled hills of Garth.

Transrealistic designing embodies that format in the design of the landscapes, architecture, and graphics of the virtual environment; it moves the visitor to consider the possibility of other planes of reality coexisting within virtual spaces.

From a design standpoint, constructive imagination is the process of using your imagination, your memories, or the content of your mind's eye to construct or produce new worlds. This mental process should be initiated in the minds of your visitors as they experience your virtual environments. Exposure to what you have designed should stimulate their imaginations and influence their responses and actions. Your virtual environment should be the sandbox of imagination; not necessarily a blank canvas, but more like a series of recombinant options or an "imagination system." Give your visitors opportunities to construct their own story and play their own games in your environments. This is beyond role-playing games (RPGs), because it allows for the visitor to combine realities within your system instead of just playing a part.

1.2.1 Designing for Transrealism; Some Initial Considerations

Every day in our modern lives, we get closer and closer to living full time in the state of transrealism. Perhaps you have experienced a lucid dream, entering directly into a dream while remaining aware that you are dreaming. Maybe you have experienced the sensation of viewing yourself from up above while you walked down the street, just as your camera would view your avatar character in a virtual world. These altered points of view, intersecting with your mind's eye and imagination, are the harbingers of transrealism. As we bring the realms of virtual reality closer to our bodies with devices that are wearable and mobile, as we develop them for motion tracking and global positioning, we intensify the state of transrealism. Augmented reality (AR) brings virtual objects and information into our real world experience, whereas augmented virtuality (AV) brings our real selves into the virtual world. We experience transrealism through a continuum of the reality of real life blending into virtual life.

In 1946, Jorge Luis Borges (1899–1986) wrote *On Exactitude in Science* with Adolfo Bioy Cesares (1914–1999), under the joint pseudonym of B. Lynch Davis. This one-paragraph story was written as a quote from a fictitious book.

On Exactitude in Science

In that Empire, the Art of Cartography attained such Perfection that the map of a single Province occupied the entirety of a City, and the map of the Empire, the entirety of a Province. In time, those Unconscionable Maps no longer satisfied, and the Cartographers Guilds struck a Map of the Empire whose size was that of the Empire, and which coincided point for point with it. The following Generations, who were not so fond of the Study of Cartography as their Forebears had been, saw that that vast Map was Useless, and not without some Pitilessness was it, that they delivered it up to the Inclemencies of Sun and Winters. In the Deserts of the West, still today, there are Tattered Ruins of that Map, inhabited by Animals and Beggars; in all the Land there is no other Relic of the Disciplines of Geography.

Suarez Miranda
Viajes de varones prudentes, Libro IV, Cap. XLV, Lerida, 1658 [3]

This arresting image of a full-scale map lying in tatters over a grim reality inspired Jean Baudrillard to say "we live in a desert of the real" [4]. Baudrillard believed that we live in a hyper real world, where the images we see, the sounds we hear, the emotions we feel for our celebrities, television characters, and news stories are more real to us than actual reality. The clamor of various realities, each seeking to gain the sole attention of your audience, creates a desperate need for design mediation. As a designer of virtual environments you must ask the following questions on each project:

1. Where are the real and virtual boundaries in my designs?
2. How and why should I show these boundaries?
3. What will the visitor gain from seeing or sensing these boundaries?

1.2.2 Constructive Imagination and Your Visitors

As the designer of a virtual world, you can play God, creating everything you can imagine. We have all seen beautiful examples of this throughout the Metaverse. In many of these perfectly created, highly detailed spaces, it often seems that your goal is to go through the environment as fast as possible and "level up" to the next experience. Perhaps this is a good idea when these environments are as static and airless as the period rooms in a museum. They leave no possibility of fingerprints or any mark that a player has been through. Any sandbox region in Second Life or OpenSim has a more interesting and dynamic environment. For instance, consider the Linden Endowment for the Arts (LEA) sandbox in Second Life. All week long, artists come by to rezz large builds, try out new ideas, and create an ever-changing landscape of their developing projects. By Friday, the fields and work platforms are filled with content and must be cleared. Imagine how interesting and dynamic a virtual gaming or massively multiplayer online role-playing game (MMORPG) environment would be if it constantly grew and changed through visitor interaction. Perhaps some form of creative "weather" that interacted with objects to alter them over time could be included in the environment. Both *Minecraft* (created by Mojang, 2009–2014) and *EverQuest Landmark* (created by Sony Online Entertainment, 2013–2014; now owned by Daybreak Games, 2015) have been developed from the concept of player interaction and creation of content with the environment. One key question designers should ask themselves: How is your visitor's constructive imagination stimulated within your virtual environment?

1.2.3 Fitting Transrealism and Constructive Imagination Together by Design

How do you, the designer, fit all of this together? This process parallels the behavior of creation through narrative imagination. Just as a child will make up narrative stories based on characters they have seen on television or action figures they have in their toy chest, the designer looks for the narrative stories that arise from and can be connected to their environments. For instance, suppose that you were given a promotional umbrella from an event that you attended, and then absentmindedly left it on the subway train. The umbrella has embarked on a journey of its own across the city. Ask yourself the following questions:

1. What kind of story can be told about that umbrella?
2. As various scenarios are invented, what level of reality can be utilized to make the story have more meaning?
3. How will other people be involved with this story, or interact with the umbrella?

Essentially, the elements for a good story come first, then the plot, and then the world. The designer weaves his tapestry of designs on that loom. It is entirely possible that the designer could write the story and the plot as well. As you can see in Figure 1.1 the concepts of transrealism and constructive imagination are intrinsic and interconnected parts of a virtual world environment.

Now, the next question is, When does it become a game?

1.3 DESIGNING FOR GAMES IN YOUR VIRTUAL ENVIRONMENT

Think about this question for minute: What is a game? If you look around, you will discover many definitions of what a game is. You may think of games as structured activities requiring graphically decorated boards or handheld consoles. You may define virtual worlds or life itself as a game. Games are ubiquitous; they can be fit in anywhere. In any language or scale, they all contain style, structure, and the capacity to encourage emergent play. They are, in a word, “constructivist” by nature, because they help us build the frameworks of experience that let us learn about the worlds around us, and the behavior of people in them.

1.3.1 Major Design Aspects for a Game in a Virtual Environment

Games can include many things. Jesse Schell gives this broad definition in his book *The Art of Game Design*: “A game is a problem-solving activity, approached with a playful attitude” [5].

Designing a game is also a problem-solving activity, and it can be approached with a playful attitude as well. There are two major aspects of design in gaming: (1) game style or the visual/aural identity of the game, and (2) game structure or the guided flow of the game play.

1.3.1.1 Designing the Game Style

This is all about knowing your personal relationship to history and culture. Everyone is surrounded by art, art history, and the culture that contains it. The more you observe these things and study their interconnections, the more you will begin to see the panorama of style across our shared cultural experience. Style is a distillation of culture, filtered through your personal taste and need for self-expression. Everything you design has a bit of your personal style, because it has your design “DNA” in it, and will reflect your conscious and subconscious relationship to the worlds around you. Observing your own style, adding to it, refining it, and making it serve the needs of the project at hand is what a designer does. Style is a powerful tool for storytelling; it reaches our senses on many levels, it can find a common ground with a person’s own sense of identity, or profoundly alter their perception of what style could be.

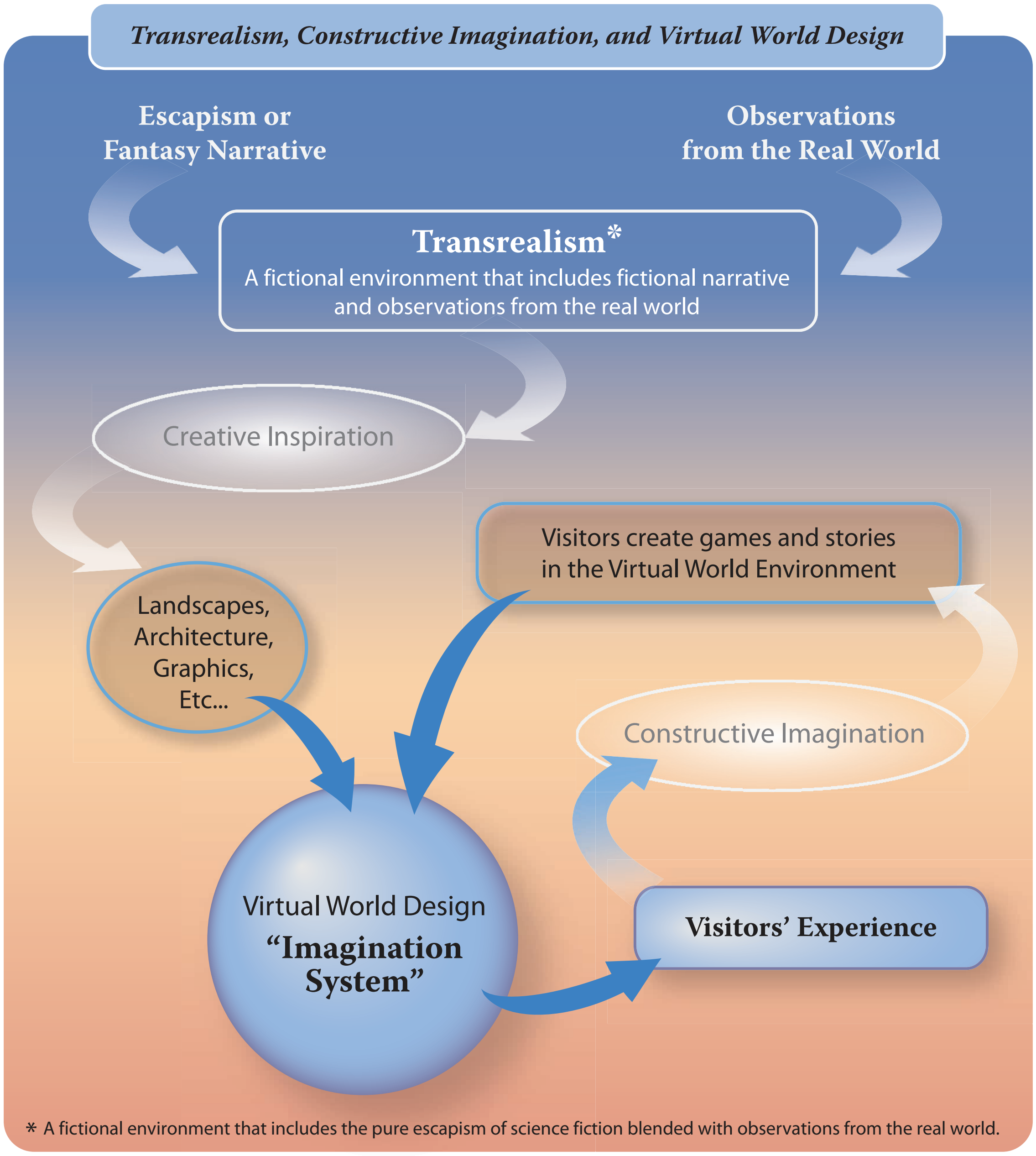

Transrealism, Constructive Imagination, and Virtual World Design
Escapism or Fantasy Narrative
Observations from the Real World
Transrealism*
A fictional environment that includes fictional narrative and observations from the real world
Creative Inspiration
Visitors create games and stories in the Virtual World Environment
Landscapes, Architecture, Graphics, Etc...
Constructive Imagination
Virtual World Design "Imagination System"
Visitors' Experience
* A fictional environment that includes the pure escapism of science fiction blended with observations from the real world.

FIGURE 1.1 The interconnections of transrealism, constructive imagination, and design inherent in virtual environments.

A game exhibits style on many levels. Usually the first impact on a player is made through the visual design choices, displayed on the opening splash screens. For example, the mythic background of the *EverQuest Next* game (created in 2013 by Sony Online Entertainment) and the bleak landscape of *Dear Esther* (commercial release in 2012 by The Chinese Room) both establish the visual style of the game environs and what they contain. Usually this is supported by a musical score or sound loop that hooks the player into the emotional tone of the game, pounding drums for a military campaign, lilting woodwinds for a fantasy game. And so it goes, every element in the game shares some portion of the style DNA, creating a cohesive experience.

1.3.1.2 Designing the Game Structure

Designing for the game structure is all about influencing the behavior and motivation of your player. There are many kinds of players in a game, all playing for various reasons. A good game design has elements and challenges that interest all players; a great design unites them in a common goal. *World of Warcraft* (created in 2004 by Blizzard Entertainment) contains one of the great game structure designs. *WoW*, as it is called by the gaming community, contains a complex set of quest challenges and player levels, providing constant motivation for the player. Over the years, players have joined together to form guilds that quest together for the benefit of all. Your game does not have to be as complex as *WoW* to have great game structure or flow. It just has to have a clear consistent set of rules and clearly defined goals that engage the player's interests, support communal cooperation, and encourage collaborative play.

1.3.2 Designing for Emergent Play

One of the most interesting and exciting things a designer can experience is the phenomenon of emergent play within their game. Emergent play is complex play behavior that develops in a game through the use of game elements by the players in unexpected or unplanned ways. For example, in 2008, the Alchemy Sims build group made a season-changing sim in Second Life called Sundial. Built to resemble the midlands of Scotland, all the trees, terrain, and water features on the sim would change their seasonal textures every 15 minutes as commanded by the central timer script, located in an equatorial sundial at the entrance. The environment could also be changed by verbal command from any visitor. If someone shouted "season winter" the trees within a 96 meter sphere would drop their leaves, and snow would start to fall. The water in the loch would also freeze over and become hard enough to support the weight of a visitor's avatar walking on it. One night, the builders of Sundial discovered a group of visitors playing a game with the seasonal commands. They would command the water to freeze on the loch, run out on it, and then shout "season summer" to unfreeze the water. They competed to see who could get off the ice before it thawed and dropped them into the water. The players in Second Life had found a way to make a game out of a simple interactive script that was not initially intended to be part of the game play. Designing for emergent game play is all about understanding how players in a game will respond to objects and interactive programming that contribute to openness and collaboration in a game-based environment. This understanding comes from observation of human behavior and preferences in video gaming. The need for achievement, the need for social connection, and the need for immersion are three of the most common motivators in these environments [6]. In Figure 1.2, there is a diagram of how these factors, these needs, interact with the elements of a game-based virtual environment.

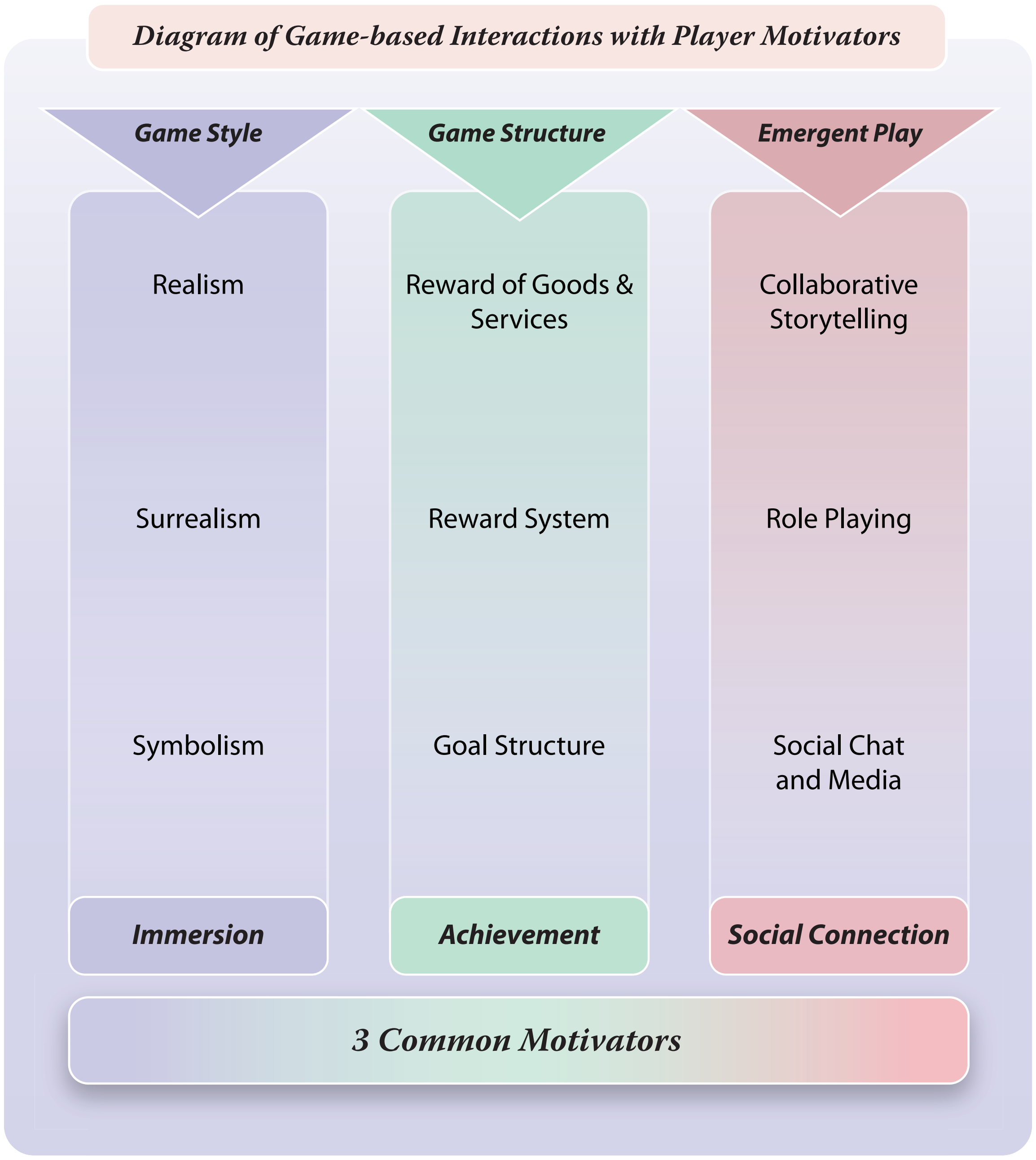


FIGURE 1.2 The interactions of game style, game structure, and emergent play with the three common motivators for a game player.

1.4 DESIGNING FOR THE REALITY–VIRTUALITY CONTINUUM

Plato described levels of reality in his Allegory of the Cave. His pupil, Aristotle, sought to refine those theoretical observations in his works, as he worked on creating the foundation of scientific thinking [7]. For centuries, philosophers, scientists, writers, filmmakers, and others have worked to understand what reality is and express their vision of how we perceive it. Timothy Leary (1920–1996) coined the term "reality tunnel" and Robert Anton Wilson (1932–2007) expanded on that idea in his book *Prometheus Rising* (published 1983, New Falcon Publications) summing up with "Truth is in the eye of the beholder." Perhaps Anais Nin said it the best: "We don't see things as they are, we see them as we are" [8].

Each of us spends some time every day modifying reality to our personal taste. Setting up subscriptions on Reddit so you can read news about your favorite topics or grooming your Facebook page are examples of this. All of us are the designers of our own personal media-based realities. As we develop technologies that allow us to connect more and more with synthetic realities like virtual worlds, we gain greater access to the destinations along the reality–virtuality continuum (R-VC). Currently there are four major landmarks on that continuum: (1) reality or the real world, (2) augmented reality, (3) augmented virtuality, and (4) virtuality. Each of these places has a great need for the services of a designer who is a creator, curator, editor, and mediator. These are designers who specialize in the creation of culturally related artifacts by curating a collection of images and objects from a wide spectrum of available icons and memes, who use those choices to edit the focus of the message, and mediate the flow and exposure of the virtual environment for the visitor. Let us look at how a designer gets involved with the four major locations along the R-VC. In Figure 1.3 is a diagram showing a symbolic explanation of this concept and possible ways that a designer can interact with the reality–virtuality continuum.

1.4.1 Designing for Reality

Reality is the doorway. Each design in this part of the R-VC concerns itself with the qualities of 3D space as perceived through our immediate senses. The dimensions, weight, feel, smell, and taste of an object are all impacted by the design process. For many designers, this is the place where the templates for virtual environments are made. For example, a designer may create a clay sculpture of a character avatar for digitizing into a virtual format, or they may create a kiosk for use with associated virtual websites and online virtual worlds. As you see in Figure 1.3, at this point, there is a photographic image of an object designed for the real world that serves as the seed for the other designs on the continuum.

1.4.2 Designing for Augmented Reality (Mixed Reality)

Augmented reality is the place where virtual objects and the information they contain are merged into the real world. Access to this part of the R-VC requires a device worn or held by the visitor. A designer in this realm should be concerned not only with the configuration and look of the real space, but also with the visual compatibility of the virtual elements to that real space. Some examples of this include the displays of weather information behind and around your local news weatherperson, or the first down line overlaid on the football fields during television sports coverage, or perhaps you have heard about light-field images that can create the same kind of 3D structures in our visual field as the real world does when we observe it. When these light-field images are displayed to our eyes, they allow us to see true depth in a virtual environment rather than a stereoscopic version of a flat display. These kinds of images form the basis of the Magic Leap system, under development for use with AR since 2014 [9]. In Figure 1.3, you can see how the real-world object has been made into a virtual element, and then incorporated back into the real world to augment the reality there.

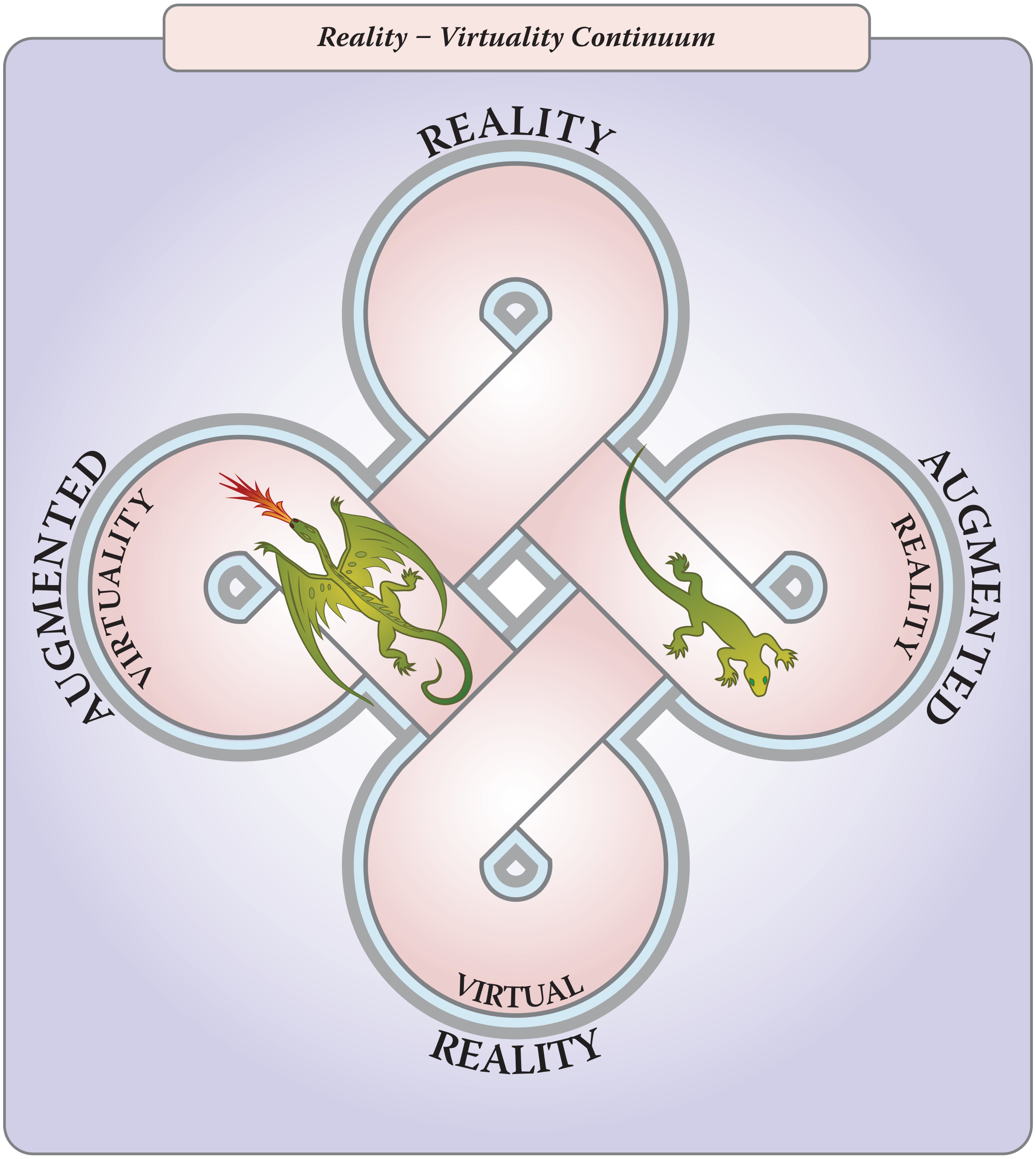


FIGURE 1.3 A symbolic representation of the reality–virtuality continuum.

1.4.3 Designing for Augmented Virtuality (Mixed Reality)

Augmented virtuality is where reality-based images and information are brought into a virtual environment, such as a live video feed being streamed into a virtual world or a performer's image being added into a virtual stage setting. In this kind of space, a designer should be concerned with relative scale and visual consistency of the combined sources of visual information. Examples of this are prevalent throughout the Metaverse, and are found in the many virtual environments used for group meetings and performances. The virtual space will contain screens that carry a media stream, and the area may be set for audio streaming as well.

1.4.4 Designing for Virtual Reality

In the widely available public virtual environments, such as OpenSim, a designer has to work within the limitations of two senses, sight and sound. All of the qualities of real-world items must be implied through the visual appearance and sound of objects. Although this may change in the near future, with today's limited haptic technology, only the crudest sense of touch can be transmitted. In a virtual world, physical behavior of objects and avatars is determined by the code of the physics engine available and not always realistic. This can be exploited by the designer in many ways, for example, through the use of teleportation, flying, super scale, and nonphysical or phantom objects. Examples of this are found in video games and virtual worlds like Second Life.

1.5 DESIGNING FOR THE INTERCONNECTED VIRTUAL ENVIRONMENT; FROM A RHIZOME TO THE VIZOME

About 1 mile southwest of Fish Lake, Utah, in the United States, lives Pando, who is also known as "The Trembling Giant." Pando is the largest single living organism on our earth; he is a clonal colony of male quaking aspen trees (*Populus tremuloides*). Pando is estimated to be 80,000 years old; he weighs about 13,000,000 pounds (5900 tonnes) and covers 106 acres (43 hectares). Pando is connected; every tree in the grove shares a networked root system, or rhizome [10]. The World Wide Web, and the webpages it connects, supersedes the size of Pando's network by many orders of magnitude [11] but does the same thing: it shares essential elements across the system. For Pando, these essential elements are vital nutrients and minerals; for the World Wide Web, they are packets of vital information and search results. We all live and work in this vast system, this interconnected "rhizome" of shared information. You may even think of the people in wired populations of the world as those trees, linked into the shared system, and still physically responsive to the real world. When you are designing for virtual worlds, you are designing for that interconnected system.

1.5.1 Introducing the Concept of a Vizome

From the study of nature-based rhizomes (like Pando), a new kind of philosophical concept was developed by Gilles Deleuze and Félix Guattari. In the "Capitalism and Schizophrenia" project (1972–1980), they sought to create a model based in the rhizomatic biological structures that would describe a new way of looking at informational exchange and the concepts it supports. In the subsequent book about the project, *A Thousand Plateaus: Capitalism and Schizophrenia*, they define the behavior of a rhizome in language and culture in this quote: "A rhizome ceaselessly establishes connections between semiotic chains, organizations of power, and circumstances relative to the arts, sciences and social struggles" [12]. This quote defines the crux of it and provides us with a conceptualization of what is going on under the surface of a virtual world. We could

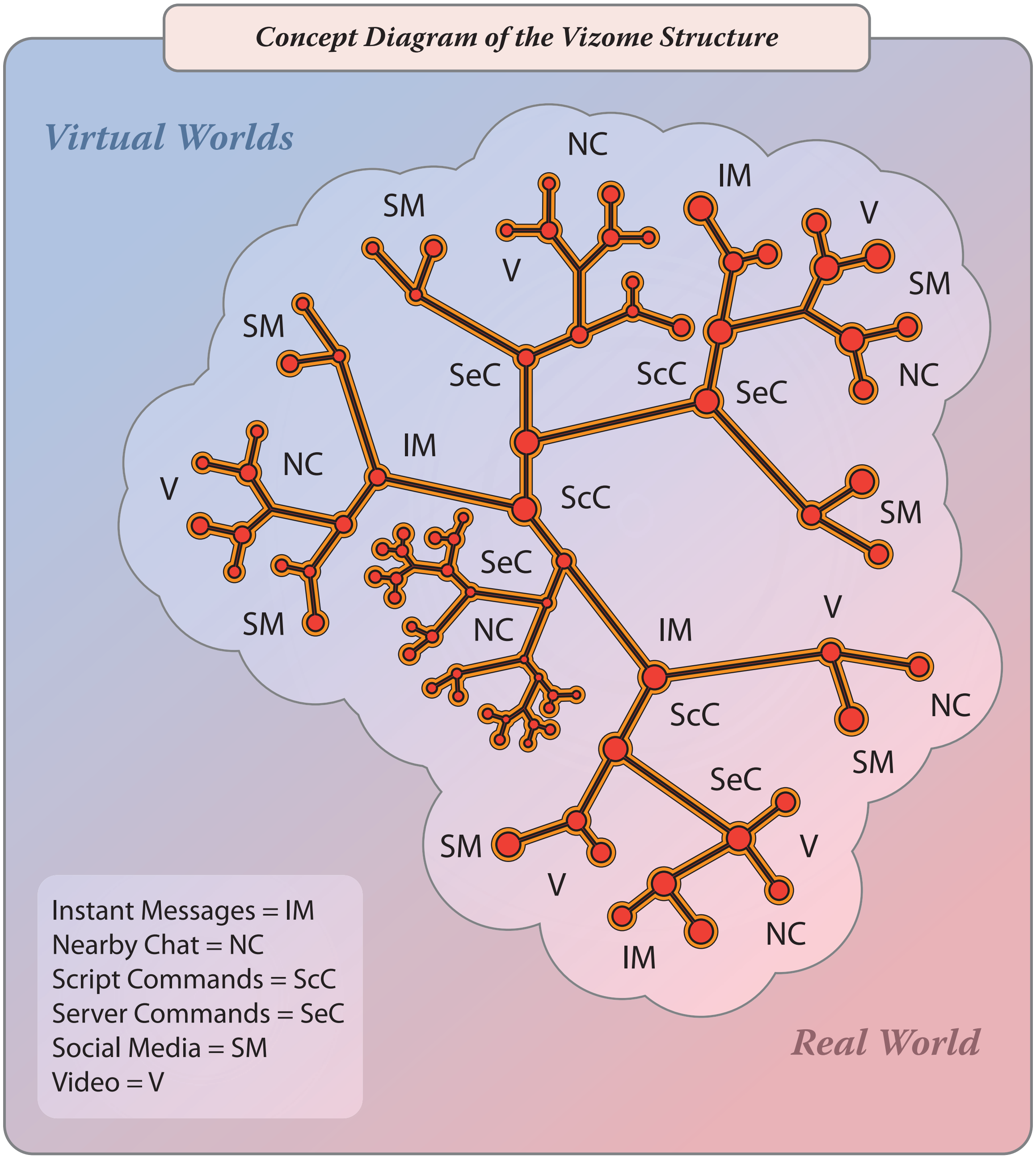


FIGURE 1.4 The vizome interconnecting virtual worlds of a constantly expanding Metaverse.

call this a vizome, if we make a portmanteau from the words, virtual and rhizome. In every virtual world, a portion of the vizome is there, ceaselessly making connections in instant messaging (IM) and nearby chat between avatars, in the channel-based communications between objects containing scripting and the simulation, between the simulation and outside servers, between simulations that contain regions and grids, and so on. The levels go on and on. This is what you design for a virtual world, the connective structure of communication, and the more you understand it, the greater reach and power your designs will have. In Figure 1.4 you will see a concept diagram of the vizome structure that interconnects the real and virtual worlds within our constantly expanding Metaverse.

1.6 UNIVERSAL DESIGN AND SAFETY IN VIRTUAL ENVIRONMENTS

Designing for the virtual world involves an understanding of designing for the real world. It also means that you should design for universal access, or what is called *universal design*. Behind every avatar is a human being that sees, hears, and touches the world in different ways. Throughout this book there will be notes about universal design as it applies to various aspects of designing and creating for virtual environments. Also in this book will be information about how safety can be added to the virtual environment by design.

1.6.1 The Common Principles in Universal Design in Virtual Environments

This is not an exhaustive list but rather some common principles to keep in mind at all times when you are creating a design that can be accessed by all.

1.6.1.1 Visual Design

All visual design should be checked for the following characteristics: (1) that it has the appropriate level of detail (LOD) at most common draw distances (90–256 meters), (2) it is visible to the color blind through the use of color blind testing plugins for your 2D graphics programs, (3) text on signage is clear and large enough for people with low vision, (4) pathways and directional signage is clearly laid out and easy to understand, and (5) the WindLight or environmental light settings on your region have been set to enhance the clarity of your visual presentation.

1.6.1.2 Audible Design

Audible design should include the following qualities: (1) the recording or sound clip is of the highest quality and clarity possible, (2) the soundscape should enhance and clarify the meaning of the landscape, (3) when possible the visible signage should be backed up with an audio clip that reads what the sign says, (4) sound levels should be set to focus on the area where they support the meaning of an object or environment, and (5) the sound emitter should be near the default camera position, so the majority of visitors will hear it without having to change their basic settings.

1.6.1.3 Tactile Design

Tactile design is still a neonate in virtual world design. Until there is wide access to haptic accessories that can bring the full spectrum of touch sensation to the visitor, the abstract tactile qualities you can focus on are things like "visual texture" and "sound texture." Here are some of the qualities that you can strive for: (1) keep the visual texture at the level where it supports the overall message of the environment or object, that is, a brick wall should use Shininess (specular) and Bumpiness (normal) graphics that enhance the apparent grittiness and bumpiness of the brick surface; and (2) use the sound effects and musical clips to create an overall mood and symphonic texture for the visitor, that is, a visit to your haunted house should provide an end-to-end flow of sound effects and background music.

1.6.2 Designing Safety into a Virtual Environment

In 1998, referring to the real world, James Howard Kunstler said, “The best way to bring security to streets is to make them delightful places that honorable and decent citizens want to walk in” [13]. C. Ray Jeffery, a proponent of crime prevention through environmental design (CPTED), and Oscar Newman’s defensible space design theory have contributed to the developing practice of designing private and public spaces with security and safety in mind all over the real world [14]. It is reasonable to assume that as our virtual environments become more sophisticated and realistic, design for security and safety will become as important in the virtual space as it is in the real world. There is a set of observations in CPTED called the *3Ds*, and they are adapted here to enable you to evaluate any space from a virtual user’s perspective [15].

The 3Ds for virtual spaces are

1. Designation of purpose is shown within the space. Can the user tell what it should be used for by its layout, signage, dress code, and décor?
2. Definition of what behavior is encouraged by the designated purpose. Is the user given a clear idea of how to behave in the space by the management and the other users?
3. Design to inspire the appropriate behavior in the space. Is the user inspired to exhibit appropriate behavior in the space by the dress code, layout, signage, and décor?

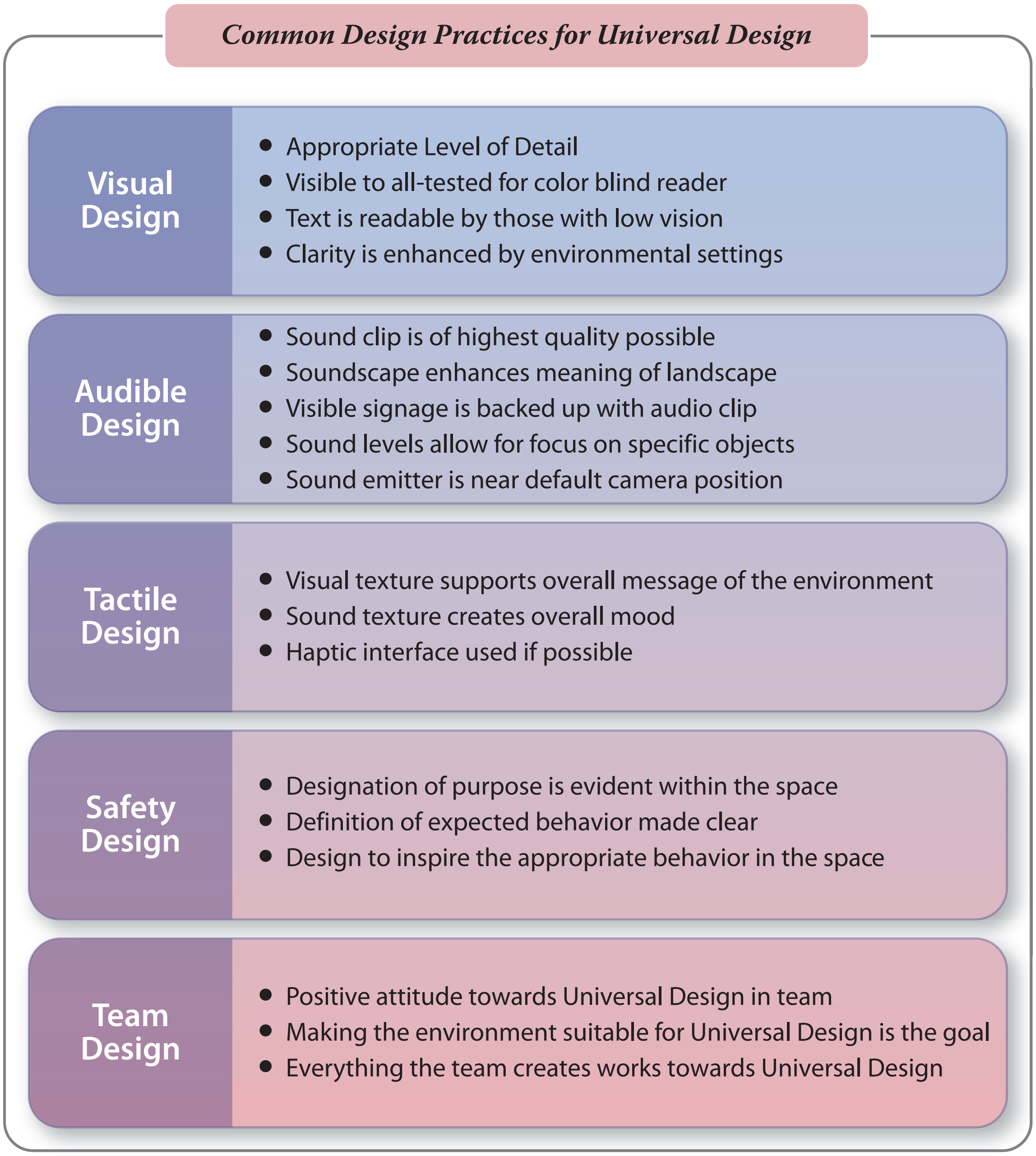
Think about your real-world environment for a minute and ask these questions.

1. What are the areas that people consider safe and why do they feel that way within them?
2. Where do criminals like to hang out and misbehave, and why do they feel emboldened to behave that way in those spaces?

Typically the qualities of a safe environment include good visibility or surveillance of the surrounding area at all times of day or night. There are people around, social activities are going on, and children are supervised. The safe areas are inviting and comfortable, and their boundaries are clearly marked, so people know when they are going into private or “off limits” areas. On the other hand, high-crime areas are often abandoned, dark, unsupervised, and unmarked. Now ask yourself, What are the virtual world equivalents to these two kinds of environments? Where and when do you feel safest in a virtual world? In your home? At the local sandbox? In a virtual world, have you ever been threatened by someone with a weapon on your land or in a public space? What could have been done to prevent that? Throughout this book, observations about safety and notes about how to add it into your virtual environment design will be included. A safe online virtual environment is a busy and successful space within a thriving community.

1.6.3 Working toward a Team Approach in Universal Design

Design coordination across your team not only includes the sharing of assets. It also includes the sharing of a positive attitude toward universal design. Universal design may seem to require more thought and work at first, but the reward of creating a virtual environment that everyone can use is greatly satisfying. Make the achievement of universal design a clear goal for your team, and urge them to strive for it in everything they create. In Figure 1.5 there is an illustration displaying common practices for visual, audible, tactile, safety, and team design that can be used to create universal design in your game-based environment.



Common Design Practices for Universal Design

Visual Design
- Appropriate Level of Detail
- Visible to all-tested for color blind reader
- Text is readable by those with low vision
- Clarity is enhanced by environmental settings

Audible Design
- Sound clip is of highest quality possible
- Soundscape enhances meaning of landscape
- Visible signage is backed up with audio clip
- Sound levels allow for focus on specific objects
- Sound emitter is near default camera position

Tactile Design
- Visual texture supports overall message of the environment
- Sound texture creates overall mood
- Haptic interface used if possible

Safety Design
- Designation of purpose is evident within the space
- Definition of expected behavior made clear
- Design to inspire the appropriate behavior in the space

Team Design
- Positive attitude towards Universal Design in team
- Making the environment suitable for Universal Design is the goal
- Everything the team creates works towards Universal Design

FIGURE 1.5 Common design practices for universal design.

1.7 DESIGN WITH CODE FOR VIRTUAL ENVIRONMENTS

Procedural modelers are everywhere. Procedural modelers range from simple form generators like 3D fractal makers to complex stand-alone software packages that can generate complex terrain and cityscapes. In fact many 3D modelers like 3Ds Max, Blender, and Maya have built-in procedural programs such as procedural texturing (wood, marbles, clouds, etc.) and procedural modeling (terrain surfaces from noise, procedural creation of buildings with controller objects and customized properties). Creating a graphic/texture that wraps the surface of a complex form without showing ugly seams is what procedural textures do. For instance, with one of the procedural wood textures, the user simply selects the grain colors and the amount of grain detail they would like to see, and the whole object takes on a grained wooden surface. Creating complex forms like terrain or 3D fractals are made more manageable with procedural generators that plug into a 3D modeler or work as a stand-alone program. The designer initially sets up various parameters such as height, complexity, and scale, and the procedural generator does the rest. A designer can also create content in a virtual world through the use of scripting. Objects can be coded to create copies of themselves in various arrangements and scales, creating complex patterns on the ground or branching treelike structures. The sky is the limit with this kind of creation.

1.8 TAKING A DESIGN DISCOVERY VOYAGE

Design is a creative-thinking and problem-solving process. We are inspired to design by the need to communicate and to help others communicate across a wide spectrum of human understanding. Our communication may be visual, audible, and tactile, or involve the senses of smell and taste. But in order for it to be successful, we need to know how this communication is being perceived. All throughout this book are examples and projects that are about the communication and design involved in the creation of a game-based virtual environment. Virtual worlds are waiting to be explored, their peoples have messages to express, and designers are needed. Only the principles of design can be taught. To become a designer, you must embark on a voyage of self-discovery, compile an understanding of human behavior, and learn how to collaborate. You must embrace the part of yourself that is eternally curious and playful; this is where the ideas for great creations come from.

1.8.1 Course Study Planning with This Book

In Chapters 3 through 13 there are interrelated projects that ultimately conclude with the completion of a game-based virtual world environment. Some of them will require significant work, perhaps more than is achievable in one class period, even with the prebuilt content that will be available to download. Overall, this book could be used as a guide for one or two semester's worth of work for a class about virtual-game-based environments.

1.8.2 Self-Study with This Book

Although it is more fun to do these projects with other people, this book can also be used for self-study. The projects and content provided are designed to be manageable by a single individual. In fact, you could even create the game-based sim projects in this book in OpenSim with Diva Distro's latest download (http://metaverse ink.com/Downloads.html or http://simonastick.com), and carry it around with you on a USB drive.

1.9 CHAPTER SUMMARY AND FINAL THOUGHTS

In this introductory chapter we have skipped over the surface of several deep pools of thought, and hopefully this will encourage you to dive in and explore all the chapters in this book. One thing to keep in mind is that design is communication. No matter where you are designing on the reality–virtuality continuum or how much transrealistic designing you will be doing, what matters is the message: the message that your client wants to convey, the message that the users send back in response, and the meaning you add to the message as the designer of its conveyance. It is a big, interconnected world out there, the vizome adds to itself every day, strengthening and deepening our communication channels. We play games to communicate and learn on all levels of reality, and as a designer you will need to know how to make these universally accessible to all people. The communication you will design for is more than person to person, it also connects person to machine. Understanding how coding and scripting enhances your design process and how to work with people who specialize in that is crucial to your success as a designer of virtual environments. So, gather your favorite supplies, take a deep breath, and plunge right in.

REFERENCES

1. Darrell Etherington, “Apple Patents 3D Gesture Control Via Hover-Based Input on Touchscreen Devices Like the iPad,” *TechCrunch*, August 20, 2013, accessed May 19, 2014, http://techcrunch.com/2013/08/20/apple-patents-3d-gesture-control-via-hover-based-input-on-touchscreen-devices-like-the-ipad/.
2. *Wikipedia*, s.v. “Transrealism (literature),” accessed May 20, 2014, http://en.wikipedia.org/w/index.php?title=Transrealism_(literature)&oldid=631139126.
3. *Wikipedia*, s.v. “On Exactitude in Science,” accessed May 21, 2014, http://en.wikipedia.org/w/index.php?title=On_Exactitude_in_Science&oldid=592405114.
4. Doug Mann, “Jean Baudrillard: A Very Short Introduction,” accessed May 21, 2014, http://publish.uwo.ca/~dmann/baudrillard1.htm.
5. Jesse Schell, *The Art of Game Design: A Book of Lenses* (Elsevier, 2008), 26 (or Kindle edition, 1147).
6. Nick Yee, “Motivations of Play in Online Games,” *Journal of CyberPsychology and Behavior* 9 (2007): 772–775, accessed May 25, 2014, http://www.nickyee.com/pubs/Yee%20-%20Motivations%20(2007).pdf.
7. *The Stanford Encyclopedia of Philosophy*, s.v. “Aristotle,” by Christopher Shields, ed. Edward N. Zalta, accessed May 26, 2015, http://plato.stanford.edu/archives/spr2014/entries/aristotle/.
8. *Wikipedia*, “Reality Tunnel,” accessed May 25, 2014, http://en.wikipedia.org/w/index.php?title=Reality_tunnel&oldid=637824598.
9. Tom Simonite, “How Magic Leap’s Augmented Reality Works,” *MIT Technology Review*, October 23, 2014, accessed April 24, 2015, http://www.technologyreview.com/news/532001/how-magic-leaps-augmented-reality-works/.
10. *Wikipedia*, “WikiPando (tree),” accessed May 26, 2014, http://en.wikipedia.org/w/index.php?title=Pando_(tree)&oldid=631631828.
11. World Wide Web Size, accessed May 26, 2014, http://www.worldwidewebsize.com/.
12. Gilles Deleuze and Félix Guattari, *A Thousand Plateaus: Capitalism and Schizophrenia*, trans. Brian Massumi (University of Minnesota Press, 1987), 7.
13. James Howard Kunstler, *Home from Nowhere: Remaking Our Everyday World for the 21st Century* (Simon & Schuster, 1998), 130.
14. *Wikipedia*, s.v. “Crime Prevention through Environmental Design,” accessed May 28, 2014, http://en.wikipedia.org/w/index.php?title=Crime_prevention_through_environmental_design&oldid=613281484.
15. *Wikipedia*, s.v. “Talk: Crime Prevention through Environmental Design,” accessed May 30, 2014, http://en.wikipedia.org/w/index.php?title=Talk:Crime_prevention_through_environmental_design&oldid=572936402.

This page intentionally left blank

2 About the Viewers and Content Created for This Book

Language for me narrates the pictures in my mind. When I work on designing livestock equipment I can test run that equipment in my head like 3-D virtual reality. In fact, when I was in college I used to think that everybody was able to do that.

Temple Grandin

2.1 OVERVIEW OF ABOUT THE VIEWERS AND CONTENT USED FOR THIS BOOK

This chapter contains an overview of the "nuts and bolts" of virtual world viewers, the necessary computer gear for accessing the virtual environments. It is important for you to understand how viewers work, and how to set them up for access and security. It is equally important for you, the designer, to understand how computer platforms affect the design choices you will make. The concepts included in this chapter are

- Client viewers and their settings impact your design immensely.
- Learning how to choose a viewer is a necessary design skill.
- Design thinking should include the use of head-mounted displays.
- Designing for accessibility and safety can start with the viewer settings.
- Practicing design with 3D modular content teaches the fundamentals of good design.

2.2 VIRTUAL WORLD VIEWERS AND GAME INTERFACES USED IN THIS BOOK

For the purposes of creating this book, the Phoenix Firestorm Viewer versions 4.6.9 to 4.7.3 (Release x64 with OpenSimulator support) were used. This viewer is available for download at http://www.firestormviewer.org/. If you are going to do the projects in this book, it is recommended you download and install this viewer. You must use a mesh enabled viewer for the projects in this book as all the 3D content you will download for them is made from meshes. If you are new to working with meshes, the Second Life Wiki has lots of information about requirements for mesh here: http://wiki.secondlife.com/wiki/Mesh. The Phoenix Firestorm Project offers versions of their viewer tailored specifically for Second Life and OpenSim, so choose the one for whatever type virtual world you find yourself visiting most regularly.

2.2.1 Head-Mounted Display Viewers Used in This Book

There are some other viewers that were used with this book. The CtrlAltStudio Viewer (http://ctrlaltstudio.com/) was used for Chapter 13 to explore design for head-mounted displays (HMDs) such as the Oculus Rift DK2. The Second Life standard viewer (Second Life 3.7.28) as well as its Project Oculus viewer and the Project Experience viewers were also used now and then to compare functionality. The shared virtual reality environment called High Fidelity, which is now open for alpha use, provides its own interface, which is available at https://highfidelity.com/.

2.3 HOW THE CLIENT VIEWER IMPACTS VIRTUAL WORLD DESIGNING

Learning how to choose a viewer is a necessary design skill because client viewers such as Phoenix Firestorm, Second Life, and CtrlAltStudio impact your ability to design and build in a virtual world. Each viewer sets up its menu interface differently, making the creation and editing tools more or less accessible depending on your personal work methods. Some of the viewers have a higher level of accuracy such as Firestorm, which allows for object creation, modification, and building to the fifth decimal place. Some viewers, such as the Singularity viewer, are hybrids of older Second Life viewers and newer technologies. Some viewers group the tools into packages, such as the Phototools menu in Firestorm. The list goes on and on. As design choices are often driven by platform choice, knowing how to utilize the best viewer for any given platform is essential for a virtual environment designer. Let us look "under the hood" of an average viewer to see how that special program affects the following: (1) image rendering on your screen, (2) editing and building in the virtual world, and (3) sounds and communication. There are also some server-specific tasks that it does not perform directly that you should know about; however we will just look at the design-related features. If you would like a comprehensive overview, there is a long list of the interconnected parts in this software at http://wiki.secondlife.com/wiki/Viewer_Architecture. Please note, this list is the open architecture from Linden Lab's Second Life viewer 1.2.2, and because of the ongoing changes by Linden Lab, such as Server Side Appearance rendering for avatars (SSA 2013), the addition of the Communication Hub User Interface (CHUI 2013), and a new materials editor there have been changes in third-party viewers too, since they share a significant amount of code with the Second Life viewer. Please refer to Figure 2.1 for a general overview of the tasks being accomplished by the viewer while you are accessing the virtual environment. This diagram was developed from information on http://wiki.secondlife.com/wiki/Viewer_Architecture.

2.3.1 What the Viewer Does for Image Rendering and How to Set Up for It

One of the first things you should do as you set up your preferences in a new version of a client viewer is to turn on the Advanced and Developer menus on the top bar. To do this in Firestorm, go to Avatar/Preferences/Advanced and make sure that the Show Advanced Menu and Show Developer Menu is checked. Apply, choose OK, and close the Preferences window.

Let us take a look at some of the things these menus can show us. In Figure 2.2, there is a screenshot of these menus and their roll out submenu sections, as they appear in the Firestorm viewer.

Under the Advanced menu on the top bar, the submenus that directly impact your rendered image are called Rendering Types and Rendering Features. Try unchecking items on the sublist in each of these menus to see what it affects in your onscreen image. As you may notice, the viewer is breaking the rendered view into lots of categories. For instance, there is a rendering type for Tree and a separate one for Avatars. Under the rendering features list, you can toggle on/off such things as the UI (user interface), fog, and foot shadows. (Do not panic if you turn off the UI, Ctrl+Alt+F1 will turn it back on again in Firestorm.)

Under the Developers menu on the top bar, the submenus that provide an interesting look at what the viewer is doing are called Render Metadata and Rendering. Again try toggling these items on/off to see what the viewer adds or takes away from the screen.

Now, let us go over three of the major systems mentioned in the viewer architecture list, and think about how these can be leveraged to improve the rendering performance. These systems are called Culling, the Image System, and the Rendering System.

1. Culling is one way that the viewer reduces the rendering tasks for your graphics card. By eliminating the rendering of any obscured elements, such as the objects outside of the room you are in, it cuts down on the information that needs to be processed. The efficiency of the culling system is

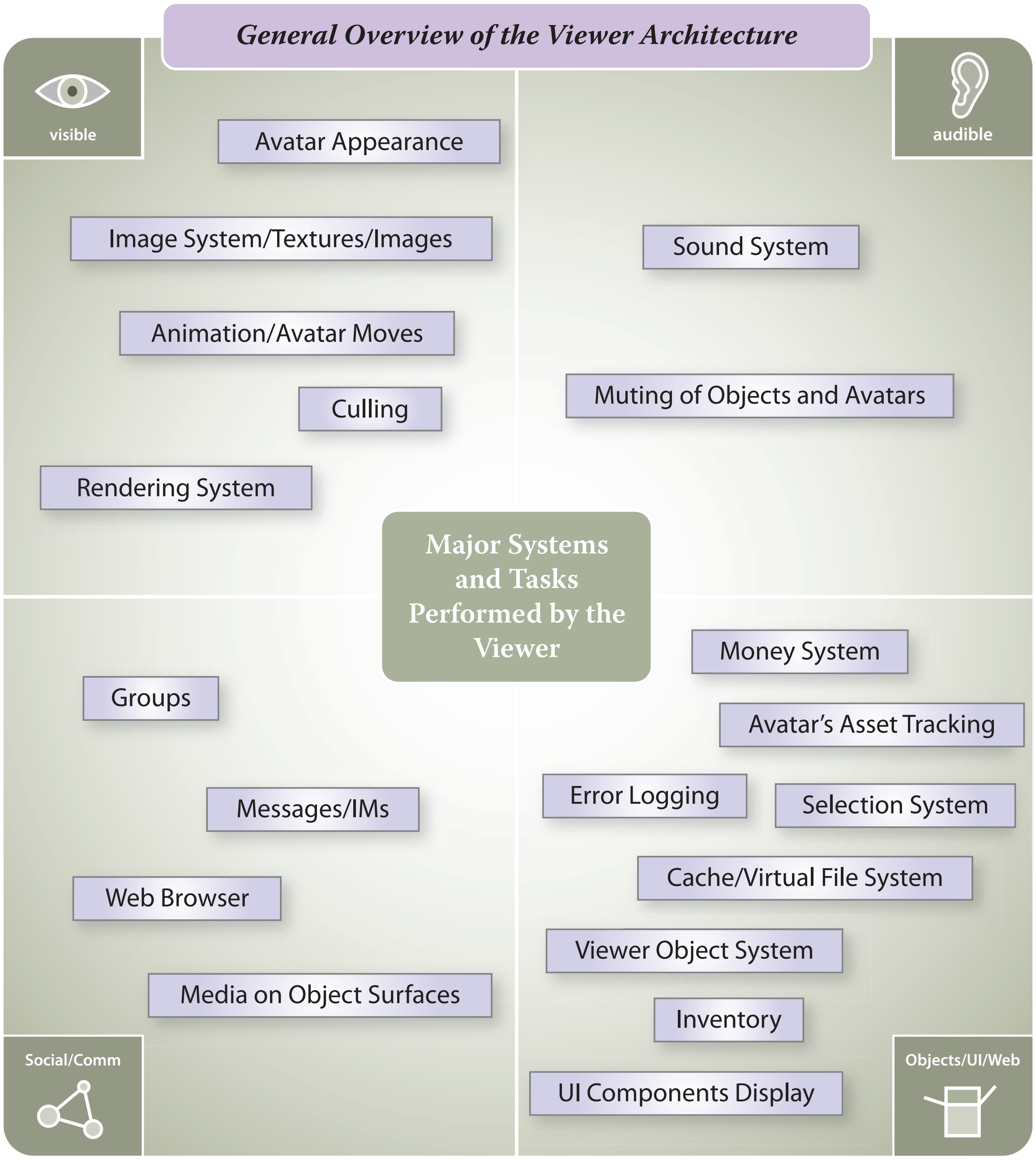
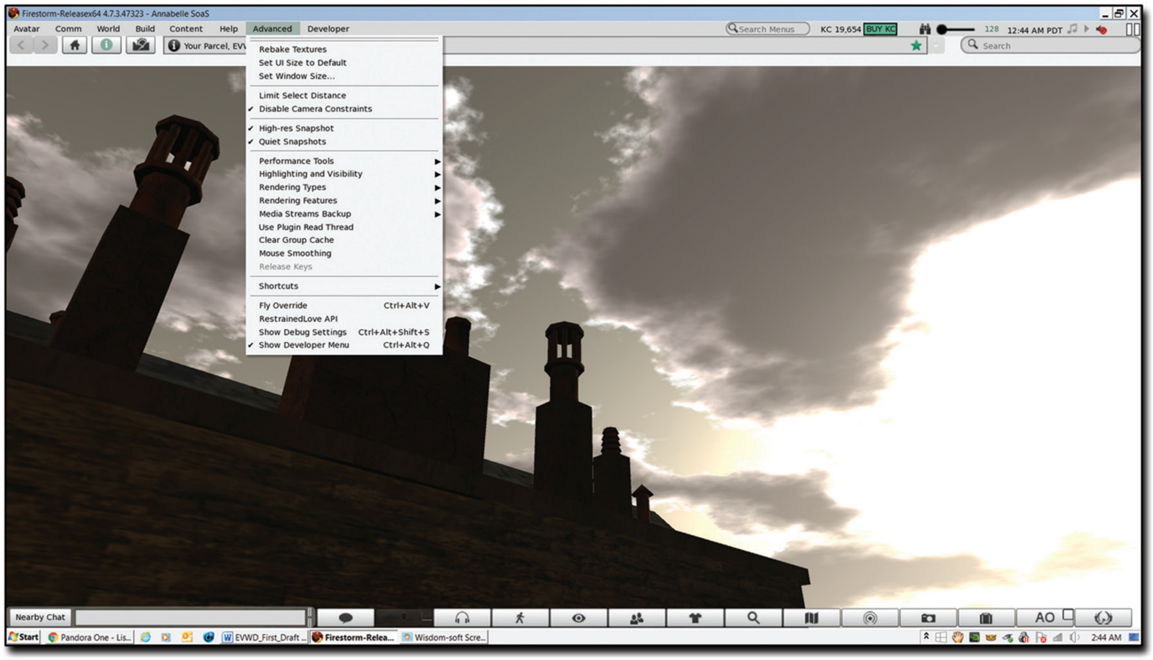


FIGURE 2.1 Overview of the client viewer architecture showing systems and tasks performed by the simulator.



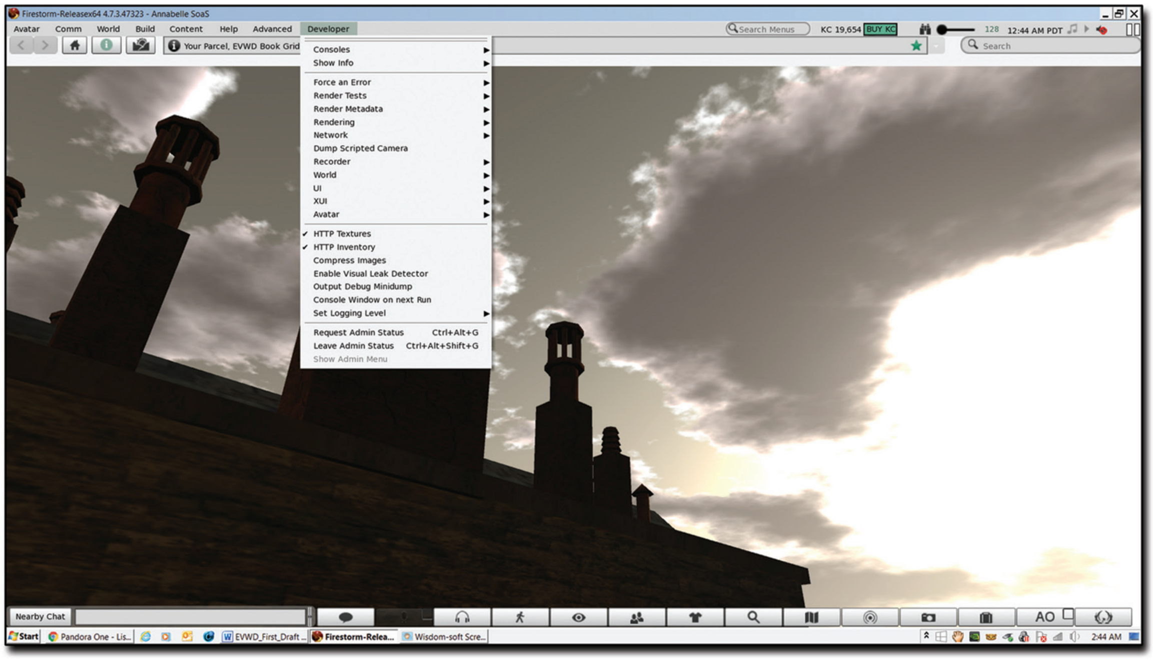

Firestorm-Releasex64 4.7.3.47323 - Annabelle SoaS
Avatar Comm World Build Content Help Advanced Developer
Your Parcel, EVWD Book Grid
Consoles
Show Info
Force an Error
Render Tests
Render Metadata
Rendering
Network
Dump Scripted Camera
Recorder
World
UI
XUI
Avatar
✓ HTTP Textures
✓ HTTP Inventory
Compress Images
Enable Visual Leak Detector
Output Debug Minidump
Console Window on next Run
Set Logging Level
Request Admin Status Ctrl+Alt+G
Leave Admin Status Ctrl+Alt+Shift+G
Show Admin Menu
Nearby Chat

FIGURE 2.2 Advanced and Developer menus in the client viewer (Firestorm).

often hampered with the use of transparent graphics, so they should be used judiciously in your environments. If you need to keep rendering performance up, think about removing that perfectly transparent window or replacing it with an opaque graphic/texture. Another thing to watch out for is unintentional openings in your building walls; a small crack or sliver of space between the architectural elements deactivates the culling process on the items outside of the room.

2. The Image System in the viewer uses highly compressed versions of your graphics to move them from your avatar's cache to OpenGL so they can be displayed. By creating the smallest possible graphics for your content's textures, that is, 512 pixels by 512 pixels maximum, you can enhance the performance of this system.
3. The Rendering System instigates the series of events required for the rendering of geometry as the camera moves about the scene; it creates the camera frames that allow for the real-time rendering and the illusion of reality. It also performs render passes for material qualities such as glow and bumpiness, and it calculates information to produce shadows, lights, and fog. An awareness of all the functions of the rendering system will help you trim the nonessential qualities and adjust the efficiency of your scene when the frame rate gets too low for realistic movement or the machinima you are recording.

2.3.2 What the Viewer Provides for Building and Editing

All of your design tools and toolsets are located and accessed in the viewer. Included within are the creation, selection, and manipulation of the geometry and materials of objects; the terraforming tools; mouse pointing and looking; and the mouse/screen coordination inherent with combining these functions, such as a drag/copy move. The gun tool is also here to provide crosshairs for targeting purposes.

2.3.3 What the Viewer Provides for Communications and Sound

Your communication preferences for text chat and notifications are controlled via the Preferences/Chat menu under the Avatar tab on the top bar of the viewer interface. Here you can set up the radar settings too, which enables a level of security for your avatar. Obviously most of this is a personal choice, and the settings can be highly customized. Communication settings on the viewer interface that are important for the designer would obviously be those that allow for clearer discussion of the project as well as providing access to confidential channels. The "switchboard" that organizes all the text chat and instant messaging (IM) is in the actual simulator server itself, which supports the entire region. Settings for the sound used in your virtual environment are located in the Avatar/Preferences/Sound & Media tab. Under that tab you can customize all the sound you hear and how you hear it (camera position or avatar position) as well as setting up your avatar to display speaking animations.

2.3.4 What the Simulator Server Does Compared to the Viewer

Supporting every region in Second Life and OpenSim is the simulator or simulator server. This server process concerns itself with running the physics engine, detecting collisions between objects, terrain, and other avatars as well as keeping tabs on where every object is located. The simulator server will send updates to the viewer as things change. The viewer updates the location of objects as they appear on screen and does some simple physics calculations to track movement, but does not concern itself with collision calculations. A virtual world designer should always be aware that there is a simulator running these calculations underneath and that large amounts of physical objects, collisions, avatars, and script can overwhelm the processor

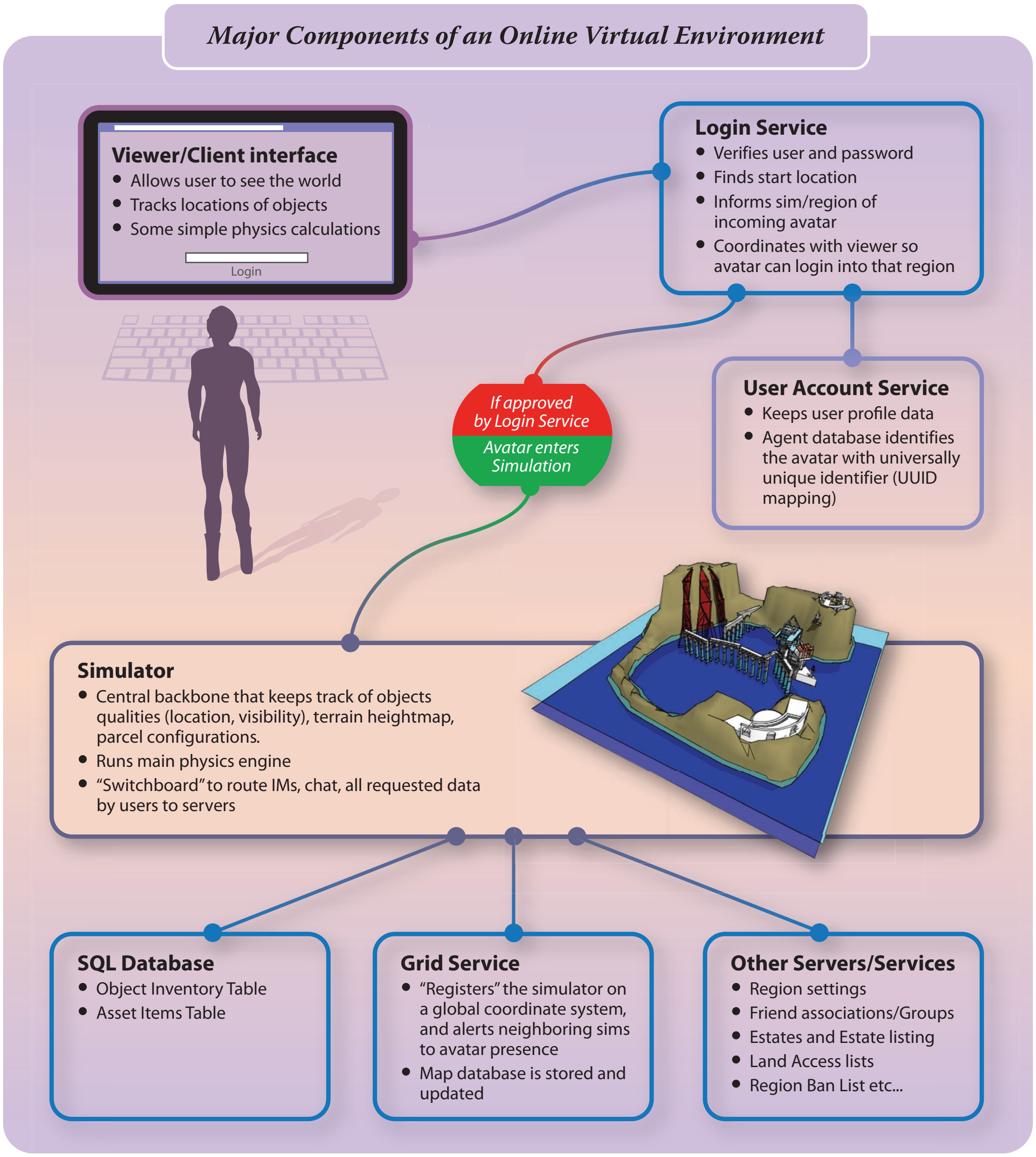

Major Components of an Online Virtual Environment
Viewer/Client interface
• Allows user to see the world
• Tracks locations of objects
• Some simple physics calculations
Login
Login Service
• Verifies user and password
• Finds start location
• Informs sim/region of incoming avatar
• Coordinates with viewer so avatar can login into that region
If approved by Login Service
Avatar enters Simulation
User Account Service
• Keeps user profile data
• Agent database identifies the avatar with universally unique identifier (UUID mapping)
Simulator
• Central backbone that keeps track of objects qualities (location, visibility), terrain heightmap, parcel configurations.
• Runs main physics engine
• "Switchboard" to route IMs, chat, all requested data by users to servers
SQL Database
• Object Inventory Table
• Asset Items Table
Grid Service
• "Registers" the simulator on a global coordinate system, and alerts neighboring sims to avatar presence
• Map database is stored and updated
Other Servers/Services
• Region settings
• Friend associations/Groups
• Estates and Estate listing
• Land Access lists
• Region Ban List etc...

FIGURE 2.3 Interconnections between a client viewer and the virtual environment.

and slow the screen updates to a standstill. There is a lot more information about the simulator server and all the other sorts of servers that support the virtual world here in this wiki: http://wiki.secondlife.com/wiki/Server_architecture. For a diagrammatic overview of what the viewer does to help you view and interact with a virtual environment, please refer to Figure 2.3.

2.3.5 Project Snowstorm and Viewer Development

Back in 2010, Oz Linden, the director of Open Development from Linden Lab, started "Project Snowstorm." This ongoing project is dedicated to coordinating viewer development with the open source development community. With a process of proposals, reviews, beta tests, and bug tracking, his team develops new features in the Second Life open source viewer. If the proposed features from third-party developers meet the criteria and vision that Linden Lab has for its Second Life viewer, the process of development may lead to the eventual inclusion of these new features into the viewer code. Other groups at Linden Lab are working on the development of new Second Life-based viewers [1]. This includes a group called RIFT, who are developing a Second Life viewer for the Oculus Rift, a head-mounted display (HMD). Information about the kind of activity going on in each development group is available on the Bug Tracker site found here: https://jira.secondlife.com/browse/STORM.

2.4 SELECTING A VIEWER FOR SECOND LIFE AND OPENSIM

Even though the Firestorm viewer was the "go to" viewer for this book, this is not your only option. Accessing an array of viewers is actually a good idea from a design standpoint because that will expose you to the wide selection of custom tools each of the third-party viewer developers have created. A comprehensive list of all third-party viewers is provided at http://wiki.secondlife.com/wiki/Third_Party_Viewer_Directory. Take a look at that list and then consider the following questions as you go about selecting a specific viewer for the design job you have in mind.

1. What is your computer system? Mac, PC, or Linux? If your team has a mixed-platform profile, your best bet is to choose client viewers that support all three platforms.
2. Will you be doing lots of building and creating? If so, then pick from among the viewers designed for that purpose, like Firestorm, Dolphin, Catznip, and Singularity.
3. Do you still love the old Second Life viewer and want that look and feel? That would point the direction of your choice to Singularity, which incorporates the new code into an older interface.
4. Do you have to create large-scale projects under pressure? Then, you need a viewer with the most stability and lowest crash rate. That would be Firestorm, followed by the Standard SL viewers.

From time to time, you should review the list of third-party viewers to see if there are any new offerings.

2.4.1 Viewers for Use with a Head-Mounted Display (HMD)

Design for virtual environments experienced with an HMD involves an interface that blocks your view of the desktop, keyboard, and mouse. The viewers for that kind of display are developing concurrently with the display hardware, and all sorts of interesting interfaces are bound to appear in the next few years. In Chapter 13 we will explore some aspects of design that you will have to consider for a game-based virtual environment that will be experienced on a head-mounted display. For Chapter 13, both the SL Viewer Second Life 3.7.8 (289834) May 7 2014 20:56:44 (Second Life Project OculusRift) and the CtrlAltStudio Viewer 1.2.1

(41169) May 16 2014 12:36:17 (CtrlAltStudio-Viewer-Release) with OpenSimulator support were used. The CtrlAltStudio Viewer can be downloaded from https://ctrlaltstudioviewer.codeplex.com/.

2.4.2 Considering Input Devices for Safety and Universal Access

We may look back on the first decade of this century as the dawn of a golden age for computer-based interactive devices. Never before have we had such wide-ranging speech to text availability, or been able to use our gestures and thoughts to move 3D objects in virtual space. Everywhere, new kinds of input devices are being offered to the computer user, and this has benefited universal access along with the development of universal design in virtual environments. The key interface for the majority of the virtual world users, Firestorm, can also be a powerful tool for setting up a safer virtual environment and universal access. Let us look at how this can be done.

2.5 VIEWER SETTINGS TO ENHANCE YOUR SAFETY

Woven throughout the menu interfaces of Firestorm are all sorts of security-enhancing features. Obviously nothing is going to replace good common sense and knowing the rules of online safety, but that level of protection can be enhanced with the use of safety settings in the top menu bar tabs named Avatar, Comm, and World. Table 2.1, is a list of menu items that can be utilized to add more online security and privacy settings for your avatar in the virtual worlds. You should test drive these settings until you have found a good balance between enough security, privacy, and accessibility for your avatar.

2.6 VIEWERS, SETTINGS, AND CONSIDERATIONS FOR UNIVERSAL ACCESS

Virtual worlds help people feel free of daily troubles and toil. This is especially true of people with disabilities, who have found the capacity to freely move their avatars bodies to be an inspiring and healing experience [2]. Second Life, OpenSim, and other virtual worlds offer significant international social contact for those who are housebound, immersing them in a common experience that lets them explore the world. One of the first groups to recognize the need for accessibility in the virtual space was Virtual Ability (http://www.virtualability.org/). Led by Alice Krueger (known as Gentle Heron in Second Life), this group is involved with various projects including the promotion of universal design for virtual worlds. Universal design for the real world cannot be directly translated into the virtual world; however some design techniques can apply. As the leaders of Virtual Ability state: “We advocate that, in most virtual world situations, designers incorporate accessibility principles that are appropriate and germane for that virtual world, and not be constrained to a literal translation of Real World standards and requirements. We conclude that this is a definition of Universal Design of Virtual Worlds” [3]. The Radegast viewer is one example of a viewer designed to increase accessibility. The developers of the Radegast viewer have created a text-based viewer for people whose eyesight is limited. Find more information at http://radegast.org/wiki/Accessibility_Guide. If you are designing with the Firestorm Viewer and you want to increase accessibility for your visitors, there are several options as well. In Table 2.2 you will see menu settings for the Firestorm viewer that will help you make the first steps toward increasing accessibility in virtual worlds.

2.7 SOME NOTES ABOUT THE GEAR

These days, virtual worlds can be run on everything from a smartphone to a desktop computer, and on the three main platforms of Mac, Windows, and Linux. In fact, with the trend going more toward mobile, it is

TABLE 2.1
Viewer Settings to Enhance Safety

Location in Viewer Menu Tabs	Settings and Aspects of Security Involved
Avatar/Preferences/General Tab	Content Maturity Rating
	Log out after being away
Avatar/Preferences/Chat Tab	E-mail me IMs when I'm offline
	Mark objects with (no name) when they speak to avoid spoofing
	Radar
	Keyword Alerts
Avatar/Preferences/Sound & Media Tab	General Tab/Mute when minimized
	Voice Settings Tab/Enabled Voice
Avatar/Preferences/Move & View Tab	View/Disable camera constraints
	Movement/If movelock built-in into LSL bridge is active
	Movement/Double click on land to teleport
Avatar/Preferences/Notifications Tab	Notify me when:
	Friends log in or out
	When the region's total script count changes
	Report collision notifications in nearby chat
Avatar/Preferences/Privacy Tab	General Tab—consider all settings here
	Save (chat logs) Tab—consider all settings here
	LookAt Tab—consider all settings here
	Autoresponse Tab—consider the settings here
Avatar/Preferences/Advanced Tab	Allow Advanced Menu and Developer Menu to show
Avatar/Preferences/User Interface Tab	General Tab—consider all settings here
Avatar/Preferences/Firestorm	Protection Tab—consider all settings here
Avatar/Preferences/Backup	Consider all the settings here
Avatar/Movement/Move Lock (Ctrl+Alt+P)	On push enabled regions, this will lock your avatar in place
Avatar/Avatar Health/Stop avatar animations	Will stop any unwanted animations from being performed by your avatar
Avatar/Avatar Health/Undeform Avatar	Will work to help reform your avatar back to its proper shape after a griefer attack or use of over scale avatar body
Comm/Online Status	Consider the settings in this category
Comm/People	Nearby (Radar)—consider settings in this section
	Friends—consider settings in this section
	Groups—consider settings in this section
	Recent—consider settings in this section
	Blocked—consider settings in this section
	Contact Sets—consider settings in this section
Comm/Voice Morphing	Voice Filters (Linden Labs product/Second Life only)
Comm/Conversation Log	Conversation Log—consider settings in this section
World/Nearby Avatars	Accesses the Nearby Tab inside the People window
World/Radar	Accesses the Radar window
World/Mini-Map	Accesses the Mini Map
World/World Map	Accesses the World Map

(*Continued*)

TABLE 2.1 (CONTINUED)
Viewer Settings to Enhance Safety

Location in Viewer Menu Tabs	Settings and Aspects of Security Involved
World/Show More	Consider the settings in this submenu for accessing more information about the current location of your avatar
World/Teleport Home (Ctrl+Shift+H)	Instant teleportation to your home location
World/Area Search	Area Search window allows access to information about the whereabouts of all objects in the region
World/Asset Blacklist	Derendering and Blacklisting menu allows access to control over visibility of avatars and objects around you
Help/Report Abuse	Reporting abuse window provides format for reporting instances of griefing or other sorts of abuse in Second Life
Help/Bumps, Pushes & Hits	Bump, Push & Hit window allows access to information about pushes against your avatar from other avatars, and hits and pushes from projectile objects fired at your avatar

Source: Data from the Phoenix Firestorm Project Wiki, "Firestorm Top (Menu) Bar," accessed May 31, 2014, http://wiki.phoenixviewer.com/archive:fs_menus_fs451.

TABLE 2.2
Firestorm Viewer Settings to Enhance Universal Access

Location in Viewer Menu Tabs	Settings and Aspects of Universal Access Involved
Avatar/Preferences/Chat/General Tab	Chat window font size
	Onscreen console font size
Avatar/Preferences/Color Tab	Chat Color—consider settings for clarity of color perception
	Name Tags—consider settings for clarity of color perception
	Minimap—consider color settings for clarity of communication
Avatar/Preference/Graphics Tab	Many settings here will improve the visibility of the rendered image and are system dependent, so they should be adjusted for your system specifically
Avatar/Preferences/Sound & Media Tab	Many settings here will improve the audio quality of the virtual environment and are system dependent, so they should be adjusted for your system specifically
Avatar/Preferences/Move & View Tab	Movement/Joystick Configuration—consider settings for movement with devices other than a mouse or keyboard
Avatar/Preferences/User Interface Tab	General Tab—consider changing the UI scaling to increase the UI size
	Font Tab—consider changing font for better readability
Avatar/Preferences/Skins Tab	Consider changing the skin of your viewer for greater contrast and readability

Source: Data from the Phoenix Firestorm Project Wiki, "Firestorm Top (Menu) Bar," accessed May 31, 2014, http://wiki.phoenixviewer.com/archive:fs_menus_fs451.
Note: These settings can be used in conjunction with other software and devices to increase the level of accessibility.

likely that we will all be accessing the virtual spaces of the future continually while we are moving about. How you live and access virtual worlds is probably the single most important factor in deciding what kind of gear you will need to get. If you are going to design these worlds as well, then you will need to view your work on the same platforms as your visitors to test the effectiveness of your creative efforts.

2.7.1 Building Your System for Your Design Needs

Second Life lists a set of system requirements for all three operating systems at https://secondlife.com/support/system-requirements/. In general, having a speedy CPU, a good strong graphics card, a fast Internet connection, and an adequate memory are the essential elements. After that it is your choice for bells and whistles. Throughout this book, there has been an effort to be platform agnostic. You may be a PC user or a Mac or Linux-based system user, and there are advantages and disadvantages to each platform in terms of virtual world design. In general, all key commands in this book are given with the PC in mind; the alternative keys for the Mac are given in Table 2.3. Whatever computer system you decide to use, it should be a good quality one backed up by warranties. Virtual world image creation is demanding on a computer system, and you may find that your settings in terms of lighting/shadows and draw distance cannot be the same as other users if your system is old or on the slower side of processing speeds.

2.7.2 Computer Specifications from the Equipment Used for This Book

For the creation of this book, a custom-built computer by JNCS (http://www.JNCS.com) was used. Its operating system is Windows 7 Professional, with an Intel Core i7-4770K CPU at 3.5 GHz. The system RAM is 16 GB, and it is running a 64-bit setup. The graphics card is an NVIDIA GeForce GTX 680.

2.7.3 Some Notes about the Peripheral Gear

Now, onto the “bells and whistles” part. Used in connection with the gear described earlier are the following devices: a LeapMotion Mouse, a 3Dconnexion SpaceNavigator mouse, and a developer kit (DK2) Oculus Rift HMD. Each of these devices will work with Second Life and OpenSim. The LeapMotion mouse and the Oculus Rift are under heavy development by the worldwide gaming community. Also used are various headsets with microphones including the Sennheiser PC350 SE and a Logitec USB version.

TABLE 2.3
PC Keyboard Commands and the Mac Alternatives

PC Commands (Windows)	Apple/Mac Commands (Mac System)
Control (Ctrl)	Command (for Most Shortcuts) or Control
Alt	Option key
Windows/Start	Command/Apple
Backspace	Delete
Shift	Shift
Delete	Del
Enter	Return
Right click	Hold Control key and click
Zoom with mouse middle wheel	Control and scroll

2.8 HOW TO GET AND UPLOAD THE CONTENT FOR THIS BOOK INTO YOUR VIRTUAL WORLD

Every effort has been made to provide good-quality content for use in the projects outlined in this book. However, since they have no control over how this content is used, modified, or redistributed once released into the world, the book's author, Ann Cudworth, the content creator, Tim Widger, and any members of the publishing company CRC Press are not responsible for anything that happens to your computer or virtual world in which you are working while you are using this content or to anyone you have given it to. The content will be periodically reviewed and updated if necessary, and update notes will be placed on the Ann Cudworth Projects site, http://www.anncudworthprojects.com/.

2.8.1 Where to Get the Content

The primary source for content for this book will be http://www.anncudworthprojects.com/ in the Extending Virtual Worlds Design Book Download section, organized by book chapter. Information will be posted on the blog and other pages on the site as tutorials are developed, and reader feedback posted. Content will also be available in Second Life and possibly some other grids, as the need develops, and notice of such availability will be made on the website and through social media channels. Feel free to like Ann Cudworth Projects on Facebook, and follow Ann Cudworth Projects on Google+ and @afanshaw on Twitter.

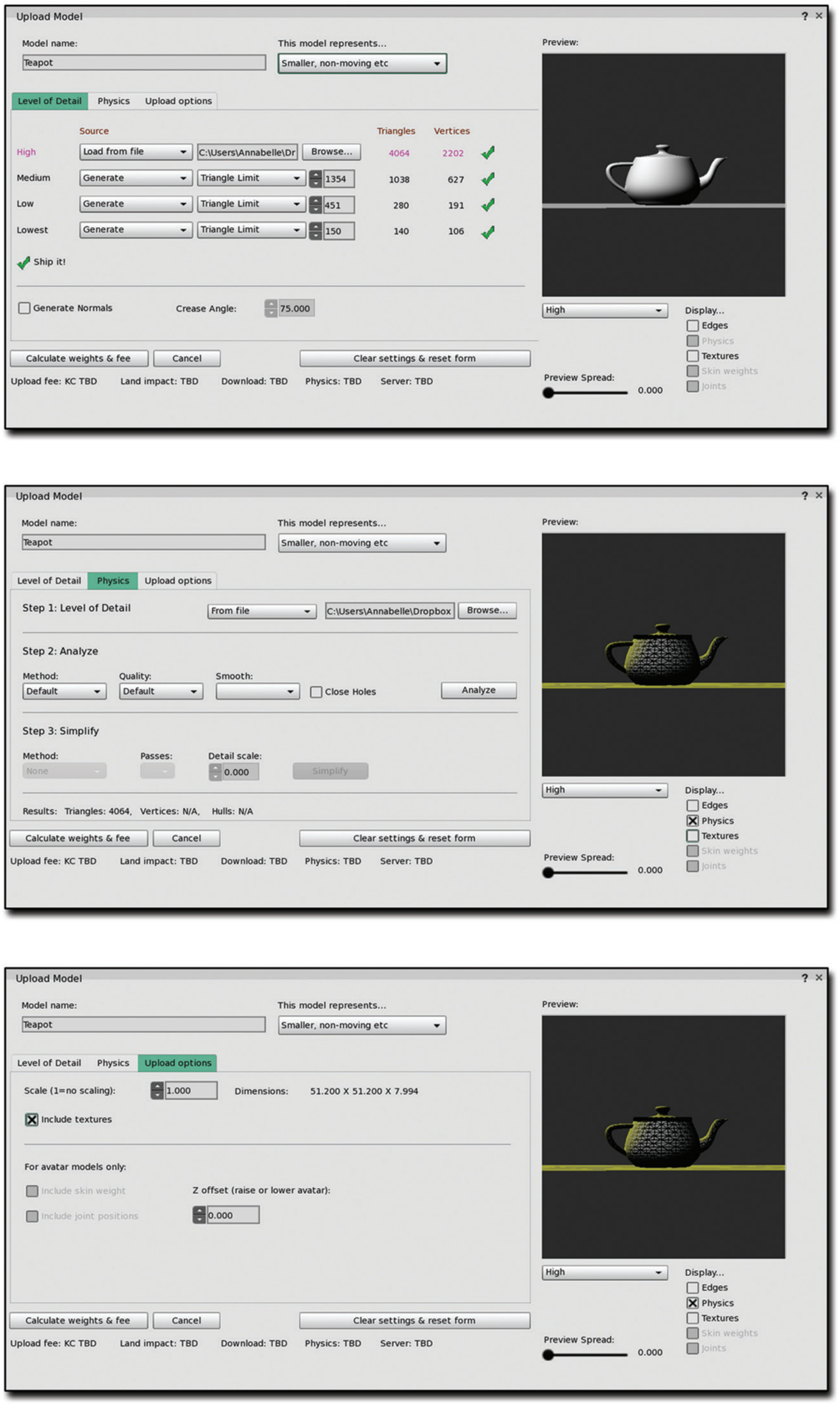
2.9 SETTINGS GUIDELINES FOR UPLOADING THIS BOOK'S CONTENT INTO YOUR VIRTUAL WORLD

It is highly suggested that you name each mesh model file you upload to your virtual world from the Ann Cudworth Project site with exactly the same name as the original file to avoid confusion while you are working on the projects in these chapters. If you are uploading the 3D content from this book into OpenSim or Second Life as a COLLADA.dae (Digital Asset Exchange) file, please be aware that download times may vary significantly depending on your bandwidth, processor speed, and such. Some of these models are large and will take a while to load into your virtual environment. The following suggested guidelines have been written to help you with the content uploads from this book. You can access the upload menu from the Avatar/Upload/Mesh Model rollouts on the Firestorm Viewer interface; this will bring up a menu that looks like the image in Figure 2.4. Once you have selected the model you want to bring into your virtual world, the Upload Model menu will appear. Notice the three tabs just under the model name called Level of Detail, Physics, and Upload Options.

In the following sections are suggestions for the settings you can use in each of these tabs while uploading a mesh model to Second Life or OpenSim. Note: There is a more information about this process in Chapter 7, Section 7.5.

2.9.1 Level of Detail (LOD) Setting Guidelines

In Second Life for High source setting use "Load from file," for the Medium source setting it should be "use LOD above," and for Low and Lowest source settings use the "Generate" setting. This gives you a good resolution on the model at most distances, without boosting the Land Impact or costing too much in Upload fees. In OpenSim, where the Uploads are free of charge, you can use "Load from file" settings for High and Medium LOD uploads, and in the Low source use "use the LOD above," keep the Lowest at "Generate."



Upload Model

Model name: Teapot

This model represents... Smaller, non-moving etc

	Source			Triangles	Vertices	
High	Load from file	C:\Users\Annabelle\Dr	Browse...	4064	2202	✓
Medium	Generate	Triangle Limit	1354	1038	627	✓
Low	Generate	Triangle Limit	451	280	191	✓
Lowest	Generate	Triangle Limit	150	140	106	✓

✓ Ship it!

☐ Generate Normals Crease Angle: 75.000

Upload fee: KC TBD Land impact: TBD Download: TBD Physics: TBD Server: TBD

High Display... ☐ Edges ☐ Physics ☐ Textures ☐ Skin weights ☐ Joints

Preview Spread: 0.000

Upload Model

Model name: Teapot

This model represents... Smaller, non-moving etc

Step 1: Level of Detail From file C:\Users\Annabelle\Dropbox

Step 2: Analyze

Method: Default Quality: Default Smooth: ☐ Close Holes

Step 3: Simplify

Method: None Passes: Detail scale: 0.000

Results: Triangles: 4064, Vertices: N/A, Hulls: N/A

Upload fee: KC TBD Land impact: TBD Download: TBD Physics: TBD Server: TBD

High Display... ☐ Edges ☒ Physics ☐ Textures ☐ Skin weights ☐ Joints

Preview Spread: 0.000

Upload Model

Model name: Teapot

This model represents... Smaller, non-moving etc

Scale (1=no scaling): 1.000 Dimensions: 51.200 X 51.200 X 7.994

☒ Include textures

For avatar models only:

☐ Include skin weight Z offset (raise or lower avatar): 0.000

☐ Include joint positions

Upload fee: KC TBD Land impact: TBD Download: TBD Physics: TBD Server: TBD

High Display... ☐ Edges ☒ Physics ☐ Textures ☐ Skin weights ☐ Joints

Preview Spread: 0.000

FIGURE 2.4 Upload Model menus in the client viewer.

2.9.2 Physics Settings Guidelines

Use this for both Second Life and OpenSim when you are uploading meshes.

For “Step 1: Level of Detail” (for the physics collision shape), select “From file” and load in the model’s physics.dae file. This physics file will have a name that matches the mesh model. This is a simplified version of the mesh model structure, with no textures on it, which will be used by the mesh model as a physics collision shape. Please note if you use the Upload menu for the generation of a physics shape; it is a trial-and-error process. You will save more time if you make your own simplified version of your mesh model, and use that for a physics shape file.

Leave Step 2 and Step 3 alone on this menu tab for these uploads.

Note: In Second Life, the physics shape type on a mesh is set to “convex hull” under the Features tab of the Build/Edit menu, by default. You will need to change this to “prim” for every mesh you upload. The prim setting for physics shape type is the default in OpenSim.

2.9.3 Upload Options Settings Guidelines

Use the scale spinner to enlarge/reduce the mesh model on upload if the project calls for it.

Texture (graphics) guidelines to follow on upload: The graphics for the model must be in the same folder as the model in order for them to be applied as textures to the surfaces of the model during an upload. These images will be included with the model, as long as the “Include textures” box is ticked when you upload.

2.9.4 Known Problems with Uploads in Second Life and OpenSim

Make sure that mesh models uploaded in Second Life are set to “prim” under Physics Shape type in the Features section of the Build/Edit menu. In both Second Life and OpenSim, mesh models that have multiple parts may not be uploaded with the same key or main prim the mesh model was initially created with. If you are uploading a mesh model and need a specific part to be the key prim (outlined with yellow) in the link chain, you may need to unlink it and re-link it when the mesh is rezzed inworld. Loading models from SketchUp into Second Life can be problematic. Even though SketchUp will save a COLLADA.dae file that will upload to Second Life, you may find that the physics shape will not function properly. If you are going to make many mesh models for Second Life, it is recommended that you use a modeler like Blender or 3ds Max. There seems to be no such problems in OpenSim with uploading a mesh model from SketchUp; it will take a COLLADA.dae from SketchUp as well as the physics “From file” setting. Sections of a model that were built with a transparent material on them may become opaque because of a state change in the upload. To make them transparent again, go into Edit Linked and select that part, not the whole linked object. Then, in the Edit menu, under the Texture Edit tab, select the element and change the transparency to what you need.

2.10 LICENSING INFORMATION

All the 3D model content, Ann Cudworth’s sound content, and other content that she, Tim Widger, and Michael Thome have created for this book are provided under the Creative Commons Attribution 3.0 license (http://creativecommons.org/licenses/by/3.0/deed.en_US), which means you are licensed to share it (to copy, distribute, and transmit the work), remix it (adapt the work), and make commercial use of the work, under the following condition. You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that the author or licensor endorse you or use of the work). A suggested form of attribution is "*Extending Virtual Worlds* is a design book written by Ann Cudworth, model created by Tim Widger, and scripted by Michael Thome in 2015" in the description box of the mesh model you are going to use from this book in your virtual world.

2.11 HOW TO USE THE LSL SCRIPTS PROVIDED

Regarding the utilization of the LSL (Linden Scripting Language) scripts provided with this book, please follow these guidelines. Each script, shown in the tables of this book, is available in the Extending Virtual Worlds Design Book Downloads section, listed by chapter, on the website http://www.anncudworthprojects.com/. If you are going to use the LSL script take it from there as a WordPad or text file, not from the tables in the book. The tables have extra enumeration that will not be needed inworld when the script is going to run. Some of these scripts have been created by other people, such as the LSL scripts in Chapters 8 and 12, are under their own licenses and provided for use under those terms. For the most part, this means keeping the header of the script intact, so the original source of the script is known. Read each script to be sure you know its specific license terms before you use it.

2.12 CHAPTER SUMMARY AND THOUGHTS

We have covered a lot of varied ground in this chapter. Starting with a discussion of what the client viewer can do and how it can be utilized as part of universal design, accessibility, and safety, we moved on to look at the viewers used for head-mounted displays like the Oculus Rift. After a quick look at the kinds of computer gear used to gather images and information for this book, we ended this chapter with a look at how and where the content for all the upcoming projects can be procured. This chapter has much information in it that you will want to refer to again throughout your progress in the book. Good luck with the projects as you set about working with the design and construction of a game-based virtual environment.

REFERENCES

1. "Linden Lab Official: Alternate Viewers," *Second Life Wiki*, accessed June 3, 2014, http://wiki.secondlife.com/wiki/Linden_Lab_Official:Alternate_Viewers.
2. Wagner James Au, "Second Life's Best (Perhaps Only) Killer App: An Immersive, Open-Ended Social Platform for the Disabled," *New World Notes*, February 13, 2013, accessed May 31, 2014, http://nwn.blogs.com/nwn/2013/02/second-life-killer-app-help-disabled.html.
3. Alice Krueger, Ann Ludwig, and David Ludwig, "Universal Design: Including Everyone in Virtual World Design," *Journal of Virtual Worlds Research* 2, no. 3 (2009), accessed May 31, 2014, http://journals.tdl.org/jvwr/index.php/jvwr/article/view/674/524.

This page intentionally left blank

3 The Vizome, Developing Creativity, and Virtual Environmental Design

Too many brands treat social media as a one way, broadcast channel, rather than a two-way dialogue through which emotional storytelling can be transferred.

Simon Mainwaring

3.1 OVERVIEW OF THE VIZOME, DEVELOPING CREATIVITY, AND VIRTUAL ENVIRONMENTAL DESIGN

As a virtual world designer, you are a poetic architect [1]. Your design approach, or point of view, is formulated from observations of life and creative expression. You work with the aesthetic proportions of architecture, landscape, clothing, and character avatars. Your understanding of the underlying meanings and metaphors of form, shape, and color are intrinsic to what you create. As a designer, you will work with ideas big and small, hidden and obvious, on many levels simultaneously, as you explore the underlying interconnections of a virtual environment, which we shall call the vizome, a portmanteau of *virtual* and *rhizome*. Introduced in Chapter 1, Section 1.5.1, the vizome represents all that connects us in a virtual world. It is the stuff that our virtual world experience is made from, and as a designer you will be immersed in it. Following are some key ideas related to the process of designing within the vizome, which we will explore in the following sections of this chapter.

- Discovering the existence of the vizome is important to designing effectively.
- Exploring the meaning inherent in the vizome enriches virtual world design.
- Developing a "creation system" will support your design process.
- Social interactivity is part of design for virtual worlds.
- A design response or creative approach is subject to your point of view.
- Creativity is a resource that can be fostered and developed in all designers.
- Creativity requires time, emotional space, and intellectual mindset.

At the end of this chapter we will begin a series of interrelated projects intended to introduce the various principles of extended virtual world designing to you. Extended design goes beyond the tenets of basic design and takes a deeper dive into the subjective and personal approaches to designing a virtual environment. Starting with this chapter and progressing through the book, you will be challenged with the creation of a game-based virtual environment, an environment that reaches out to deliver a message to your visitors and expresses your creative concepts for game play in a virtual world.

3.2 DISCOVERING THE VIZOME

As mentioned in Chapter 1, Section 1.5.1, the term *vizome* is a portmanteau of *virtual* and *rhizome*. A rhizome is a botanical term for a shared root structure, and virtual signifies the nonphysical world that we create with the use of computers. You may initially conclude that a vizome refers to the interconnecting structures of the Internet, the servers, mobile devices, desktop computers, software, and so on. That is one concept of how the system works. However, in our designer-based context, the vizome refers to the visitor's experience in a virtual world, as well as the movement of objects, ideas, and memes in a virtual environment. If you break away from hierarchical definitions of design and allow yourself to aggregate the influences of the whole experience in a virtual environment, then you can let go of a traditional "solve the problem with an answer" frame of mind as you approach your next design project. You can enter or immerse yourself in the design of your virtual environment, becoming an influence across the network of connections. Think of your designs in the virtual environment in a new way, as part of the vizome, the virtual rhizome.

3.2.1 Early Concepts That Lead to the Vizome

Let's take a look at its basic qualities. In their seminal book, *A Thousand Plateaus: Capitalism and Schizophrenia*, Gilles Deleuze and Félix Guattari define the six principles of the rhizomatic structure:

"1 and 2. Principles of connection and heterogeneity: any point of a rhizome can be connected to anything other, and must be" [2]. To be more specific, there is no beginning or end to a rhizome; it can be started or entered at any point. For a virtual-world-based comparison, think of how many access points a virtual world has. It is possible to interact with the inhabitants and objects in a virtual world via your mobile phone, your desktop computer, via text chat, or 3D visuals, and in fact, you can open multiple windows on your computer screen and run multiple avatars that are interacting with others in multiple locations.

"3. Principle of multiplicity: it is only when the multiple is effectively treated as a substantive, 'multiplicity,' that it ceases to have any relation to the One as subject or object, natural or spiritual reality, image and world" [3]. This speaks to the concept that to understand the rhizome you need to understand that it takes its identity from being made from multiple parts, that it is a multiplicity rather than an entity.

"4. Principle of asignifying rupture: against the oversignifying breaks separating structures or cutting across a single structure. A rhizome may be broken, shattered at a given spot, but it will start up again on one of its old lines, or on new lines" [4]. As any gardener knows, plants with a rhizomatic root structure are hard to contain as they seek to spread out through the garden. The very nature of a rhizome protects it from destruction, by providing many sources of connection to its sustenance. The same thing happens in the Metaverse, as data is rerouted around disabled servers or as people use mobile-phone-based social media to communicate in areas whose physical networks are disrupted during disasters.

"5 and 6. Principle of cartography and decalcomania: a rhizome is not amenable to any structural or generative model" [5]. This speaks to the general nature of the rhizome, which cannot be flattened into a 2D outline or diagram. In fact, the book *A Thousand Plateaus* is itself a rhizome, full of complex interconnected ideas that are organized around five types of problems: epistemological, ontological, anthropological, ethical, and political [6]. Throughout the book, Deleuze and Guattari assemble philosophical ideas, scientific theories and psychological concepts, mixing and remixing the interconnected references, like improvisational jazz musicians, to bring the sensation of understanding the rhizome to the reader. *A Thousand Plateaus* can be started from any page, read out of order, and yet will still connect back into its main ideas and theories. Unfortunately, Deleuze and Guattari both died in the 1990s before the birth of the Internet, so we will not know how they would have applied their rhizome concepts to the World Wide Web, much less a virtual world. Dr. Janet Murray (Georgia Tech) mentions the rhizomatic structure as one of the forms of agency, a method of navigation in postmodern hypertext narratives [7]. We have all spent hours online just jumping from one link to another, collecting and connecting information from the vast rhizome of hyperlinks. Those are the waters that you must navigate as a virtual world designer. It is your primary task to help the virtual explorer find visual, aural, and semiotic connections. Let us explore how.

3.2.2 Examining Parts of the Vizome: How Things Are Connected

We are real creatures who define our very existence by relative directions. Gravity pulls us down, our eyes look out of the front of our faces, we hear from the sides of our head. Because we are designed to be directionally oriented beings, our lives and relationships are often defined by a collection of horizontal and vertical descriptions. We use terms such as "upwardly mobile," "upper crust," "lateral promotion," and "underling." Think about what you would consider to be horizontally related connections. These connections share aspects of common locational position, age range, and personal selection such as

- The neighbors on the floor of your building
- People in your virtual world groups or guilds
- Your Facebook, Twitter, and Google+ friends and their posts

Now think about vertical connections you have. These connections do not share common aspects, but there is influence based on positional orientation and relative status in the community or the workplace.

- The neighbors above or below your apartment
- People-based avatars who own the virtual company you work for
- Your social media posts and their popularity or ranking

Of course real life in most parts of the world contains both vertical and horizontal social mobility; there can be generational elevation in familial social status, and individuals can move laterally to start new occupations that hold equal status. In the virtual world, this network of connections has developed under the influence of "social capital" and "virtual families." This creates an interesting perspective for a designer of virtual environments. Ask yourself, How does the population of a virtual environment connect socially through the vizome and how do those kinds of connections influence their social capital and its related content? Some of the answers to this question may be

- They connect through joining virtual groups with common interests, and share content that is valuable to their interests and cause.
- They connect through personal representations such as profiles or blogs, and promote content that conveys status.
- They connect through personal expression and create content that conveys status by being unique or highly popular.

To deepen your understanding of these connections, ask yourself the following questions.

- What are the personal horizontal or vertical connections in your virtual world design?
- What are the social horizontal or vertical connections in your virtual world design?
- How do these influence the visitor's view of the virtual environment and its content?
- What other aspects do these connections add to your visitor's point of view?

These questions will enhance your design capabilities, because they increase your awareness of the vizome, the people who occupy it, and the kinds of social connections they have made in it. Knowing about those structures can open avenues of opportunity for designing content that delivers a message that resonates with the population, content that is useful to them because it provides for their needs.

3.2.3 Plateaus of the Vizome; the Assemblage of Concepts and Connections

"A plateau is always in the middle, not at the beginning or the end. A rhizome is made of plateaus" [8]. That statement is from the introduction to *A Thousand Plateaus*, and it brings some interesting ideas to mind. If the rhizome is made of plateaus, and they are in the middle of all connections, and the meaning of that information, or even the experience is in the mind of the visitor or virtual resident, where can a virtual environment designer find their "visualization point" or a perch from which to start their conceptualization? The secret lies in approaching your design problem with the "and, and & and" to paraphrase Deleuze and Guattari [9]. The plateau of your vizome is the middle, and the milieu, and the assemblage and the multiplicity. To create a design with the "and, and & and" approach, you start by finding an idea and letting it connect to many other ideas. Each connection made on that design plateau will add a level of connection and depth to the overall concept. Some tools/methodologies to consider using for this design approach would be a mind map, or an image collage, or perhaps you just need to make a list of as many keywords as you can think of on a big sheet of paper. After all, the Internet runs on keywords and the search engines that find them, so why not leverage that approach for brainstorming your design?

3.2.4 A Case Study: Using the "And, And & And" Approach in Designing a Virtual Environment

To exemplify, let us examine a case study of how the "and, and & and" approach was used for designing a virtual environment. In 2010, a game-based virtual environment was built in Second Life at the IBM Exhibit Program by the Alchemy Sims build group. The project was called "Inland: Search for the Sy" [10]. Essentially, the design of this game-based virtual environment encompassed three distinct realms that were interconnected through a series of puzzles. These puzzles incorporated elements from a textual source, a diary containing the backstory of three scientists whose avatars have disappeared from a region in Second Life. The first thing a visitor to Inland: Search for the Sy would see was the diary lying on the steps of an abandoned transit terminal. The diary contained a narrative based on the notes of Dr. Aubrey Wynn, her observations about the strange events in the region (see Appendix B), and challenged the visitor to solve the mysteries of the region and help bring the scientists back to their virtual world.

3.2.4.1 Design References and Processes for Inland: Search for the Sy

When the Alchemy Sims build group started to design the content for Inland: Search for the Sy they started with these three major ideas: (1) that without the influence of people-based avatars, the biosphere of a virtual world would devolve and return to a more primitive state (in this case, one that resembled the insect-dominated Carboniferous period of our earth); (2) that silicon-based life forms could exist and occupy the silicon-based components of machines like our computers; and (3) that parallel worlds could be connected through anomalous portals appearing within the virtual worlds. As the design progressed, each one of those big ideas spawned more concepts that were woven into the overall concept and game structure, as well as the creative network of the Alchemy Sims build group. For instance, the silicon-based life forms of our real world, the diatoms in our oceans, were studied for their intricate structures. Those diatomic patterns, along with 3D fractal patterns created by applications like Mandelbulber (http://sourceforge.net/projects/mandelbulber/), were utilized in the design of the silicon-based Sy creatures that were major characters in the game. In the backstory, the Sy connected to us through the virtual world portal, and during the course of the game, each player had to become Sy in order to solve the final puzzle, reach Sy Space, and collect their finishers' trophy. On and on it went, as they designed the game-based environment, each new concept was interconnected to an aspect of the major ideas, and new graphics/textures and visual layers were added. Previously unrelated ideas came together and created another interconnected plateau of concepts contained in the design for this virtual environment.

3.2.4.2 Takeaways from This Case Study

The question to ask yourself when you are looking for a design point of view on any project is, What is my first idea, and what is my second idea, and what is my third idea? Write them down, rearrange them, combine them, or split them into smaller bites. Find out where these ideas intersect and you will find your design plateau. This plateau contains the milieu of design considerations for that virtual environment and virtual content. You will soon discover that each concept is more than part of the network and each connection is also a world unto itself. Eventually more and more plateaus will appear as your design concepts deepen and develop. In Figure 3.1 you will see a diagram depicting the "and, and & and" approach to design on the Inland: Search for the Sy project that was built in 2010, Second Life, IBM Exhibition Region.

3.3 PUSHING THE DESIGN FURTHER BY USING OBSERVATIONS WITHIN THE VIZOME

If design of a virtual environment comes from observations, then you can leverage your observational skills to push the design further. Observing how visitors interact with your design will generate ideas about how to enrich and deepen the experience for your visitors. For example, suppose you observe that avatars consistently gather around the entrance to one of the popular art regions in a virtual world to chat and show off their new outfits. To transmute those observations into design ideas ask these questions:

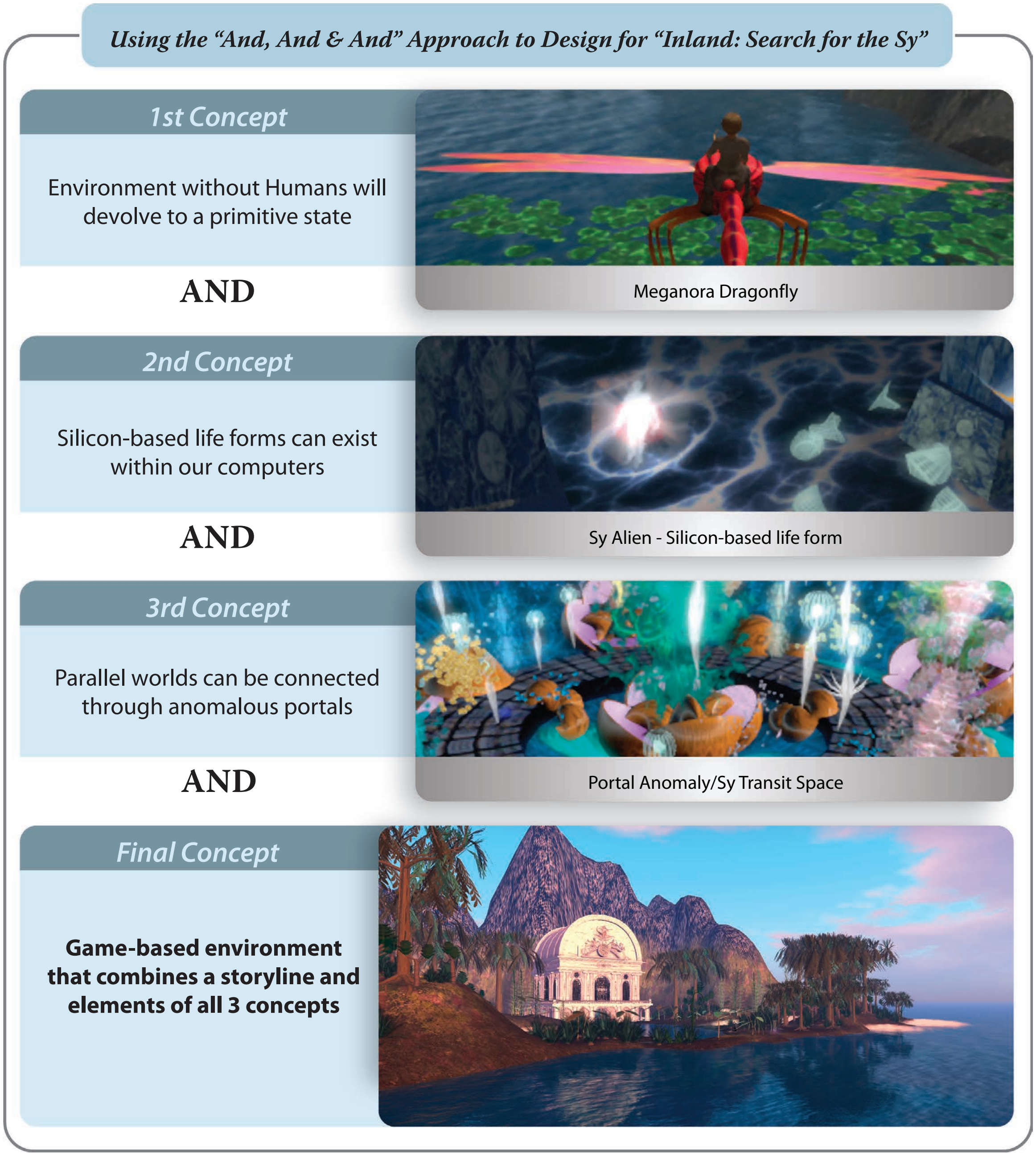
- Why are the avatars gathering in this specific place?
- What are the avatars doing besides chatting and displaying themselves?
- What does the pattern of the gathering look like? Are they in small groups, pairs, or a large circle?
- Does the design of the space affect or promote that gathering pattern?
- Are the avatars interacting with the environment in any way besides occupying it?

Now suppose that from these questions you discovered the following information.

- The avatars like to gather here because it is a consistently popular and busy area, and this is where they like to chat and show off their new outfits while they stand around in a large circle.
- You observe that the large plaza with steps and fountains around them provide a natural amphitheater space for them to gather in, but that they do not interact directly with any of the structures around them.
- However, they are actively interacting with each other via text chat, instant messages, voice chat, and visual display.

You may theorize that this kind of plaza space is a natural draw to avatars who want to socialize because of its configuration. That kind of spatial analysis can present the observant designer with additional questions such as

- Would you conclude that the space lives up to its potential?
- Is the space accessible and safe?



Using the "And, And & And" Approach to Design for "Inland: Search for the Sy"

1st Concept

Environment without Humans will devolve to a primitive state

AND

2nd Concept

Silicon-based life forms can exist within our computers

AND

Sy Alien - Silicon-based life form

3rd Concept

Parallel worlds can be connected through anomalous portals

AND

Portal Anomaly/Sy Transit Space

Final Concept

Game-based environment that combines a storyline and elements of all 3 concepts

FIGURE 3.1 Using the "and, and & and" approach to design for Inland: Search for the Sy.

- If you think it falls short of those goals, what would you do to improve it?
- If the space visually represents one of the plateaus in the vizome, what kinds of connections are being made here? Social chat connections to be sure, but what if someone was streaming their computer screen view of this plaza to an online site like Hitbox (http://www.hitbox.tv/)?
- What are some new ways that this space can become interactive?
- Can a game be woven into the area, responding to text chat or mouse clicks?

These questions can go on and on, and as you deepen your understanding of interconnecting aspects of this virtual environment, your capacity to analyze the virtual environment for its design possibilities will grow. In his book *The Fold: Leibniz and the Baroque*, Gilles Deleuze interprets various aspects of the Baroque period and the writings of Gottfried Wilhelm Leibniz (1646–1716), a noted mathematician and philosopher. Deleuze says, “We witness the prodigious development of a continuity in the arts, in breadth or in extension: an interlocking of frames of which each is exceeded by a matter that moves through it” [11]. This statement could also describe the mutability of art in a virtual world. We witness this as 3D geometry becomes architecture, wearable content, or parts of an interactive vehicle, and in turn is represented in a machinima, a website, or blog post. To enhance your understanding and enhance your perception of this continuum of art and communication, activate tools that let you observe parts of the vizome as it functions: the nearby radars and location maps, text chat, voice chat, object chat, statistics bars, and other tools that you can find on the top bar of the viewer menu under the Advanced and Develop tabs. (Note: If these tabs are not turned on, you can activate them in the Avatar/Preferences/Advanced menu on the top bar of the Firestorm viewer menu.) Try toggling the various settings under the Advanced menu: Performance tools, Highlighting and Visibility, Rendering Types, and Rendering Features tabs. Then try toggling the settings under the Developer menu: Render Tests, Rendering Metadata, and Rendering tabs. As these various render settings affect what becomes visible in the viewer, they reveal the metadata it is contributing to the scene. Ask yourself these questions:

- What are the ways in which this metadata connects to the visitor, to the simulator, and to the overall environment?
- How does the design of various objects in the virtual environment affect that connection and meaning?

3.3.1 The Vizome and Designing for Temporal Location

As people (and machines) interact more via virtual environments, the primacy of our physical location decreases while the importance of our temporal proximity increases. Typically one of the first conversational exchanges in a new online friendship is about relative temporal proximity, the standard “Where in the world are you?” question. The time difference between the two new friends and their individual daily schedules will often define the nature and development of the relationship.

3.3.1.1 Conceptual Elements for Temporal Location Design

Designing for when is an interesting problem. Let us take a look at a few important concepts to keep in mind: temporal reference, connection overlap, and shared experience.

Temporal reference—There is a need for a temporal gauge in any environment. Just as media newsrooms the world over have a bank of clocks on the wall showing the current time in major cities, you will want to make sure that visitors to your space have a sense of their relative temporal location. Visually this may be accomplished with elements such as the environment settings (called WindLight in Second Life, LightShare in OpenSim), scripted moving shadow elements, digital readouts incorporated into a moving wall texture, and other similar systems. The possibilities are endless; just research the history of timekeeping and its artifacts for inspiration.

Connection overlap—If you are designing a virtual space that will be used for creative collaboration, then you must support the real-time communication connections for people in different time zones. Visual accessibility in the space is useful for communication, so perhaps they share a double-sided desk or a large work platform. Having a virtual space designed so that members of the same workgroup can see who is currently in the space working, without directly invading their visual privacy, provides for good communication flow. You should provide spaces in your environment that allow for the virtual equivalent of the watercooler chat or the elevator chat.

Shared experience—With screen capture, first-person virtual world experiences can be preserved, replayed, and viewed by others. Personal experience can echo across your connections and strengthen the perception of a shared experience within the virtual environment. Ask yourself these questions.

- How can you design areas in the virtual environment that encourage the making and sharing of recorded experiences across the vizome?
- Would you actually provide a virtual TV studio or viewing room?
- Would you add several viewing screens to the space, on the walls, or perhaps the surface of the desk?

Designing a location that supports the need to reach other temporal locations requires that you pay careful attention to the way groups communicate. Perhaps a method for storing messages and their playback is required or perhaps a device that records changes to 3D content for future playback to collaborating members of a build group. Jon Brouchoud and Ryan Schultz from Studio Wikitecture had such a device with the Wiki-tree in Second Life, circa 2008 [12]. This device allowed for collaborative creation on the design of architectural models, and recorded the various modifications made by the group members for playback on demand.

3.3.2 The Vizome and Designing for Social Interactivity

Imagine if Rapunzel had access to social media. She might not even hear the prince calling to her from the base of the tower to let down her hair. She might not even care; perhaps her prince was somewhere else online. Fantasy characters aside, human beings are social animals, and through the global access afforded by virtual worlds we have found new ways to form social groups and online societies. Compare the layouts of a Facebook page, the Firestorm viewer, and your LinkedIn profile. Observe what these social interfaces have in common. You will probably notice that they share the following components.

- A place to establish your identity and customize your profile
- A list of the groups you have chosen to belong to
- An area where you can view the text chat and links your connections are sharing with you and other members of the group

These are three basic components of a social media interface. Now, let us measure these components against the standard for good design described by Don Norman [13]. He tells us that good design makes us happy and that this happiness is created by three factors: (1) beauty, (2) functionality, and (3) reflection. Essentially, he is telling us that we are happy when we can use things that are designed to be beautiful (think of your favorite sleek sports car), when they are really easy to use (think of your favorite tool or software application), and when they allow us to reflect or be reflected on in a positive light (think of how you feel when you use an energy saving or green device). Using the format in Table 3.1, let us compare the components from a Facebook page, the Firestorm viewer, and your LinkedIn profile to those good design standards. In Table 3.1, you can comparatively score these sites on a "happiness" scale. Did anything score in the 80 to 90 range?

Now try scoring your favorite hair, shoes, or heads-up display (HUD)-based content from a virtual world using Table 3.2. Some of the left-hand boxes have been left blank for you to enter your own personal content. (Note: Tables 3.1 and 3.2 will be available for download as MSWord documents. The location will be http://www.anncudworthprojects.com/, in the Extending Virtual Worlds Design Book Download section, organized by book chapter.)

Now, activating your design detective instincts, ask yourself these questions.

- Does any of the content you just scored overlap into the social criteria?
- Does that new hairstyle provide you with the same social esteem that a great profile picture does?
- Did you share information about its purchase to other members of one of your groups?

Here is the vizome again, overlapping and interconnecting our content with social media in a virtual world.

TABLE 3.1
Scorecard for Good Design That Makes You Happy on a Social Interface

	Rate on Scale of 1(Bad) to 10 (Good)		
User Interface to Be Analyzed	**Beautiful Does the Design Please the Eye and Feel Good to Use?**	**Functional Is the Design Understandable and Comfortable to Use?**	**Reflective How Does Using This Area Feel? Does It Raise Your Self-Esteem?**
The place where you establish your identity and customize your profile	(Put your score here) ____________	____________	____________
The areas where there is a list of the groups you have chosen to belong to	____________	____________	____________
The area where you can view the text chat and links your connections are sharing with you and other members of the group	____________	____________	____________
Overall score: Total down and then across to far right for overall score (highest possible score is 90)	____________	____________	____________

Source: Created from information from Don Norman, "The Three Ways That Good Design Makes You Happy," YouTube video, 12:41, posted by TED Talks, March 9, 2009, accessed June 15, 2014, http://youtube/RlQEoJaLQRA.

TABLE 3.2
Scorecard for Good Design on Content

	Rate on Scale of 1(Bad) to 10 (Good)		
Content to Be Analyzed	**Beautiful Does the Design Please the Eye and Feel Good to Use?**	**Functional Is the Design Understandable and Comfortable to Use?**	**Reflective How Does Using This Area Feel? Does It Raise Your Self-Esteem?**
"Fat pack" of hairstyle from my favorite designer	(Put your score here) ____________	____________	____________
Aircraft from my favorite content builder	____________	____________	____________
Photo studio HUD for taking pictures inworld	____________	____________	____________
Adjustable fitted mesh dress	____________	____________	____________
	____________	____________	____________
	____________	____________	____________
	____________	____________	____________
	____________	____________	____________
	____________	____________	____________
Overall score: Total down and then across to far right for overall score (highest possible score is 90)	____________	____________	____________

Source: Created from information from Don Norman, "The Three Ways That Good Design Makes You Happy," YouTube video, 12:41, posted by TED Talks, March 9, 2009, accessed June 15, 2014, http://youtube/RlQEoJaLQRA.

3.3.2.1 When Social Media and Virtual Content Go Wrong: Two Cautionary Anecdotes

3.3.2.1.1 Anecdote 1: The Noob and the Noisy Bed

Sometimes, the connection of virtual content with social media has unanticipated and amusing results. A friend of mine told me a story about her early days in Second Life. As a "noob" (or new user) to the world, she had no idea about the functionality of scripted objects or their limits. Hoping to please her boyfriend and join in with the art-loving groups, she rented a private room at a popular arts venue and installed a fully functional, scripted sex bed. Aside from a huge choice in sexual positions, the bed also provided its user/owner with a full range of erotic sounds. Not aware of this feature, my friend left to buy some new shoes in another region. These were expensive and highly scripted shoes, which animated the walk of the wearer. When she got a message from the owner of the arts venue that her bed was audio spamming the other members of the arts venue with unwanted erotic noises, she tried to teleport over to remove the offending object. Unfortunately, the scripting in her new shoes was so demanding on the server that her return teleportation was prevented and she could not return to remove the offending bed. For three days she tried to return, not realizing that her shoes were holding her fast in place, while the pleas from the arts group manager changed to panicked demands. Eventually, my friend figured out what the problem was, jettisoned the shoes, and popped over to remove the noisy bed from her rented room. She could not return to the "scene of the crime" for several months, as the mere thought of it made her cringe in embarrassment. This is a memorable example of private interactivity being pushed into the social realm, and it serves to remind us that people-based avatars have a sense of personal boundaries even in a virtual world. Some new users may not realize that their content could violate those boundaries, create a social faux pas, and embarrass them. Every designer and content creator should be very aware of this and tell their customers about it. A default silent setting and a clear set of instructions on how to quiet that noisy sex bed would have prevented my friend's embarrassing situation.

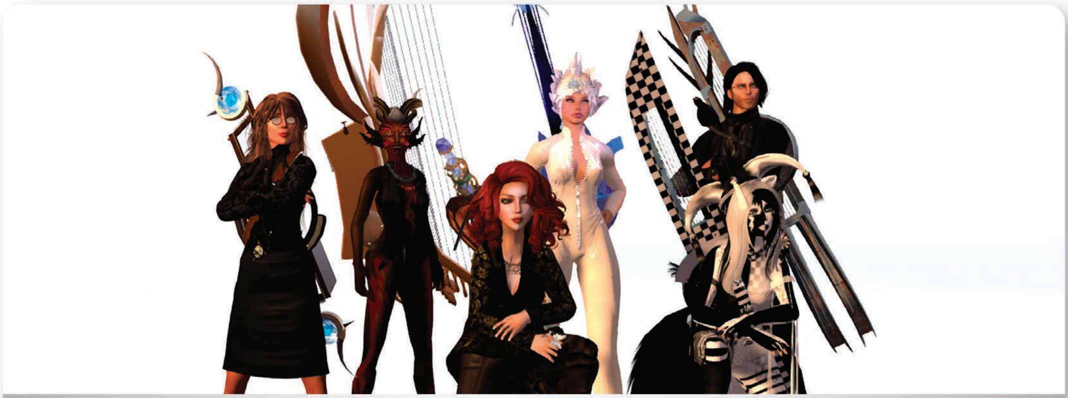
3.3.2.1.2 Anecdote 2: Kill the Sion Chicken!

Another interesting example about the negative impact of socially based content concerns breedable virtual animals. In 2009, Sion Zaius introduced the Sion Chicken to Second Life. The popularity of this virtual pet skyrocketed as people bought his content so they could set up their own chicken breeding businesses. The chicken populations increased drastically and soon their physics scripts overwhelmed the servers supporting the regions of Second Life, creating a huge lag effect that slowed avatar movement and physics response times to a crawl. This detrimental effect continued even after the "lag-less" chicken was introduced [14]. The whole franchise arrangement that Sion Zaius set up led to a fowl population explosion, and then a backlash that included organized chicken hunting and biowarfare. As the Second Life population split into pro- and anti-chicken factions, the social atmosphere of that virtual world got more fractious. That friction ignited new ideas about the social uses of a virtual world. New social modes were introduced in the virtual environment as people brought with them behaviors from their own gaming and social cultures. In his 2009 Ignite NYC presentation, Patrick Davison asked, "How do I perceive the function of the platforms I use online and to what degree does this perception of a function lead to the conflicts that arise there?" [15]. He noted that some people behaved like Second Life was a chat room with 3D avatars, and others treated it like a MMORPG (massively multiplayer online role-playing game) with no rules or rewards or goals. He also observed that some groups use virtual environments for social experimentation and will exhibit behavior not usually accepted in polite society. The disruptive aspects of the Sion Chicken content on the physics engine and overall functioning of virtual environments is one example of how these differences in perception and social use of virtual worlds can be escalated into large online conflicts when the design of someone's content disrupts other people's personal standards of acceptable social behavior. Eventually, new rules and land covenants were created, the novelty of breeding chickens faded, and eventually the social outcry settled down. Ultimately, the Sion Chicken story is a cautionary tale about how much impact a radically different kind of content can have on the daily life in a virtual world and how difficult it may be for it to finally establish its niche in the virtual ecosystem. These days there are many forms of breedable creatures for purchase in Second Life and the OpenSim Hypergrid.

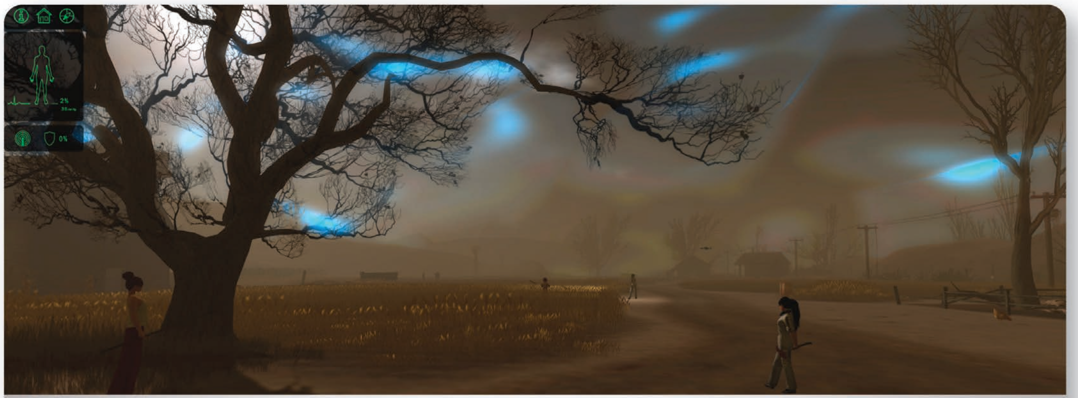
3.3.3 Social Content with Strong and Weak Ties: Dunbar's Number

One more important factor to keep in mind when you are designing content with enhanced social interactivity is the persistence of *Dunbar's number* [16]. In 1998, Robin Dunbar concluded a study of social groups in animals and formulated the concept called "Dunbar's number for the species." He stated that the maximum number of individuals in a connected and stable social group could be averaged at 150 (statistically, the number ranges from 100 to 230, at 95% confidence). Dunbar's later studies of groups in historical militaries and ancient cultures supported this theory. The connections between the 150 people are considered strong and will include frequent updates within a close proximity, but if a group exceeds 150 in number, social ties diminish and communication weakens. To maintain strong connections, the number has to remain around 150. However, anyone who has a social media site these days is aware that the number of "friends" or "connections" that they have often greatly exceeds 150. While there may be strong connections within someone's social media friends list, there are also weak connections, reflecting the widespread and shifting nature of social communication in our modern world. It stands to reason that when you are designing content with a

Social Content Used in a Virtual World



Social content created by members of the Metaharpers Cabal. The members each create a personal Metaharp as an identifying symbol of their individuality and membership in the group. Pictured left to right, Vicki Brandenburg, Darlingmonster Ember, Gloriana Maertens, Sho Kyong, Tozh Taurog, and Arrehn Oberlander.



Social content made for "UNIA" (uniathegame.com), a game developed for Second Life by MadPea Productions 2013. HUD creation/scripting by Arduenn Schwartzman. Special thanks to Kiana Writer and Harter Fall.

FIGURE 3.2 Examples of social content used to connect a social group. Shown are the Metaharpers' personal harps and the MadPea Productions game pieces.

social aspect to it, you should consider the kind of social network ties that would be involved with the consumption and use of that content. Think about these questions.

- Is the content you are creating for distribution among those who are closely and strongly connected, or is it for the wider and more weakly connected members of the social group?
- How can this be differentiated?

An interesting example of content that connects a group with strong ties is found within the Metaharpers' group (http://www.metaharpers.net/) in Second Life. Many members of that group have created their own abstract "harp" that they can wear or display as a personal icon. On the other hand, the MadPea Productions group (http://madpea.blogspot.com/) makes content that is given away as components in its Second Life hunt games. This content serves to identify you as a player in the game and may be your only tie to the group. In Figure 3.2 are some screen grabs to illustrate examples of strong and weak socially connected content.

3.3.4 The Arrival of Social Reality on Virtual Reality Social Sites

New to the virtual social space are self-styled virtual reality (VR) social sites, sometimes called social reality. Essentially, these are social media sites that provide social spaces to be accessed with head-mounted displays being developed by Oculus Rift, Sony, HTC Vive, and others [17]. Two example sites are Altspace VR (http://altvr.com/) and Converge (http://www.convrge.co/). It remains to be seen how popular these prove to be, if people will wear VR head-mounted displays to interact with social media, chat with their friends, and comment on posts from their connections. Another interesting aspect to contemplate is how these sites might influence the interactions of people with each other and a spatial interface.

3.4 DESIGNING WITH SEMIOTICS, COLLECTIVE INTELLIGENCE, AND PERSPECTIVISM

Semiotics is the study of signs and symbols. The text-based signage that directs you to a location in a building or the red octagonal shape that means stop at an intersection are important signs and symbols to be sure, but they are just the top layer of semiotic meaning in an environment. Think like a wilderness tracker or crime scene investigator for a minute, and look around your immediate environment.

- What do you see lying about on the tabletop or floor that gives you an understanding of the activities that are happening or have happened in the space?
- What are the signs of life and human presence in your space?

As a virtual environment designer, you use signs and symbols of all sorts to create *purpose*, *meaning*, and *mood* in your spaces. Purpose is created with signage that instructs and guides the visitor in your environment. Meaning is given through the use of symbols and signs that are integrated into the design and décor of the space, such as a cross may be used in a church or barber pole outside a barbershop. Mood is more ephemeral. The specific signs and symbols you can add to create a given mood are highly subjective. Adding a picture of a dead ancestor to the wall of your virtual environment may not create the mood of deep sadness you hope to elicit in the visitor. However, combine that image with some dead flowers, a half-burned candle, and some canned beans on a plate, and you begin to create a picture in the visitor's mind of a grieving person just hanging onto life by the barest of threads as they mourn the loss of a loved one.

3.4.1 Collective Intelligence and the Four Spaces

In his fascinating book *Collective Intelligence*, Pierre Lévy defines four anthropological spaces: *earth space*, *territorial space*, *commodity space*, and *knowledge space* [18]. As you read the following quotes from *Collective Intelligence* about these spaces and how human civilization interacts with them, think of the parallel manifestations they have in a virtual world.

- Earth space is described as "the immemorial space-time to which we cannot assign an origin. It is the space that is 'always already there,' that contains and overflows the beginning, unfolding and future of the human world. The earth is not a planet, not even a bio-human world, but a cosmos in which humanity communicates with animals, plants, landscapes, locales and spirits. The earth is the space in which mankind, the stones, vegetables, beasts and the gods meet, talk, come together and separate in a process of unending re-creation" [19].
- Territorial space is composed of the "domestication and rearing of animals, agriculture, the city and state, writing, the strict social division of labor—the order of appearance of such innovations has differed from place to place. But whenever they connect and mutually reinforce one another, they acquire an irreversible force, a power of expansion, permanence such that a new reality is established, the sedentary world of civilization" [20].
- Commodity space, the offspring of capitalism, is described as "the great cybernetic machine of capital, with its extraordinary powers of contraction and expansion, its flexibility, its ability to insinuate itself everywhere, to constantly reproduce the relations of commerce, its epidemic virulence seems invincible, inexhaustible. Capitalism is irreversible. It is economy and has made economy the permanent dimension of human existence. There will always be a commodity space, as there will always be an earth and territorial space" [21].
- Knowledge space is described in this way by Lévy: "Etymologically, it is a u-topia, no-place. It is embodied nowhere. But if it is not realized, it is already virtual, waiting to be born. Or rather, it is already present, but buried, dispersed, travestied, intermingled, sprouting rhizomes here and there." And he continues, "Knowledge, in the sense I am using the term, is a knowledge-of-living, a living-in-knowledge, one that is coextensive with life. It is part of a cosmopolitan and borderless space of relations and quantities, a space for the metamorphosis of relationships and the emergence of ways of being, a space in which the processes of individual and collective subjectivization come together."

These four spaces provide an interlocking conceptual framework of the real world and could very well serve to provide a template for virtual world design as well. Perhaps a way to start is to ask these questions.

- Can the earth space concept be interpreted through a virtual terrain that allows the residents to communicate directly with the virtual flora and fauna?
- Does the virtual world design encompass the territorial space by allowing virtual civilizations to arise through territorial division and developing a virtual culture?
- Will the commodity space be supported by the content and means of exchanging content, goods, and services as commodities in your virtual environment?
- Can the knowledge space be experienced in the virtual environment through interaction with the simulation supporting it, and does it promote a state of being that supersedes the immediately visible format of the avatar, allowing for multiple layers of information and contact with other avatars in the space?

All difficult questions to be sure, and the answers may come together piece by piece over a long period of design and experimentation. What is important is that you, the designer of a virtual world, realize the plasticity of a simulation and that you can change perceptions in the minds of the visitors with your design choices.

3.4.2 Perspectivism: Finding the Point of View for Your Design

Friedrich Nietzsche, the great German philosopher, created the term *perspectivism*. Essentially it means that there are many possible conceptualizations of an idea, value, or "truth" depending on the visitor's perspective. This does not imply that all observations are equally valid, but only that there is more than one approach to conceptualization. Designers of virtual environments understand and practice a perspectivist attitude by maintaining an open mind, allowing them to tap into a deep and wide flow of ideas that will enrich and enliven their design work. By opening your mind and embracing this spectrum of perspective, you will strengthen the foundations of your design point of view, picking and choosing what elements best support the design for your virtual environment. Let us look at an example of the process. Suppose you want to create a virtual environment that supports emergent game play, and evokes a romantic image of the earth's historical past as the Pre-Raphaelite painters did in the mid-19th century. The research you do into the art and architecture of that period in England will reveal a strong vein of related content. However, rather than copy what you see in the pictures into the content of your virtual environment, try to find the point of view you have on the images you see. The Pre-Raphaelites and the images they made can be better understood if you imagine viewing them in the yellow-green gas lighting of their age. They founded a new movement in art by looking backward to the paintings made before Raphael in the "Quattrocento," or the 15th century Italian Renaissance, in an effort to defy the artistic conventions and academic standards of their day and age.

Now ask yourself these questions.

- How does that knowledge relate to what you are trying to design?
- Are you redefining an artistic format in your virtual environmental design?
- What elements of your daily life are influencing your understanding and opinion of this content, that is, what do you think about these Pre-Raphaelites?

You may be surprised to discover that in answering these questions, your appreciation of the research has increased, and perhaps as you began to dig deeper, you will absorb the historical context of the mid-19th century that produced these painters. The acceptance and rejection of art styles and manners has continually cycled through our histories, impacted by cultural tastes, political forces, and technological advances. Imagine what the paintings might look like if they were called the Pre-Picassos or the Pre-Hirsch Brotherhood, or imagine how their work may have looked if the Pre-Raphaelites had formed in 1965 and included Peter Max in their group. Any of these perspective-shifting exercises can lead you to a unique design point of view; you need only to dig in and find the one that anchors your project's visual/textual/audible style most effectively to your desired overall point of view. In Figure 3.3 there is a diagram that illustrates a microcosmic view of how just one semiotic element can reside in the four spaces of collective intelligence, and how the meaning of that element can be tinged by perspective.

3.5 SETTING UP A CREATION SYSTEM TO DEVELOP IDEAS LIKE A VIZOME

Preparing for and utilizing creative thinking is a critical skill for the designer. In the next sections we will look at various ways that you can achieve a new level of creative thinking and how to make use of some creative tools, including your subconscious mind.

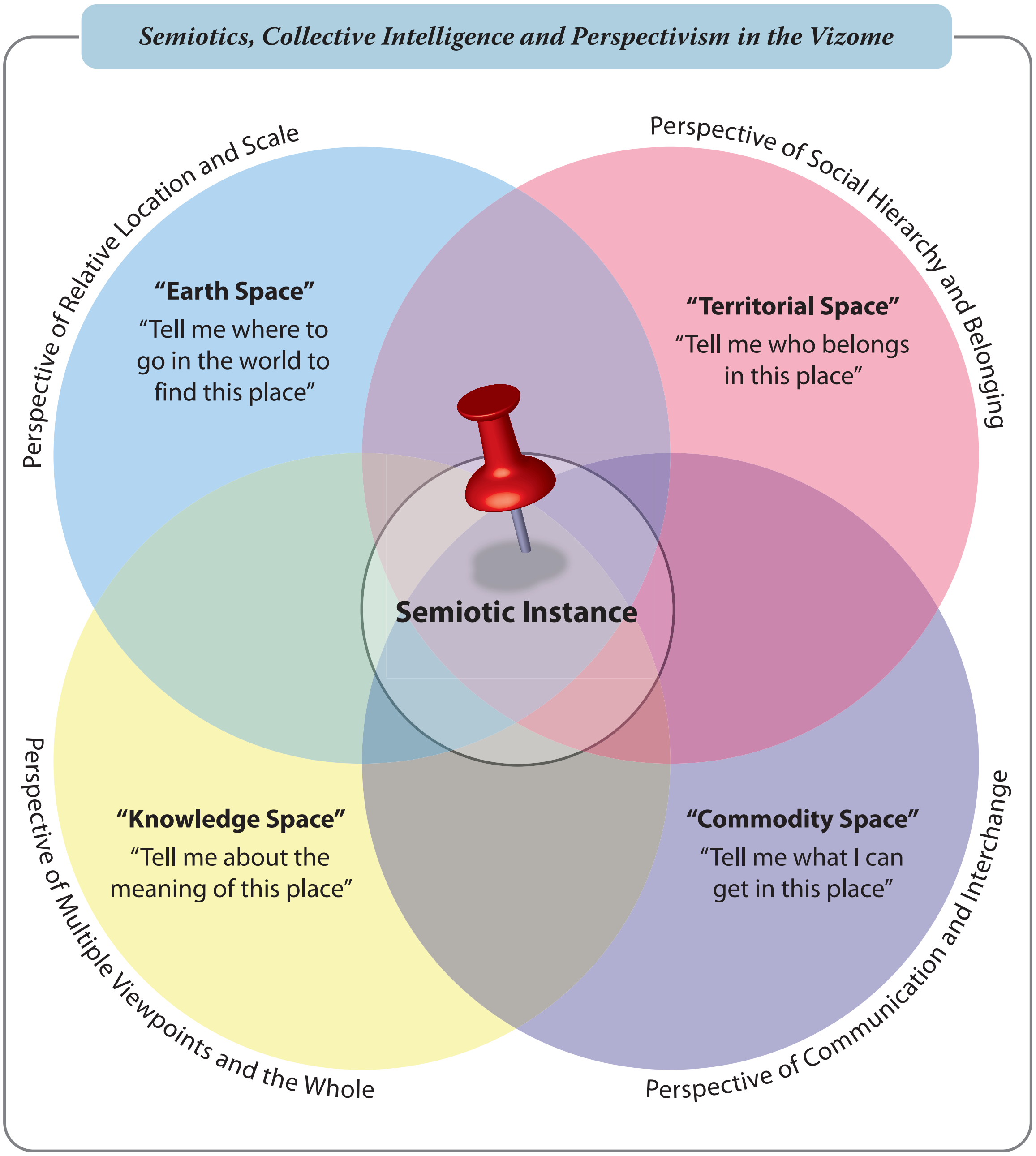

Semiotics, Collective Intelligence and Perspectivism in the Vizome
Perspective of Relative Location and Scale
"Earth Space"
"Tell me where to go in the world to find this place"
Perspective of Social Hierarchy and Belonging
"Territorial Space"
"Tell me who belongs in this place"
Semiotic Instance
Perspective of Multiple Viewpoints and the Whole
"Knowledge Space"
"Tell me about the meaning of this place"
Perspective of Communication and Interchange
"Commodity Space"
"Tell me what I can get in this place"

FIGURE 3.3 Observing semiotics, the four spaces of collective intelligence and perspectivism.

3.5.1 The Vizome and a Creative Frame of Mind

Now that you have seen how the vizome provides for a vast system of connections between the individual avatar and the virtual world, new creative possibilities can emerge. Within the artistic collective of an international virtual world, the vizome adds and creates new opportunities for creativity. In fact, the more you know about the culture of the people you are working with, the deeper you can dive into creative connections, and the more you can avoid misunderstandings about symbols and meaning. As you start to work with an international creative team, the schedule, project timing, cultural humor, and cultural play styles all become important sources of creative support and inspiration. If you are working as the creative director on the project, you should ask these questions.

- What is the virtual space available to my team, and how can we make it stimulate our creative flow?
- Have I given the team time to play and the time to develop their ideas comfortably?
- Have I supported them enough so that they have the confidence to defend their ideas?

As the creative director or lead designer, you are in charge of supplying environmental tone or overall creative mindset. John Cleese, of *Monty Python* fame, describes two mindsets in the creative process in his online lecture about creativity [22]. In it he says, "Creativity is not a talent, it is a way of operating." He goes on to describe that there are two states of mind involved with the creative process and how we function. They are open and closed, and "creativity is not possible in the closed mode." John Cleese describes how Alfred Hitchcock would keep his screenwriting team in a creative mode by telling unrelated stories to the group when he felt the pressure had gotten too high. Hitchcock knew that the group had to stay in an open frame of mind and that he needed to reduce the anxiety, and by providing unexpected juxtapositions and creating a playful attitude, he kept them in a creative flow. Humor is a great pressure reliever. Try telling a joke or funny anecdote the next time you or your team gets stuck on a creative problem. Train your team and yourself to tune into the design voice inside and to tell a funny story or think of something relaxing when that voice is blocked. Pay attention to what you see, hear, feel, and experience every day, because there is inspiration everywhere if you look for it. Be aware that there are gravitational attractions and repulsions in design creativity, sometimes looking at or thinking about something repellant can help your creative mind strengthen. Ask yourself, should your design work for or against the emotional tone to tell a more complete story? What may look like a mistake when you are in a closed frame of mind may be a creative opportunity when you are in an open frame of mind.

3.5.2 Getting the Creative Tools Organized

Now as you are entering the open creative frame of mind, it is time to get your tools together for the creative process. Tools have three categories.

1. Physical/touchable
2. Psychological/cognitive
3. Digital/computer based

Let us review them and their uses in the creative process.

3.5.2.1 Physical/Touchable Tools

These are the tools you can hold in your hand or experience physically with your body. First and foremost, you need to record your creative ideas. Always have a means to do that at hand. Personally, I like to keep lots of little notebooks around—on my desk, in my backpack, on my nightstand by the bed. I even have a "pilot's" pen that will light up on the writing end, so I can make notes in the dark without turning on the room lights. In a pinch, when I find myself with an idea and no notebook handy, I take out my cell phone and make a voice recording of the idea for playback later.

Another, very important physical tool is the space where you do creative thinking. Obviously you want some sort of sanctuary, which helps you relax, frees your mind from the day-to-day thoughts, and selectively closes off the distractions of life. Unfortunately, for many of us, the qualities of this space are not exemplified in our daily workplaces. If you want to boost your creativity in your workplace try to include the aspects of a relaxing sanctuary for yourself, and utilize professional methods for controlling and managing the distractions of continual e-mails and phone calls.

3.5.2.2 Psychological/Cognitive Tools

The hippocampus and frontal cortex of your brain, as well as your short- and long-term memory, are key tools in creative thinking. Here is our personal rhizome, our interconnected, collective experience of recalled images and sensations. The more you use your brain for creative thought, the better it gets at entering that state of playful cognition. Sometimes I think of it like this: I imagine a huge cloud of memory images spinning around in space over my head. I do not try to control their movement; I just let them float by. Now and then, when I see an interesting one, I add a keyword to describe it. For instance, when I remember a street scene from my college years, a young man standing in the doorway of his building waiting for the rain to stop, I say doorway to myself, and that becomes the tag on that image. I have linked the sound and look of the word *doorway* to that image. Later on, when I look at other doorways on another street or do research on doors, I have a link back to that image in my mind, and the verbal connection triggers the related image. With word-tagged memory images, you can build a personal image library in your mind, deepen your visual palette, and design image vocabulary. Utilize the visuospatial sketchpad or "inner eye" of your brain. There you have stored images from your life experiences to be recalled for your design usage, and the more you use it, the stronger and more flexible it gets. As you prepare for a new project, you should feed your memory as many new images as you can about the objects, colors, and styles that inspire you. A Google search for images is a good place to start, but you should also take the time to visit museums, historic locations, and actual locales that relate to your project. The observations of your senses—the smells, sounds, temperature, lighting, tactile sensations, air pressure, humidity, and emotional feel of the space—will all add to the creative process of designing a virtual environment.

3.5.2.3 Digital/Computer-Based Tools

Digital and computer-based tools can be a great help in the creative process of design. Once a month or so, run a search in your favorite application store for tools that list "creativity" and "creative" as search words. In the list you will probably find desktop and mobile/handheld device applications (apps) for photo editing and manipulation, color planning apps for pallet design and color matching, 3D apps for making 3D objects from images, measuring apps, and "universal design" apps like colorblindness filters. You will also find apps for sketching and painting on your tablets, and social media like Instagram (http://instagram.com/) and Pinterest (http://www.pinterest.com/) that can be used to feed your visual memory banks.

Another interesting category of apps are the mind map applications. Most of these allow you to graphically represent the mental connections you are making while planning a project and could prove useful when brainstorming about a new game-based environment. There is a list of them in the Links section at the end of this book. Finally, you should consider apps that randomly generate word connections for you. These can prove useful when you need to make up names, or identify the qualities of an object, person, or entity. Some good lists of these random generators can be found at the Seventh Sanctum (http://www.seventhsanctum.com/index.php) and at RanGen (http://www.rangen.co.uk/).

3.5.3 Using Your Subconscious as a Creative Partner

The intuitive mind is a sacred gift, the rational mind is a faithful servant. We have created a society that honors the servant and has forgotten the gift.

Albert Einstein

Throughout my artistic career, I have been interested in using my subconscious mind and the images it brings me during dreams as a source for creative ideas. You can think of this part of your mind as your own personal Library of Congress; it holds vast amounts of information from everything you have experienced. The subconscious mind has an attentive staff librarian residing in there, ready to help you find the information you need if you ask for it correctly. Your subconscious takes requests quite literally, so be careful what you ask for if you are seeking specific images or experiences. If you ask it to send you a "sign" it may send you an image of a billboard, instead of some mythical insight or philosophical guidance. Over the years, I have collected various techniques that are useful for connecting with your subconscious mind and its related components, intuition, and insight [23]. Here they are for your collective usage.

- Acknowledge your subconscious; it is a part of your visuospatial sketchpad and your creative partner.
- Nurture it with positive thinking about yourself and the project at hand.
- Practice good sleeping habits, so your subconscious has time to work.
- Connect with it personally as your creative partner.
- Silently speak to it when you want to engage it on a problem or project. You can even make your request in letter form.
- Keep your request simple and direct; the subconscious will take your words literally.
- Record what you dream; keep a journal nearby to write down the descriptions.

3.6 SPOTLIGHT ON DEBORAH THOMAS: SILLYMONKEY GAMES, TRAINING, AND SOCIAL MEDIA

Deborah Thomas understands social media and game-based learning. She acquired the foundation of her extensive media and communication expertise during her early career as a reporter for the *St. Petersburg Times* (Florida). In the early 1990s she applied those skills to training management and instructional design, working for companies like Coca-Cola. By 2006 she had founded her instructional game company, SillyMonkey LLC (http://www.sillymonkeyinternational.com/) to provide serious game design for corporate training. SillyMonkey has provided consulting and game design to many Fortune 500 companies, including Coca-Cola, Allstate, Kraft, CIBA Vision, and IBM. Furthermore, as a game developer, she shares her knowledge of game making through online courses developed with Articulate Storyline (https://www.articulate.com/) and Adobe Captivate (http://www.adobe.com/products/captivate.html).

She has teamed up with Virtway World in Spain to develop Linguo Land, a game-based virtual environment that teaches new languages to students from around the world. Additionally, she collaborated with

Virtway to create SillyMonkey Thinking Space, a virtual world that supports virtual social interaction between learners. Let us ask Thomas a few questions about using virtual environments and social media in serious instructional games.

Question: What social media format is the most applicable in serious games and how would this help enhance the learning? For instance, if you were going to make a serious game to teach people about ethics in the business place, would it be useful to have a "badges" system like Foursquare or a company chatroom like Facebook? Would this corporate community involvement promote or inhibit learning?

Deborah Thomas: I believe that social media can augment the learning event. I would like to answer this with an example. If I were going to make a serious game to teach people ethics in the business place, the badges could be useful, but the real learning takes place if the user learns the nuances of ethics. It's important for the game designer to discover the slippery slope that occurs when people are in a pivotal ethics decision-making moment. What is it that causes a real person to slide down a particular ethics path? What makes a person take the first risk? Then it's up to the instructional technologist to create a scenario that closely replicates that problem through an implementation sourced from concrete possibilities and place the player's avatar in the middle of this scenario. For instance, suppose their avatar filled out an expense report improperly and inadvertently added some expenses twice, or added an extra 0, or added a receipt that wasn't part of the business portion of the trip. Then the new person in Human Resources approved it by accident. When the travel expenses reimbursement check arrives, it is for twice the amount. Do they cash the check? Do they wait to see if any one notices the mistake before you spend the extra money from your account? Do they report the mistake? Obviously, there are many options for creating even more complex scenarios. All of these options are supported by other independent avatar actions, and must be resolved by the player's avatar in the game. Now, if you want the game to contain social media events, you have some more things to consider. I would choose the social media format based on the interactions that I want to entice. Maybe the format should include some polling options to allow people to anonymously answer the questions and then have the group discuss the results. Of course, this media arena involves programmatic solutions that the knowledgeable game designer must confront. Now in the game, learning is enhanced by making the simulated decision and then discussing the action. Badges could be included to provide incentive for good/proper answers. However, the badge award system must be carefully designed to prevent people from trying to "game" the system. If a player simply selects good answers to collect badges, they are not learning. The badge award system could require participants to explain the reason for the "proper" answer and then allow others to vote on "good" answers. Again, all of this must be carefully weighted, categorized, and tabulated. The main point is that the most important part of social learning is to provide moments when the players can think about their behavior. It is the player's behavior that learning professionals want to self-motivate towards positive change. The key to successful social media gamification is three-fold: (1) social media managers must understand what action they are trying to drive, (2) they need to know their target audience (and what motivates them), and (3) they must track everything users do with the kind of metrics which produces understanding. Social media influencers, such as popular blogs, or people with large amounts of followers, who have credibility, bandwidth, relevance, good timing, and confidence can provide huge value to a brand or company. Here are some clues about the kinds of metrics that work. Connections (as in fans or followers) are good, but interactions create more value, and loyal social media influencers can encourage further interaction among people potentially interested in a product or brand. Social media creates connections that are material (with machines, software, and people) and mental (connections with information). Gamification creates a moment of interaction: this is where the combinatorial game mechanic of material and mental connections starts. It is critical to provide the interaction at the intersection of learning objective and game mechanic. Serious games can drive actions and interaction through gamification, but beware; the overjustification effect says rewarding people with extrinsic rewards will decrease motivation. It is more important to give the player a reason to want to make the interaction than it is to earn the badge.

Question: In a virtual world, we can be anyone we desire, even set our avatars to automatically respond with an artificial intelligence (AI) program like the ALICE chatbot. Will AI become part of our social media identity and will our use of it be considered a valuable skill in the future?

Thomas: I do believe that AI will become an integral part of our social media virtual world learning identity because it is on the way and we have to deal with it, just like we dealt with electronic information. Working with an AI will become a very valuable skill in the future. It is constructive and useful, distinctly, to develop avatars that model desired behavior with their own AI, and it is essential to practice sequencing and decision making in critical learning events. Having avatars in a training game will force us to make specific game modeling considerations if we want the results in a rich data catch. There are some things that are best practiced within a simulation, something like practicing single-engine flying. Practicing is vital to learning lifesaving safety maneuvers since there are many steps to learn and the correct sequence is essential to safety training. The first responder in safety training must learn that it is critical to assess the situation before barging in to assist a burn victim. For instance, if a first responder runs in to help before noticing that there is a live electrical wire that is still touching the victim, they could also receive an electrical burn or worse. Furthermore, they need to know that when treating the victim, there are also certain steps and procedures that must be followed. This is a perfect event to practice with an avatar since redundancy takes place without igniting real danger. With an avatar they can easily practice the procedure enough times so that the sequence and process is memorized. Then when a real emergency occurs, the first responder can apply the techniques they practiced with an avatar, better trained and more assured of a successful outcome. Research has shown that learners interact socially with, and are influenced, by avatars. In fact, the research shows that an experience as an avatar can change a person's real-life perceptions. In a study conducted by Yee and Bailenson (2006), it was found that negative stereotyping of the elderly was significantly reduced when participants were placed in avatars of old people, compared with those participants placed in avatars of young people. Within 24 hours of watching an avatar like themselves run, learners were more likely to run, compared to watching an avatar not like themselves running, or watching an avatar like themselves loitering. It has also been shown that if learners watch an avatar that looks like themselves exercising and losing weight, they will subsequently exercise more in the real world, compared to a control group where this was not the case. Learning to use avatars in intelligent situations within a training event will become pervasive, and learning to use the avatar (or becoming comfortable with the technology required to use the avatar) will be fundamental. The telecom industry must teach service personnel how to correctly service equipment; there are many variations of the equipment as well as associated problems. Examining a piece of equipment and sorting out the service issues with the appropriate fix is better practiced in a virtual environment rather than atop a 45-foot utility pole. A bot could be placed in a simulated restaurant or retail establishment to act like an unruly customer and the player-learner can practice managing the situation in real-time using voice chat (voice over Internet Protocol or VOIP). In another example, a bot could be used to virtually call a practice customer service desk so that representatives master resolution strategies.

In real-time they practice the simulated event, in real-time there are interruptions, and in real-time they practice as needed. Needless to say, software sophistication is involved. By augmenting traditional answering to questions with avatar constructs, the verification dynamic is matched closer to reality since the test builder must consider questions and answers that actually play themselves out. For example, in a multiple-choice quiz, a test taker will read, digest, and answer the quiz. Even when answering correctly, there is little assurance that practical behaviors were modified; and certainly, it does not ensure that successful behaviors are accessed if a real-life scenario plays out. Practicing in a simulated situation provides a real-world example that is as close to reality, as is possible, and allows multidimensional practice, and in effect creates a comfortable environment for dealing with real-world situations.

Question: In your opinion, what are the three most important ways that a game-based virtual environment enhances learning?

Thomas: Karl Kapp's definition of gamification is the best one that I have come across. He said, "Use game-based mechanics, aesthetics and game thinking to engage people, motivate action, promote learning, and solve problems." Gamification incorporates game mechanics, dynamics, and aesthetics; these are basically the interactions and experiences between player and system used in the learning and development field to improve performance. Games include a compulsion loop that drives the player to generate an action for a reward, which leads to a new action and a new reward; when the loop is rich, the player wants to continue playing (learning) and gaining rewards. The game interactions are designed to drive engagement by providing meaningful feedback and reinforcement for learners. When a learner wants to learn, the feedback becomes meaningful even when it is negative. That is why game players will participate in seemingly endless compulsion loops, playing robotically even after repeatedly seeing "Game Fail." They will fire up the game again with the goal of conquering the game—it is about winning. Talented designers of learning games bite into the winning to promote learning achievement. With that in mind I can answer the question related to what are the three most important ways that a game-based virtual environment enhances learning:

Total immersion stories/challenges that incorporate evocative interaction with the learning concepts
Opportunity for repetition followed by meaningful feedback
Avatars to allow the "player" to embody the tasks and the actions

It is impossible to speak of game-based learning without mentioning the learning games guru, Dr. James Paul Gee at the University of Wisconsin-Madison, who said, "Good games offer players a set of challenging problems; it, then, lets them solve these problems until they have virtually routined or automated their solutions. Games then throw a new class of problem at the players, requiring them to rethink their now, taken-for-granted-mastery, learn something new, and integrate this new learning into their old mastery." The other benefit that a game provides is a way to work in repetition in a natural and fun way. Repetition is critical to learning, and that is different from rote memorization, where context is not necessarily the objective. A long playing game can include repetition via a distributed practice learning plan; repetition does not have a numerical order requirement. So, for instance, a game can jumpstart motivation because of its nature. It's the pleasure of the thing; people play games because they want to, not for safety compliance training. By creating game-like conditions, learners take a path which includes reasons for participating in courses. A game experience is not just points, badges, and leader boards. It should include a backstory, or a reason to explore a world, and provide challenges that take the learner on a journey that allows them to make decisions that have consequences, even giving them the ability to connect with other players socially. Let the learner prevent accidents, respond to them accurately, make ethical choices, or provide excellent customer service while progressing through compelling game challenges; they earn badges that demonstrate competence based on the decisions to challenges. It was a newspaper story that I wrote in the early 1980s that led me to game design as a profession. I stumbled into a game store and found a post on the bulletin board by the County Health Director of the small town where I was county health reporter. My contact was looking for a game partner to play Avalon Hill games. I wrote a two-page spread on the popular role-playing game *Dungeons and Dragons* and on the Avalon Hill games; I was sure that my contact would find a playing partner. Months go by after the article was published, and during a routine County Health call, I asked him if he had found a partner. To my astonishment, he had not. By this time, I knew how complex the games were to play, but I told him I would become his partner. We played the strategy-based board game called *Arab-Israeli Wars* (published in 1977 by Avalon Hill). I learned about war, as I used the cardboard chips on a hexagonal grid. Sorting out troop strength, determining morale, understanding the importance of terrain within an historical event, I began to play games that took weeks to complete. I began to care about my troops. Initially, I played by trying to get away from my opponent. I quickly began to understand that in the middle of a desert there is no place to hide, I had to shoot. Then I wanted to win.

3.7 CHAPTER SUMMARY AND FINAL THOUGHTS

Let us try and collect the major points of the preceding chapter into salient points. You don't need to go read all the books I have mentioned to be a virtual environment designer, but having them around in your library or on your e-reader is a good way to stay inspired, and to find new parts of the vizome in your virtual experience. So, in the order of appearance, as they say, here are the key points from this chapter.

1. The vizome is always there, so look for the plateaus where it all comes together.
2. Work with the "and, and & and."
3. Design for when.
4. Design for social interaction.
5. Design for semiotics and the four spaces.
6. Find your special point of view on each project.
7. Set up a creation system that includes physical, psychological, and digital tools.

Tune into the design voice inside yourself. Pay attention to what you see, hear, feel, and experience all the time. Interpret the sensations for yourself, and try to understand opposing or conflicting approaches to the problem. Like the dream architects in the film *Inception*, we are creating the virtual world with our imaginations. Try to design for the "city of the mind," the collective nonphysical environment that each participant has enfolded into their consciousness.

REFERENCES

1. I refer to the many instances of architecture inspiring poetry, and architects writing poetry such as the "Poem of the Right Angle" by Charles-Édouard Jeanneret-Gris or "Le Corbusier." *Wikipedia*, s.v. "Poem of the Right Angle," accessed June 14, 2014, http://en.wikipedia.org/wiki/Poem_of_the_Right_Angle.

2. Gilles Deleuze and Félix Guattari, *A Thousand Plateaus: Capitalism and Schizophrenia*, trans. Brian Massumi (University of Minnesota Press, 1987), 7.
3. Gilles Deleuze and Félix Guattari, *A Thousand Plateaus: Capitalism and Schizophrenia*, trans. Brian Massumi (University of Minnesota Press, 1987), 8.
4. Gilles Deleuze and Félix Guattari, *A Thousand Plateaus: Capitalism and Schizophrenia*, trans. Brian Massumi (University of Minnesota Press, 1987), 9.
5. Gilles Deleuze and Félix Guattari, *A Thousand Plateaus: Capitalism and Schizophrenia*, trans. Brian Massumi (University of Minnesota Press, 1987), 12.
6. Eugene W. Holland, *A Reader's Guide, Deleuze and Guattari's* "A Thousand Plateaus" (Bloomsbury Academic, Bloomsbury Publishing, 2013), Kindle edition, 588.
7. Janet H. Murray, *Hamlet on the Holodeck: The Future of Narrative in Cyberspace* (MIT Press, 2000), 132.
8. Gilles Deleuze and Félix Guattari, *A Thousand Plateaus: Capitalism and Schizophrenia*, trans. Brian Massumi (University of Minnesota Press, 1987), 21.
9. Gilles Deleuze and Félix Guattari, *A Thousand Plateaus: Capitalism and Schizophrenia*, trans. Brian Massumi (University of Minnesota Press, 1987), 25.
10. "Inland: Search for the Sy," YouTube video, 1:07, posted by Ann Cudworth Projects, November 28, 2010, accessed June 10, 2014, http://youtube/i2ZqhGID_Tw.
11. Gilles Deleuze, *The Fold: Leibniz and the Baroque*, trans. Tom Conley (University of Minnesota Press, 1993), 123.
12. For a good overview of the Wiki-tree and how it was used, see Scott Chase, "Gather 'Round the Wiki-Tree: Virtual Worlds as an Open Platform for Architectural Collaboration" (slideshow), 2009, accessed June 14, 2014, http://www.slideshare.net/schase56/gather-round-the-wikitree-virtual-worlds-as-an-open-platformfor-architec tural-collaboration-presentation.
13. Don Norman, "The Three Ways That Good Design Makes You Happy," YouTube video, 12:41, posted by TED Talks, March 9, 2009, accessed June 15, 2014, http://youtube/RlQEoJaLQRA.
14. "Sion Speaks—Apologizes—Hires PR!" *The Chicken Blog*, August 1, 2009, accessed June 15, 2014, http://slchickenblog.wordpress.com/2009/08/01/sion-speaks-apologizes-hires-pr/.
15. Patrick Davison, "The Plight of the Digital Chicken by Patrick Davison, Ep 42," YouTube video, 6:21, posted by Ignite, December 16, 2009, accessed June 15, 2014 https://www.youtube.com/watch?v=3p5d4e5e-7o.
16. Susan Weinschenk, *100 Things Every Designer Needs to Know about People* (Voices That Matter) (Pearson Education, 2011), Kindle edition, 2439.
17. Rachel Metz, "Computing News: A Startup's Plans for a New Social Reality," *MIT Technology Review*, April 6, 2015, accessed April 6, 2015, http://www.technologyreview.com/news/536366/a-startups-plans-for-a-new-social-reality/.
18. Pierre Lévy, *Collective Intelligence*, trans. Robert Bononno (Helix Books/Perseus Books, 1999), 131–132.
19. Pierre Lévy, *Collective Intelligence*, trans. Robert Bononno (Helix Books/Perseus Books, 1999), 133.
20. Pierre Lévy, *Collective Intelligence*, trans. Robert Bononno (Helix Books/Perseus Books, 1999), 137.
21. Pierre Lévy, *Collective Intelligence*, trans. Robert Bononno (Helix Books/Perseus Books, 1999), 13.
22. Yotam Oron, *John Cleese on Creativity*, YouTube video, posted March 23, 2015, accessed June 20, 2015, https://youtu.be/Qby0ed4aVpo.
23. Charles H. Burnette, "Intuition, Imagination and Insight in Design Thinking," 2013, accessed May 16, 2015, https://www.academia.edu/3737350/Intuition_Imagination_and_Insight_in_Design_Thinking.

4 Planning and Prototyping the Design of a Game-Based Virtual Environment

But Linden Labs, the company behind Second Life, had the brilliantly insightful idea to put out into the world what amounts to a multiplayer video game platform with no game!

Mike Azzara [1]

4.1 OVERVIEW OF PLANNING AND PROTOTYPING THE DESIGN OF A GAME-BASED VIRTUAL ENVIRONMENT

Virtual environments, like real-world environments, tend to be designed in accordance with the meaning and message of group goals. If you are designing a virtual environment to train people, or teach people, or engage people in a group activity, then you have probably been introduced to the idea of making these activities more gamelike to enhance the overall efficacy of your efforts [2]. This is the conceptual heart of a game-based virtual environment, it is a virtual environment designed to deliver the meaning and message of your group goals through gamelike activities. What is a game anyway? Is it an environment or scenario that gives us an opportunity to compete or collaborate within a set of boundaries or rules? Does that definition sound like life in the 9-to-5 office? What about the element of "play" or play-based activity? How does that factor in? Most of you instinctively know what a game is, although your taste for what kind of games you enjoy may vary. The overall goal of this book is to deepen your understanding of what goes into designing a game-based virtual environment, and the specific aim of this chapter is to understand game structure, game types, and game players and how this information can be incorporated into your designs. First things first, let us get to a working description of what a game contains and how those components work together. Most, but not all, games tend to have these five things in common: (1) they have rules, (2) they have a goal or goals, (3) they have obstacles that can temporarily block the goal, (4) they have a ranking or scoring system, and (5) during the process of a game you are engaged in a play-based activity. The first 4 of these components fit together like a machine; they help the game function while you play it. These four components also allow gameplay to emerge from the underlying mechanics of the game [3]. Gameplay is the interaction you experience while playing the game; it is the tactics you employ and the strategies you devise. It is separate from the visual or aural components involved in the game. Game mechanics provide for gameplay by organizing how the game is played. They work by defining the rules and structure, such as when challenges, goals, and rewards appear. In Figure 4.1 there is an overview of the interrelationships between these five major game components and the kinds of behavior that emerges from these interactions.

Although we have had games for thousands of years, technology keeps adding new aspects to the games we play. In 1958, when William Higinbotham at the Brookhaven National Lab decided to enliven his display of instruments by creating an oscilloscope-based game called *Tennis for Two*, the visiting public went wild, lining up by the hundreds to play [4]. Video games were born, and for more than 50 years we have

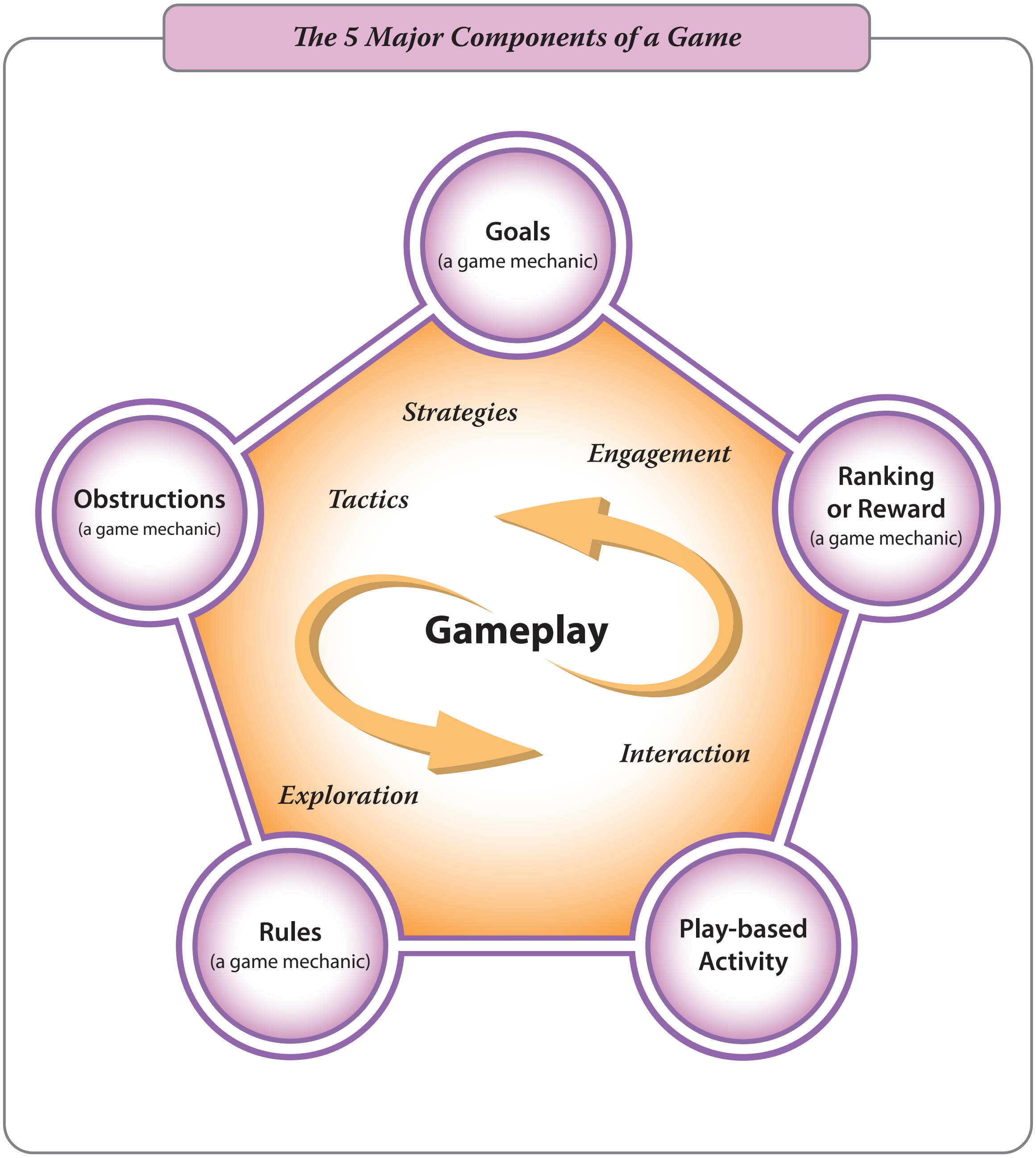


FIGURE 4.1 Overview: Interrelationships between the five major components of a game.

embellished on the idea of blending video images, software, and hardware devices with our games. When people play a video game, they are often transported mentally and emotionally to another frame of mind, and into the world of the game. Think of playing *Dungeons & Dragons* or *World of Warcraft* or *Halo*. When you play those games, your intellectual focus is on the world of the game, you respond emotionally to events that happen within the game; the real world falls away. You also take on a role; a character in the world of the game. Games played in a virtual world are just as varied as they are in the real world. It is entirely possible to play Texas Hold'em or 3D chess in a virtual world. It is also possible to create a narrative-driven, game-based virtual environment within the virtual world that provides the player with an opportunity to enter into an immersive, interactive space. This is a virtual space that creates the complexity of gameplay, game mechanics, and visual and audio effects that you would find in a massively multiplayer online role-playing game (MMORPG) video game. What you might find different is that the gameplay in a narrative-driven environment residing within the virtual world is more like "guided playing." Due to the sandbox capacities of a virtual world, the players can and often do bring their own tools and creations into your game. Imagine being able to play *Bioshock* (published by 2K Games) while your character looks like a giant rooster or to have the local text chatter about what someone had for dinner last night, overlaid on the sound and text responses of your narrative-based environment. In other words, you do not have absolute control over the visitors to your game-based environment. You are creating this environment in a virtual world that is nonlinear, player created, multicultural, multilingual, layered, and interconnected. In short, it lives in the vizome.

4.1.1 What Are the Key Ideas about Designing for Gameplay in a Virtual Environment?

It is almost certain that even if you are not directly developing games in a virtual world, you will be called upon to design environments for them at some point in your career. Games in a virtual world have their own set of unique challenges. Knowing about game design and how to plan for game-based sim requirements is a good set of tools to have in your design repertoire. Here are the key ideas involving designing for gameplay that will be discussed in this chapter:

- Games for virtual environments come in distinctive types that can be identified by their play mechanics and interactive structures.
- Game players come in distinctive types that can be identified by their playing styles and goals.
- The first step in developing a game is to create a game document that defines it.
- Game documents have several parts that describe the story, characters, levels, gameplay, art, sound, and the user interface.
- Design for an educational game is primarily distinguished by the presentation of educational content in a playful way.
- Universal design is part of game design.

At the end of this chapter there is a project about creating four game design documents. These documents will help you define and organize the creation of a game-based environment of your own design. They will serve as the master plan for the entire environment, describing all the various parts and how they will function together, and they provide you with a definitive list of tasks needed to achieve the goal of building a game-based sim in your virtual world. As you read this book and do the projects at the end of each chapter, you will see how the framework of a game-based sim comes together. Let the projects in this book guide your development as a game designer, while you think about how you would like to customize your own game-based sim.

4.2 EMERGENT GAMEPLAY IN VIRTUAL WORLDS

So many games, so little time! Consider virtual worlds like Second Life, *World of Warcraft*, or *EVE Online*. These worlds all contain online social interaction, role-play and economies, and they all develop *emergent gameplay*. Let us define that concept, look at what emergent gameplay is, and why it is important to game-based virtual worlds. Emergent gameplay happens when the player uses the game or environment in a way not anticipated or planned for by the game designer to create a new form of play within the game or environment. This kind of behavior occurs in the real world when kids take a rake and balloon and create a new kind of baseball from it, or organize into teams for hunting "dragons" on summer evenings. We carry that inventive behavior into virtual worlds with ease. In a sense, the player is creating a game within the game, just as *Minecraft* players created the subgame of *Spleef*, an arena-based block destruction competition. Emergent gameplay moves your avatar's narrative firmly into the first person. Emergent game play is "I am, I use, and I do." Of course, emergent gameplay can be collaborative, and frequently seen examples of this are found in MMORGs like *World of Warcraft* and *EVE Online*. A legendary example of collaborative emergent game play happened in *EVE Online* with the Mirial assassination on April 18, 2005. In the world of EVE, Mirial was the rich and powerful head of a large interstellar corporation, and that kind of success is sure to make you some enemies. After 10 months of planning, the members of the Guiding Hands Social Club (GHSC) swooped in, raided the offices of Mirial's corporation, ruined the business, and permanently destroyed the character herself. GHSC made a profit of 20 billion ISK for this contract, and if the EVE currency was exchangeable for real-world dollars, they would have netted just over $17,000 at today's exchange rate. However, it is not all contract killings, destruction, and mayhem; emergent gameplay can be about teaching and learning together, or building new toys. Let us look at some of the more common game types that have developed through emergent gameplay in virtual worlds.

4.3 TYPES OF GAMES FOUND IN VIRTUAL ENVIRONMENTS, AND THE MECHANICS, DYNAMICS, AND AESTHETICS (MDA) FRAMEWORK

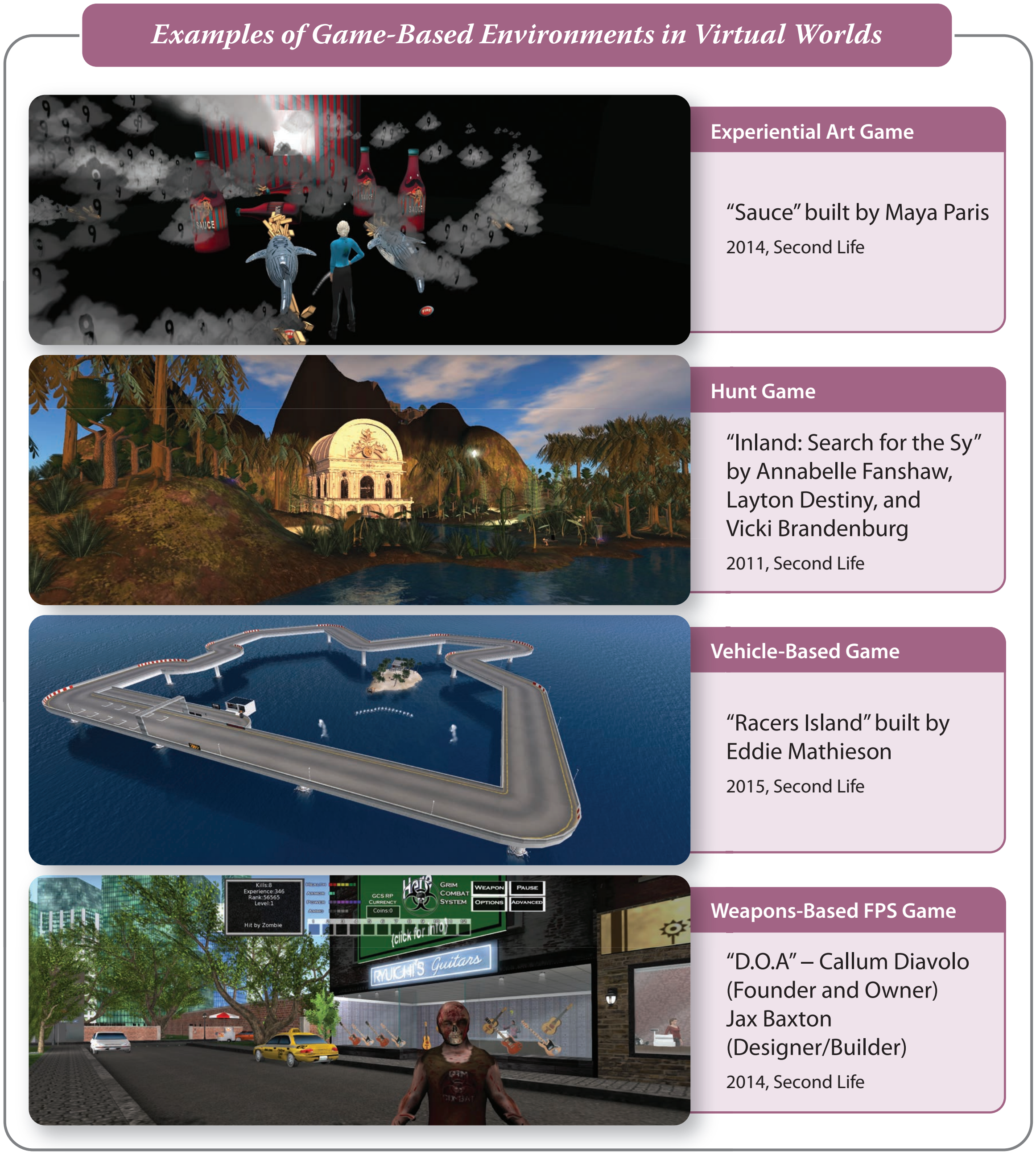
Here is a quick survey of the types of games found in virtual environments around the Metaverse. Represented next in their canonical form, there are, of course, many hybrid games that combine two or more of these formats, such as a hunt/explore game, or an art/vehicle-based game. These game types are the fundamental building blocks for a game-based virtual environment. The mechanics, dynamics, and aesthetics (MDA) framework is a methodology for assessing and understanding the relationships between game mechanics (or rules), the game design, and the game aesthetics or emotional payoff that the player experiences in the game [5]. Let us explore the game types first and then look at them with the MDA framework to see how we can utilize their qualities in our game-based virtual environment. In Figure 4.2, you will see some screenshots from various virtual environments that exemplify these kinds of games.

4.3.1 Exploration Game

The exploration game is the virtual equivalent of sticking a pin in the map, jumping on some transport, and visiting the place. As virtual worlds continue to expand, the desire to visit new places and meet new people will grow among certain types of players.

4.3.1.1 Style of Gameplay

The exploration game can be played as an individual, as a pair, or even in groups. One form is known as "sim hopping" in Second Life; this game is played by utilizing the "click to teleport" functionality of the world



Examples of Game-Based Environments in Virtual Worlds

Experiential Art Game

"Sauce" built by Maya Paris

2014, Second Life

Hunt Game

"Inland: Search for the Sy" by Annabelle Fanshaw, Layton Destiny, and Vicki Brandenburg

2011, Second Life

Vehicle-Based Game

"Racers Island" built by Eddie Mathieson

2015, Second Life

Weapons-Based FPS Game

"D.O.A" – Callum Diavolo (Founder and Owner) Jax Baxton (Designer/Builder)

2014, Second Life

FIGURE 4.2 Various kinds of game-based environments found in virtual worlds.

map. Sometimes, one avatar will go out as a "scout" and send "landmark" links to the others in the group so that they can follow the trail.

4.3.1.2 Design Aspects

Because the sim hopping game is initiated by selecting a location on the world map, the overhead view of the sim should be designed to attract interest. Any large object located below 500 meters in altitude becomes part of the icon that represents the sim or region on the world map. Some sim designers take advantage of this by installing large letters in bright colors or logos above their regions, so that an advertisement for their shop or region appears clearly on the map.

4.3.2 Experiential Art Game

Part exploration, part art experience, part game challenge, the complex experimental art game takes many forms. Usually organized around a concept or point of view on a particular subject, these "games" are often visual feasts of 3D models and interesting graphics/textures. They may be centered on a narrative such as Bryn Oh's *Singularity of Kumiko* [6], which leads you through series of tableaux connected by diary notes and narrative sound clips. Or they may be organized around observations of our society, such as Maya Paris's *Seaside Sauce* [7], which she describes as "a sideways look at dating and romance set in a cockeyed British seaside location and navigated by means of the 'own hair or teeth' (O.H.O.T) algorithm."

4.3.2.1 Style of Gameplay

The style of gameplay in these environments varies from following a defined set of interactive guidelines to playing with all that is provided on the landscape with no particular set of rules or goal. Sometimes game mechanics like guided or restricted movement, role playing, and socialized or team playing are included.

4.3.2.2 Design Aspects

Restricted movement (often by controlling the visibility of the next area), role playing (costumes given out and worn by participants), and play enhancement through a shared social experience are key components. These are game-based experiences built around the artist's style and aesthetics found in their culture and values.

4.3.3 Hunt Game (Idea Based or Shopping Based)

As its name implies, the hunt game initiates a journey across the region or the entire grid that is driven by the goal of collecting objects and/or information. In Second Life, these are very popular since they provide a way for content creators and game creators to mutually support their efforts for audience engagement. Many of these hunts are merchandise based, but they need not be. Players will enmesh with a hunt game that provides an interesting evocative experience.

4.3.3.1 Style of Gameplay

Every hunt needs to guide the players from point to point along a path, while they collect the goods or objects, or experience the next aspect of the ongoing narrative. Often there is a list of landmarks (coordinates of the location inworld) provided by a notecard or listed online. Once the player reaches the destination they must hunt for the specific object, an activity that leads them into exploring the vendor's shop if this is a shopping-based hunt game. In other kinds of hunts there can be a hybrid of narrative gameplay and shopping, as the players follow a plot and collect the goods along the way. Some hunt games are just for information and can be used for teaching purposes.

4.3.3.2 Design Aspects

Consistent branding is a must in this kind of game, especially if it leads the players to many shopping locations. Imagine how difficult it would be to find 30 small objects in a shopping mall if there was not some sort of identifiable wrapper or tag on it. Sometimes the hunt is combined with a screen-based heads-up display (HUD), and this device may keep score, provide clues, store landmarks, or augment the narrative for the player. Some method to prevent players from "seeing" the location of objects unless they walk their avatars around the premises may be necessary. The designer should strive to make the hunt as varied and interesting as possible.

4.3.4 Vehicle-Based Game (Driving, Flying, or Boating in Competition)

Car racing, boat racing, and aerial dogfight flying competitions are all part of any virtual world with a good physics engine, and playable for anyone with a relatively fast Internet connection. Elaborate racetracks have been created in OpenSim where large expanses of land are more affordable, and in Second Life, competitive vehicular use happens on the public lands, in their sky space, and on public waterways.

4.3.4.1 Style of Gameplay

There are two basic forms of gameplay in this genre: competing for time-based results and competing for physical survival. These are not mutually exclusive, and gameplay must be guided by rules that determine the limits on the kind of behaviors that are allowed in the game.

4.3.4.2 Design Aspects

Obviously the vehicular design has significant influence on the race; not only must the vehicle conform to building and scripting standards for the physics engine of the virtual world, it must also be designed for ease of use by the driver or pilot. This includes visibility and usage of HUD designs as well as the overall configuration of the vehicle's structure and the location of the avatar's camera viewpoint. Design of the racecourse also has a great impact on the overall quality of the racing game experience. Details make the difference.

4.3.5 Gun-/Weapon-Based Game (Push-Activated Regions)

Weapon-based, shooter games are an old, venerable game type, first introduced in the early 1990s with games like *Doom*, published by id Software. These kinds of games are called first person shooter (FPS) games because the camera view is limited to a forward angle, similar to mouselook or mouse cam, so that the player can aim the gun at the target.

4.3.5.1 Style of Gameplay

The styles of gameplay can vary from simple competitive shooting of targets, such as an archery contest, to a full-blown, blood-spattered battle scene full of rampaging zombies. Typically these games are designed to increase the player's adrenaline level with graphic imagery and fast action. Often they combine exploration of complex environments, such as a moon base, a bio lab, or archeological site, which provide opportunities to continuously discover new targets.

4.3.5.2 Design Aspects

Obviously the designer should provide a target-rich environment in this kind of game, but that does not mean lining them up like ducks in an arcade booth. Moving targets, pop-up targets, and scaling targets all will add to the challenge for the player. Of course, scoring and player health conditions are part of the vital information the player needs to compete and survive in this environment. This information is often displayed on an HUD so players can track their status as they traverse the environment.

4.3.6 Using the MDA Framework to Define Your Design Approach

In their short but powerfully effective paper about the MDA (mechanics, dynamics, and aesthetics) framework, Robin Hunicke, Mark LeBlanc, and Robert Zubek established a methodology to guide the designer during the creative thought process [5]. They use the word *aesthetics* to define the player experience in the game and have defined some predominant game aesthetics with these eight terms:

Sensation—Game as sense-pleasure
Fantasy—Game as make-believe
Narrative—Game as drama
Challenge—Game as obstacle course
Fellowship—Game as social framework
Discovery—Game as uncharted territory
Expression—Game as self-discovery
Submission—Game as pastime

Of course, your game-based virtual environment can incorporate several of these categories at once. For instance, you could have a fantasy-based, challenge-oriented discovery game environment (*Cinderella* told from the prince's point of view as he travels the kingdom seeking the owner of the glass slipper), or a narrative-based fellowship game environment (*The Maltese Falcon* meets Facebook). There are many possibilities, and it is your job as the designer to find the right aesthetic recipe to combine with the other two pillars of the MDA framework: *mechanics* (the rules and components of the game) and *dynamics* (the system behavior while the game is being played).

Let us analyze the five game environments (exploration, experiential, hunt, vehicle based, and weapons based) within this framework to see what we can discover about the relationships between mechanics, dynamics, and aesthetics. Please refer to Table 4.1.

As you can see, the mechanics, dynamics, and aesthetics of these five games are closely intertwined. Just by tweaking the mechanics, the game designer can greatly affect the player's dynamic experience in the game, which will in turn alter the aesthetics the player experiences. For example, suppose you initially create a game-based virtual environment that encourages the player to discover all aspects of downtown Athens, Greece, circa 507 BCE (just as they invented democracy). Later on, you change the game mechanics by adding an "info hunt" that challenges your visitor to collect info-badges about the history of Athens and to do it within a certain time limit. By shifting the mechanics, adding in the hunt and time limit, you have changed the dynamic from exploration to competitive exploration. That will add the aesthetics of fellowship, challenge, and possibly narrative to your preexisting discovery aesthetic. Given that the game mechanics are instantiated by the game designer's imagination, and code writer's abilities, the possibilities are limitless. Playtesting and fine-tuning after that will determine what works best for your design. Hunicke, LeBlanc, and Zubek sum it up this way: "Fundamental to this framework is the idea that games *are more like artifacts*

TABLE 4.1
Game-Based Virtual Environments and Their Relationship to the MDA Framework

Type of Game-Based Virtual Environment	MDA Framework			Designer Keywords
	Mechanics	Dynamics	Aesthetics	
Exploration (map-based travel game)	Maps, no-fly zones, pitfalls, puzzles, and multiple pathways	Self-directed or social collaboration, shared knowledge	Challenge/ fellowship, discovery	Random, free-form, inventive, ad hoc
Experiential (art-based story game)	Notecards, 360 environment, location-based actuated sound and visual elements that relate to the artist's theme	New visual perspective, shared experience, animated narrative elements, guided tour through it	Sensation/fellowship, discovery/ expression, submission	Meaningful, emotional, political, insightful
Hunt (scavenger or shopping-based location game)	Possible narrative structure, list of landmarks, time limit, a "theme," goods to collect, prizes awarded	Collaborative gain, status confirmation, social outreach, learning about new things in world	Sensation/challenge, fellowship/ discovery, submission	Unpredictable, rewarding, challenging
Vehicle (physics-based racing game)	Vehicles, challenging race course, time limit, health challenge, prizes, leader board	Speed runs, and player's competition, social networking, visceral adrenaline-based interaction	Challenge/ fellowship, discovery/ submission	Skillful, iconic, exciting
Weapons (attack and defense game)	Threatening enemies, obstacle course, moving targets, leader board	Historical reenactments, team play competitions, social networking, visceral adrenaline-based interaction	Fantasy/narrative, challenge/ fellowship, submission	Energetic, challenging, satisfying

than media. By this we mean that the content of a game is its *behavior*—not the media that streams out of it towards the player" [3].

4.4 DESIGNING FOR GAME PLAYER TYPES

Just like there are "dog people" and "cat people" in the world, in virtual worlds and MMORPGs there are different types of game players. Since online gaming in virtual worlds began with *Maze War* in 1974, people have played games for different reasons, and their gameplaying behaviors have changed and diversified as the variety of games to play has increased. As a designer, you should know who your audience is, who plays your game, and why they do. Let us explore some of the information that has been discovered by people who study games and the gamers who play them.

4.4.1 Dr. Bartle, Dr. Yee, and Classifications of Game Player Types

In his seminal book *Designing Virtual Worlds*, Dr. Richard Bartle (PhD in artificial intelligence) mentions player types and the questionnaire known as the "Bartle Test" [8]. Bartle wrote a paper in 1996 called "Hearts, Clubs, Diamonds, and Spades: Players Who Suit MUDS." In this work, he defined game players by their "character theories" and divided them into four groups of characters: killers (club suit), achievers (diamond suit), socializers (heart suit), and explorers (spade suit). This paper is accessible at http://mud.co.uk/richard/hcds.htm. Based on that paper, Erwin Andreasen and Brandon Downey created the Bartle Test of Gamer Psychology, an online test that analyzes your replies to 30 questions and defines your gamer character [9]. Currently available at the gamerDNA website, http://www.gamerdna.com/quizzes/, this test has been taken over 800,000 times and maintains a huge database of player responses. Take it and find out what kind of player you are. Were you identified as an explorer? Take it again and try to be identified as one of the other characters. This will help you gain a basic understanding about what kind of gamers play in your game-based sim and how to attract other types. Dr. Nick Yee (http://www.nickyee.com/) is a social scientist who works with virtual worlds to study how people utilize them for interaction and socialization. Yee (PhD in communication) has studied the Bartle Test and provides a more nuanced approach in his paper "Motivations of Play in MMORPGs: Results from a Factor Analytic Approach." This is available online at http://www.nickyee.com/daedalus/motivations.pdf. In this work you will see the analysis of player types correlated to gender, how factor analysis gives different results from Bartle's, and firsthand responses from players about why they play in virtual worlds and MMORPGs. Interesting game player incentives like "nurturance motivation," "making a difference," "the search for self," and "the search for youth" are examined [10].

4.4.2 How to Give Your Audience What It Wants While You Design What Interests You

The inertia of not being able to decide what you should create for your game-based virtual environment can be overcome if you remember these three things:

1. Know what you like to do and design a game about those things.
2. Design for yourself and it will be full of passion.
3. Design with passion, and your audience will find you.

When asked how to give game players/sim visitors what they want, Kiana Writer of Madpea Productions (http://madpeagames.com/) said, "I personally go with my gut feelings. Instead of thinking 'what would sell,' 'what do people want,' I actually take the time to interact with the community of our players and truly get to know them. While engaging the community I get a lot of feedback on what the players have enjoyed in the past and what their wishes are for the future. However, that doesn't mean that we give them what they want, it's only a rough idea of the parameters we're in … and usually we break them still. I strongly believe in my own vision for being innovative and thinking outside of the box. If I feel that an idea is worthwhile executing into a game, it usually becomes a success, because it has been something so new that the players did not even know they wanted it."

Another important factor is to do your research; make the design sing with the details. When you take the time to look up visual and textual references to the time period and style of the genre you want to create in, you reap the rewards of a deeper immersive quality, and the bonus of discovering new inspiration along the way. Google search is one place to start and that certainly makes it easy to scratch the surface on how your design should look. You now have a handy excuse to watch old movies (a tax deduction for the design professional) and to visit lots of museums and historical sites around the world. If you are pitching a game-based sim to a client, being able to knowledgably discuss your audience demographics is a plus. Sites like KZero Worldswide (http://www.kzero.co.uk/) offer insight into virtual worlds and infographics about who plays in them. Another source of inspiration might be the book *Masters of Doom: How Two Guys Created an Empire and Transformed Pop Culture* by David Kushner. In the early 1990s, John Romero and John Carmack, two of the founders of id Software, were significant contributors to the online video game industry. They had definitive roles in creating a gamer culture, and the symbiotic collaboration they developed created games that impacted the growing online gaming world. Their two most successful games, *Doom* and *Quake*, both published by their company id Software, touched off an unprecedented cascade of game development worldwide. These games created the FPS multiplayer online genre. They knew their audience.

4.4.3 Making the "Grind" Work for You and Your Game-Based Sim Design

Dr. Nick Yee describes *EverQuest* as "a Skinner Network event, where each Skinner Box is tailored to its host's needs and reinforcement schedule, and where individuals can interact with each other without sacrificing the integrity of their own construct" [11]. Those of you who took Psych 101 will remember how B. F. Skinner popularized the theories of operant conditioning and showed how behavior that is rewarded will be repeated. *EverQuest* has mastered this methodology, and its long-term success underscores the importance of incorporating a reward system into your game-based sim design. What choices you make regarding reinforcement or punishment of your players' behaviors and how you choose to deliver the positive and negative stimulus to them as they traverse your virtual environment is up to you as the game designer. Some designers have a knack for setting up complex reward and punishment systems, and other designers are better at designing the environment that physically challenges the player. Both skills are necessary to create interesting gameplay for the player, and an environment they will want to visit again and again.

4.5 DOCUMENTS, PROTOTYPES, AND GAME-BASED VIRTUAL ENVIRONMENTS

If you ask 10 game makers about game design documents, it is likely you will get 10 different answers. The debate as to what kind of game design document to use and how you should use one is continuous on gamer-friendly websites like Reddit and Gamasutra.com. In this section, we will look at the variety of documents you can utilize while you plan the design and function of your game-based sim. Bear in mind, it's not one size fits all; each project you do may require a different kind of game document, and it will probably need more than one. For instance, for a small board game you will play on the virtual terrain, a simple description of the game board and a list of its rules may be all you need to create. For a role-playing historical adventure game that you are building with a team of 20 people, much more documentation is needed, and everyone will need access to this information while they create the game-based sim. In this kind of project, the designers will need to have a list of levels, the artists will need research references, the coder/programmer/scripter will need to understand the game flow and mechanics, the animators will need to see how the non-player characters function, and so forth. As Stone Librande said in his master class, "the act of creating the document is the act of designing the game" [12]. In fact, Stone is an expert at game design and the art of game document creation, and frequently speaks about the subject at the Game Developers Conference (GDC). You can review many examples from these GDC presentations on his personal site at http://stonetronix.com/. In Table 4.2, there is an overview summary of the basic kinds of game documents and how they can be used. As we continue on in this chapter, you will learn about game documents that have been specifically tailored to game-based virtual environments.

4.5.1 How to Create Game Documents Specifically for Game-Based Virtual Environments

Let us talk specifics about how to make game design documents for your game-based sim. You may be building this for entertainment, or for education, or just because you like to build things in virtual worlds. You may be a team of one or many people. Planning is still essential, no matter the scope of the project or why

TABLE 4.2
Pros and Cons of Game Design Documents

Type of Game Design Document	Pros	Cons	Reason to Do This Kind of Game Document	Online Example
Text-based/verbal documents	Format is accessible for everyone in many formats	People often will not read long text documents	Excellent way to begin initial planning for your game	For a document specific to game-based virtual environments: http://www.anncudworthprojects.com/
Image-based/storyboard design documents	People will understand the concept readily if it is clearly illustrated	Can be laborious and may require special skills	Very useful for engaging and inspiring the team	For a storyboard template to download: http://www.anncudworthprojects.com/ To make a story board with 3D models: http://wiki.blender.org/index.php/Doc:2.4/Tutorials/Game_Engine/Storyboards
Website or wiki-based design documents	Easily accessible and can contain words, pictures and video	Requires upkeep so that the pages and links are attended to, and there is a slight learning curve for usage	If properly organized and maintained, this method provides a wealth of up-to-date information for all	https://www.dokuwiki.org/dokuwiki http://foswiki.org/ The wiki about wikis: http://en.wikipedia.org/wiki/Comparison_of_wiki_software

Source: Based on lecture by Stone Librande at the NYU Game Center, July 2014.

you are doing it. Planning your build by using game documents saves time and effort, and helps you refine your ideas in a less labor intensive way.

4.5.1.1 Textual Game Design Document

Although you ultimately may not show this document to anyone, the textual game design document is essential as your first organizational framework. Here is where you fill in your shopping list of elements you want to include in your game-based sim and start to articulate your key ideas. At the end of this book in Appendix A, you will see game-based sim building guide document. You are invited to use this document and the content in this book to invent your own game-based sim with gameplay that utilizes the stories from your own imagination. On my website, http://www.anncudworthprojects.com/, you will find this document ("Appendix A: Game-Based Sim Design Document"), which you can download and use for a template.

4.5.1.2 Image-Based/Storyboard Type Game Design Documents

Image-based game design documents run the gamut from snazzy concept sketches done for pitching the game idea to nuts and bolts illustrations about how the parts of your game-based sim fits together. You do not have to be an expert illustrator to create image-based game design documents; there are lots of ways to get it done. One of my favorite techniques is to build a rough model of the game-based sim in SketchUp (http://www.sketchup.com/3Dfor/game-design) and take screenshots of that model to illustrate how the whole build will come together. In Figure 4.3 is my SketchUp model of the game-based sim, which is featured in the

Overview of Game-Based Sim in SketchUp

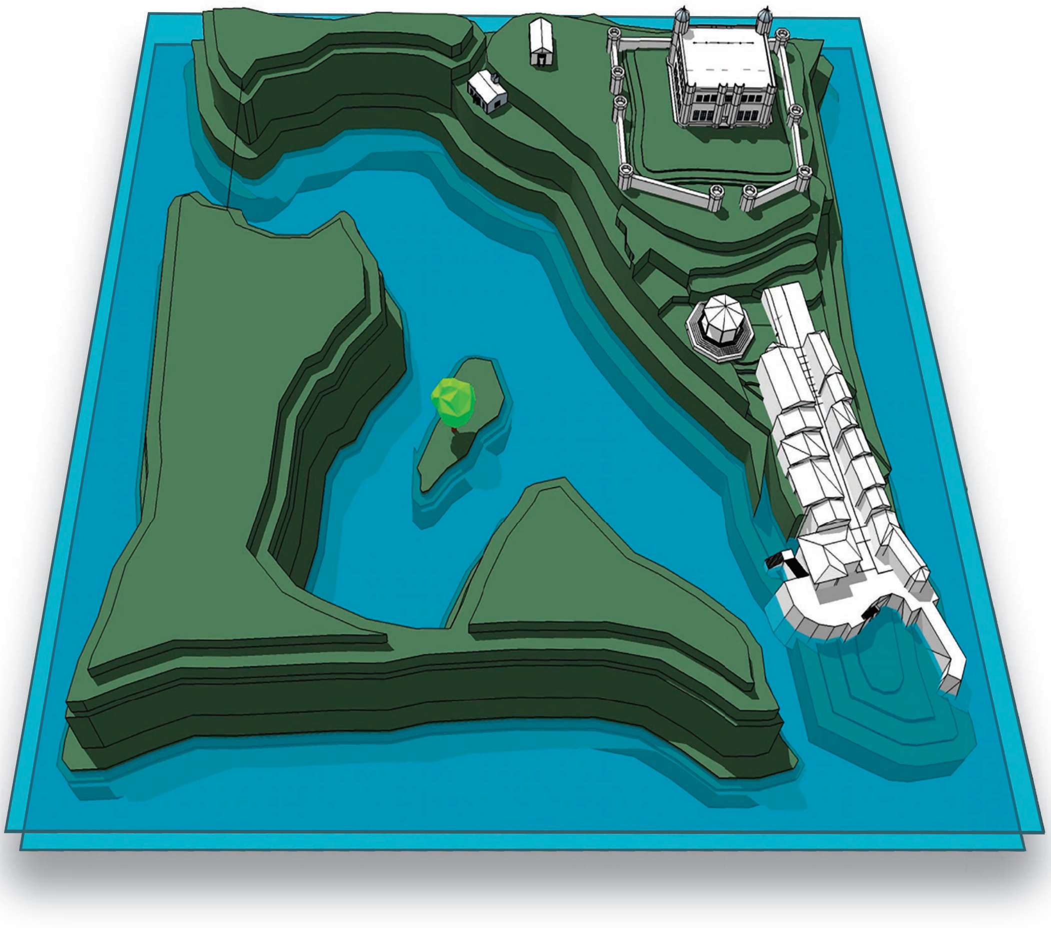
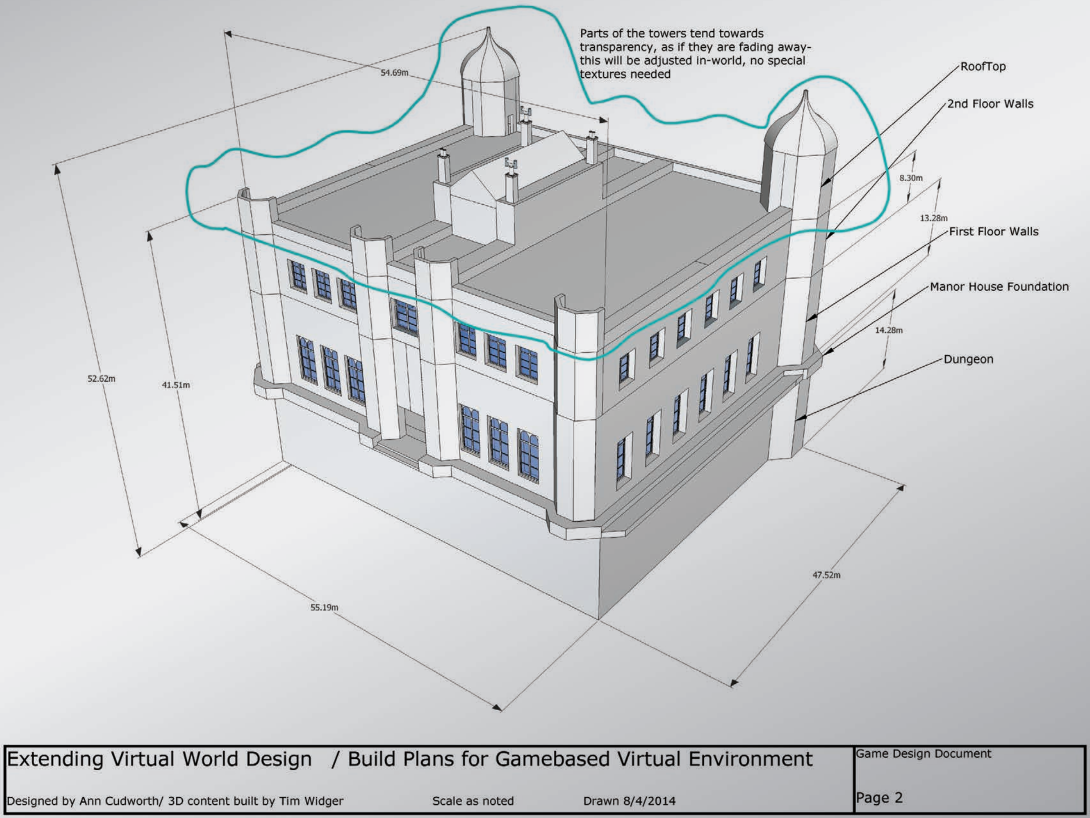


FIGURE 4.3 A 3D planning model, created in SketchUp for the game-based sim utilized in this book.

Estimation and Building Documents


Parts of the towers tend towards transparency, as if they are fading away-this will be adjusted in-world, no special textures needed
RoofTop
2nd Floor Walls
First Floor Walls
Manor House Foundation
Dungeon
54.69m
8.30m
13.28m
14.28m
52.62m
41.51m
55.19m
47.52m

Extending Virtual World Design / Build Plans for Gamebased Virtual Environment

Designed by Ann Cudworth/ 3D content built by Tim Widger Scale as noted Drawn 8/4/2014

Game Design Document

Page 2

FIGURE 4.4 Example of a build document used to guide construction of a game-based sim.

projects throughout this book. As you no doubt realize, a 3D model like this will provide you with an environment that has multiple points of view, and will lend itself not only to the creation of build drawings as you can see in Figure 4.4, but also storyboard backgrounds, which you can see in Figure 4.5.

For a good overview of how SketchUp can be used with game design, watch this video showing how Art Director Robh Ruppel used SketchUp to design the environments for *Uncharted 2: Among Thieves*, a third-person shooter video game developed by Naughty Dog (published by Sony Computer Entertainment for the PlayStation 3). The video is available on YouTube at http://youtu.be/8mkPRmqUlFw. SketchUp is not the only 3D modeler that can help you do image-based game documents. Blender, the free 3D modeling software, also has the capacity to create "animatics" or animated storyboards from your hand sketches. Because Blender has an editor tool in it, you can load your hand-drawn or screenshot images; play them in a slide show sequence, with effects like pans and wipes between the images. A quick search of Google with the keywords "storyboards with Blender" will bring up lots of tutorials for you to try.

4.5.1.3 Wiki-Based Game Document Organization

By now, you are probably beginning to feel a bit overwhelmed with the amount of documentation and data that designing a game-based sim can involve. This is especially true if you are working with a large group of people, and they are all generating content, comments, and creative input. For many years, wikis have been around, helping people get information organized, and it is not surprising that many game developers utilize them when they build games. The word *wiki* is from the Hawaiian word *wikiwiki*, which means "quickly." In 1994, Ward Cunningham needed to create a website that would allow for visitors to actively engage with the content and retrieve data quickly, and the WikiWikiWeb was born. More than 20 years later, the wiki has become part of our daily experience, as we access Wikipedia or other sorts of wikis to seek information about topics we are interested in. You may already use a wiki format in a blog you write on WordPress. If not, there is a plug-in you can add to create one there. In fact, wiki software is available in a wide variety of formats and prices, from professional, enterprise-based versions to freeware-personal versions. You will find a useful list on Wikipedia contributors on the page titled "Comparison of wiki software" [13]. If you have a dedicated wiki developer/administrator for your company, you can maintain a large wiki that will help cut down on e-mail and other forms of confusion. If you have a smaller footprint and just need one for your team of three, Google Sites may have the solution. Google Sites offers a wiki template, free to use and edit for your project, as long as you have an account with Google. I am using a Google docs wiki to track the progress of this book, and the game-based sim we are building for it. Feel free to take a peek at this link: https://sites.google.com/site/evwdpuzzlegame/home. Whatever wiki you decide to use, remember it needs useful and accurate data from its member contributors and an administrator to keep it tidy.

4.5.2 Prototypes for Game-Based Sims

By now, you have quite a few arrows in your game designers quiver. You have learned about text-based game documentation and how that helps set up the initial conceptualization for your game-based sim. Image-based storyboards can be used to preview how the game will look and feel in 3D space, and the online wiki will help you and your team stay on track during this complex process. All of this is great groundwork for building a fun game-based virtual environment, but what about the mechanics and the finer points about how a game-based sim is experienced? You may have a favorite idea about using a teleport system to get the player from one area to another or a vision of dark, forbidding mazelike walls, but will these elements really add to the game play? How will the sequence of locations of your virtual space affect your player's experience of the game?

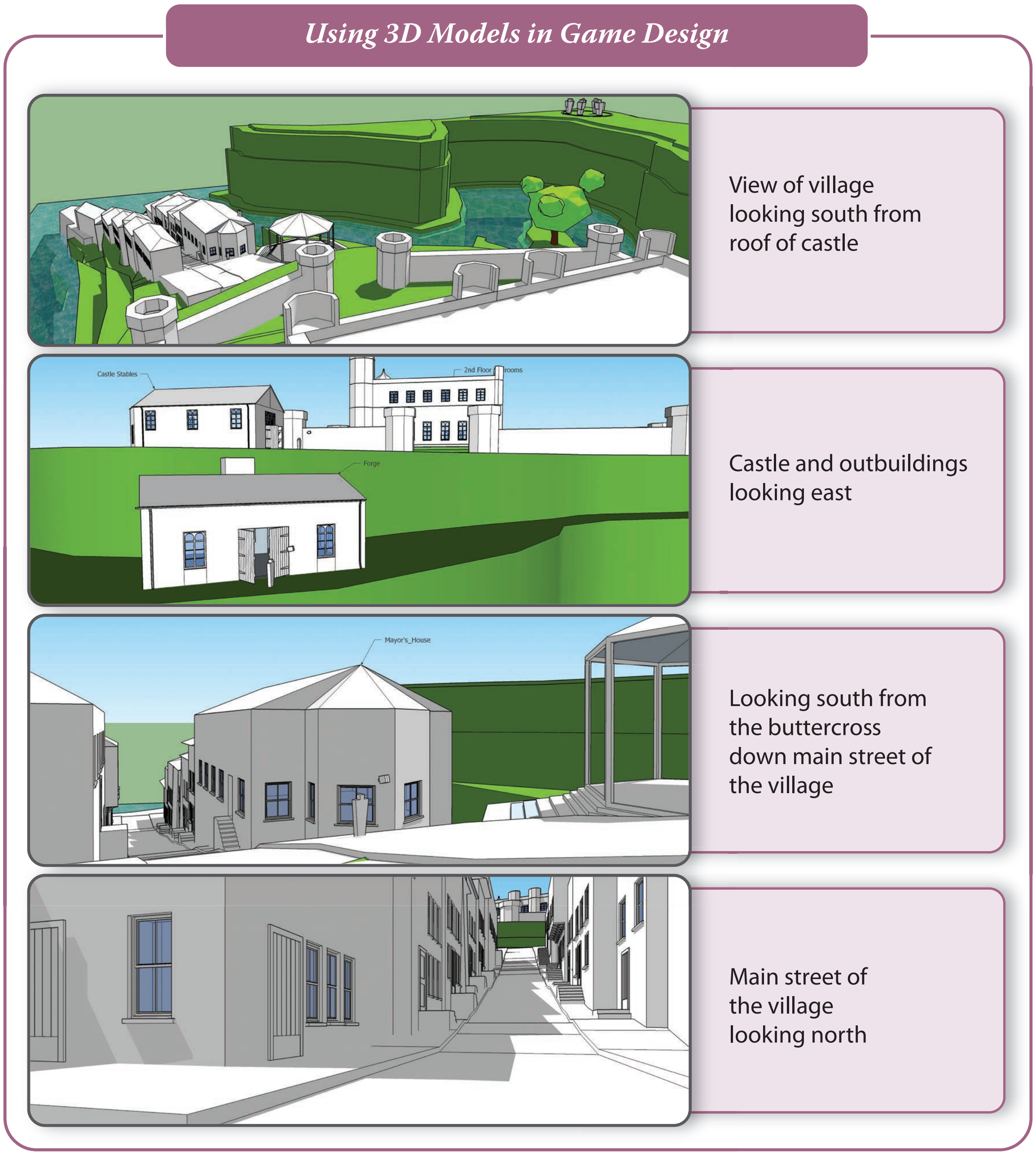

Using 3D Models in Game Design
View of village looking south from roof of castle
Castle Stables
2nd Floor rooms
Forge
Castle and outbuildings looking east
Mayor's_House
Looking south from the buttercross down main street of the village
Main street of the village looking north

FIGURE 4.5 Using the 3D model for storyboard backgrounds in the planning of a game-based sim.

4.5.2.1 Popsicle Sticks, Paper, and Pipe Cleaners

Using paper prototypes to “play” the game on your worktable becomes an invaluable process for understanding the mechanics and flow of your game without committing to building the entire environment first. Head over to your local crafts or gaming store and stock up on the following items:

- Large pad (18″ by 24″ minimum) of cheap paper, or a roll of 24″ to 36″ wide paper that you can trim off tabletop-sized pieces from
- Lots of colored pens, pencils, and markers
- Wooden or plastic shapes that can represent avatars or structures on the landscape, such as Popsicle sticks, cubes, pyramids, etc.
- Dice, spinners, timers, and other means of determining game turns
- Scissors, tape, and your favorite paper working tools
- Game parts from other games, like chess boards, markers, chips, and tokens

Now follow these steps to set up the paper prototype of your game-based sim:

1. Lay out a sheet of paper on the table between the players and draw a rough outline of the terrain in your sim. Feel free to write the names of places on the “map” if you want to.
2. Stack up bits and pieces to represent any structures and/or obstacles you plan on adding to the landscape. Note: If you have skyboxes or alternative locations not on the terrain in your game, they can be represented by another sheet of paper on an adjacent table.
3. Add in markers to represent any vehicles and other devices in your game.
4. Pick out the objects that represent the avatars and place them at the starting position.

Once you have all that figured out, it is time for a test drive. Keep a notebook nearby to take notes and write down observations. Have a camera ready to document how the paper prototype evolves as you undergo this process. Read over your rules and start to play the game. Anytime you hit a snag, revise the layout on the table or the rules, document the changes, and start again. In a few hours, you will be able to smooth out the game mechanics (or invent new ones that work better), polish the design, and get a good feel for how the player will experience your game. In Figure 4.6 is a picture I took of the paper prototype that Atlas Chen, Reynaldo Vargas, and I made during the NYU Game Center Workshop in Game Pre-Production and Design Prep with Stone Librande. We chose to prototype *Journey* (published by thatgamecompany), which is a social game played online. Since the landscape is such a big part of this game, we decided to build the desktop with colored cubes and blocks on the grid of a chessboard. To work with the social aspect, we added some rules that required that two players land on the same cube in order to jump over obstacles (jumping is a big part of *Journey*). Movement was facilitated by choosing a three-letter combination of north, east, west, and south. For example, if you wrote “N,E,W” on your turn list, your player would jump northward one square, eastward one square, and westward one square. Because you can see where the other players are, you can tailor your jumping to land on the same square as they are and together jump over the hills. The goal was to traverse the board, and sometimes one of us found it more efficient to go around the hills rather than team up. These discoveries led us to create some more rules and mechanics to foster more social collaboration in the game as we played. Who knew we could learn so much about a game by playing it in a paper prototype?

If you are interested in learning more about paper prototyping, there is lots of information about it online, including videos from GAMBIT (http://techtv.mit.edu/videos/7071-paper-prototyping-your-game-episode-1-part-1).

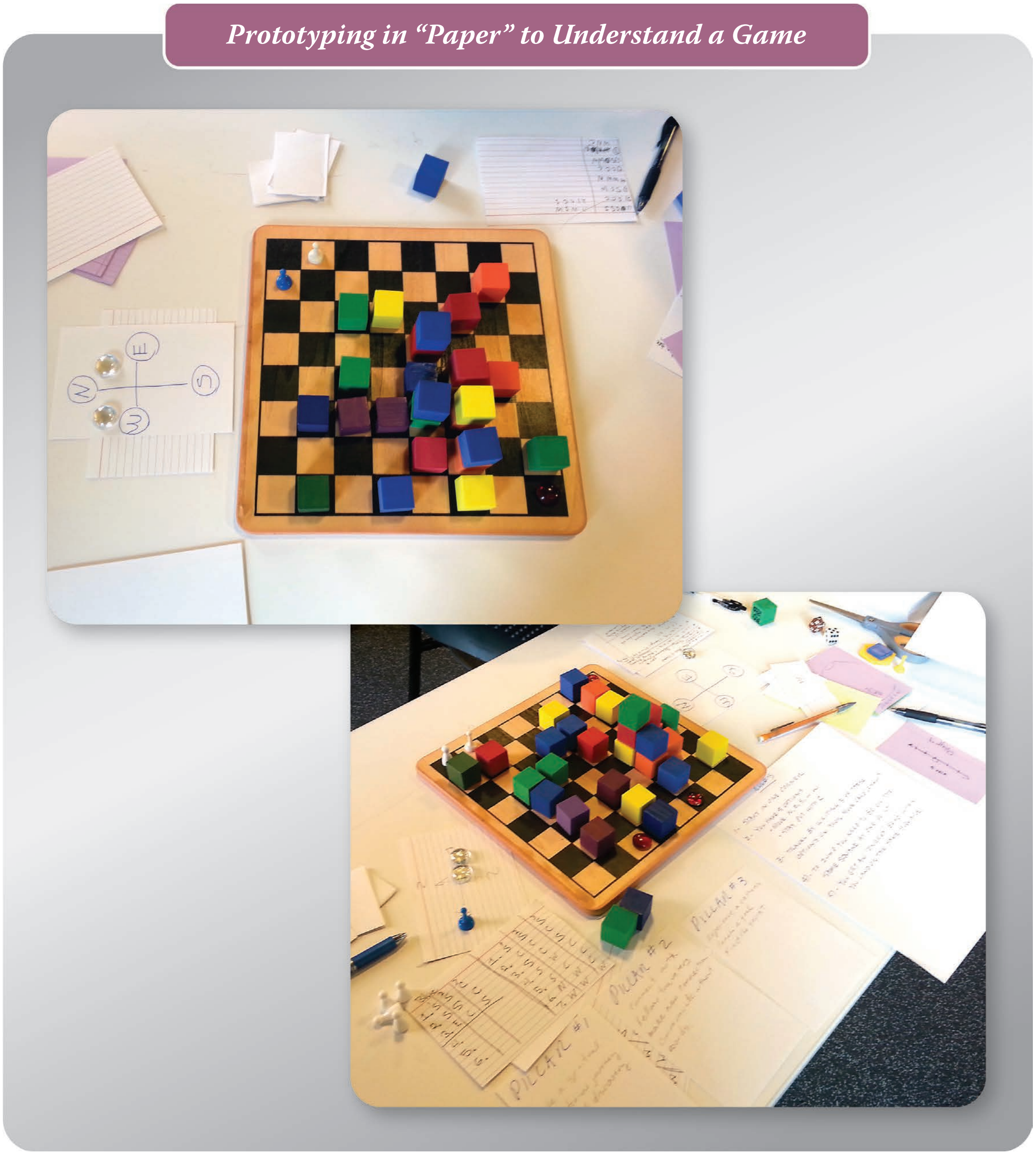


FIGURE 4.6 The *Journey* game (published by thatgamecompany) created as a paper prototype.

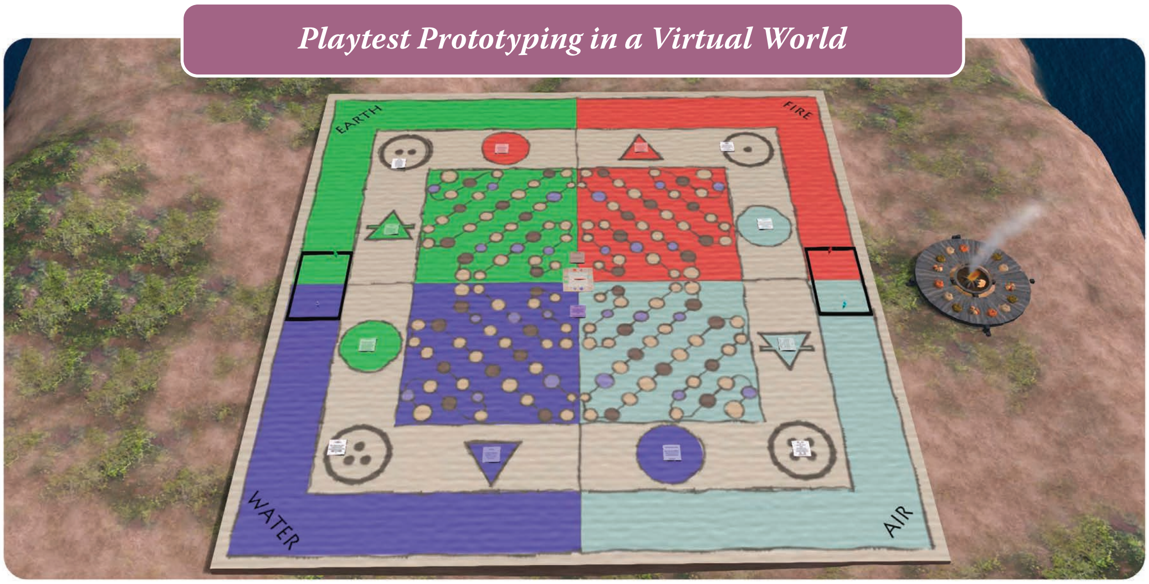


FIGURE 4.7 Screenshot of the playtesting/prototyping space for the board game version of *Tribe*.

4.5.3 Virtual Prototypes for Game-Based Sims

Sometimes you just need to get into 3D for your prototyping. This would be the case when you have game elements crucial to the play that are difficult to prototype in the physical world, such as giant-scaled props. Perhaps you want to test out the environment with another kind of display such as an immersive HMD (head-mounted display) or as an AR (augmented reality) display on your mobile device. These kinds of things can be quickly tested with prototypes from SketchUp Pro or Blender, and by utilizing the Unity platform with the Oculus Rift or other types of HMDs. There is much more about the use of HMDs with game-based virtual environments in Chapter 13. Another reason to use a virtual prototype for a game-based environment is to expose it and the game mechanics you designed to a large social group. You may want to have your game playtested to get opinions from around the world to gauge the clarity of the instructions. Online virtual worlds work well for this process, since all of your playtesters can join your avatar in the virtual play/prototyping space. In Figure 4.7 is a screenshot from the prototyping process we used to test the board game about indigenous peoples and their protection, called *Tribe* (2013 Alchemy Sims).

4.6 PLAYTESTING THE PROTOTYPE AND GAME JAMMING FOR GAME DESIGN DEVELOPMENT

Eventually, you will have to let people outside your development group come to play with your shiny new game, and sometimes that breaks it. Contrary to how you may feel about this, that is a good thing and what playtesting is supposed to do. It is amazing how quickly a playtester can find the weaknesses of your game, find a way to circumvent your planned flow, and pounce on a problem. If you apply your design “judo” properly, you can use the momentum of that discovery to drive the team development cycle around again, throw the problem on its back, and vanquish it. You can plan for design development through playtesting in the following ways:

- Connect with social groups in the virtual world that are interested in design and gaming, and set up an evening of playtesting.
- Connect with the members of online sites like www.boardgamegeek.com, and sponsor a playtesting event.
- Make a game design project part of the class you are teaching about virtual environments and have the students playtest all games that develop.

4.6.1 Design Development and the Game Jam Approach

If you look around, Game Jams are happening everywhere, for every kind of game and game platform. Essentially a Game Jam is like a musical jam session. Many kinds of game developers, designers, and programmers come together for a weekend or an evening to connect creatively while they design a game together. Sometimes there is a topic or theme they have to acknowledge in the design or experience of the game, and sometimes they are collaborating to stretch the limits of a specific gaming platform or game engine.

4.6.2 The Real Game Jam Experience

In reality, a face-to-face Game Jam, like the Global Game Jam (http://globalgamejam.org/), is highly challenging and often inspiring. You are going to meet strangers and create a game with them, possibly over a weekend. In January 2015, I met Ben, Edward, and David at the Global Game Jam site sponsored by NYU Game Center, and we made a board game called *On the Edge of Space* that was nominated for the best design category (https://www.linkedin.com/pulse/nominated-best-game-design-ann-latham-cudworth). It was exhilarating, exhausting, and ultimately we bonded as game aficionados.

4.6.3 Holding a Virtual Game Jam

It is quite possible to have a game jam in the virtual space. Here is a list of things you should prepare as you organize your first virtual game jam.

Space to work—If you have a big OpenSim grid, you can parcel out the land to participants. If you do not, participants can build their games offsite, then bring them in for display and demo at the end of the Game Jam.

Advisory group—Find a group of experienced game designers who are interested in providing advice and guidance as you plan the event.

Tools and resources—Mesh models, basic terrains, and simple scripts that people can use to build a game with are some of the tools and resources you may want to provide.

Advertising—Look for sponsors and online groups who may want to be involved by contributing tools and resources in exchange for advertising at the event and on the website.

Coaches, helpers, judges, and special guests—These are special folks you need to line up to help the game creators with scripting and/or building, virtual world access, as well as serve as judges in the final competition.

Online infrastructure and wiki—Set up space for people to make games on your virtual land. Set up a wiki to countdown to the event and post results on.

Scheduling and planning—Set up a calendar for everyone to follow, and decide on a timetable for the 6 to 12 months leading up to the event. Set up a daily schedule for the event that ensures that the contestants do get some rest and breaks for meals.

Awards and prizes—Decide on categories for the judges, so they can pick the best entries in an organized way. Give them some standards to judge by. Have content to award to the winners in each category.

4.7 PROJECT: CREATING FOUR DEVELOPMENT DOCUMENTS FOR A GAME-BASED SIM

4.7.1 Decide on the Scope of Your Game-Based Sim

This project will introduce you to methodology for developing the game documentation you will need for a game-based sim. For the purposes of the projects in this book, you will be provided with the following 3D content:

- A castle/manor house, with a stable, forge, and outer wall
- A small village that includes the main street, buildings, mayor's house, butter cross (a meeting pavilion), and quay
- Various structures for placement on the surrounding terrain, including a fairy bridge, standing stones, fairy boat, and fairy tree

For the purposes of demonstration, I have provided a game planning document (see Appendix A) for this environment, which you can customize to make any of the game types described in Section 4.3, or even a hybridized version that combines two or more of those types. If you desire to customize this environment for your own purposes, think of these two questions

1. Will the environment be used for teaching, entertainment, personal expression, or something else?
2. What type of game most suits how you will achieve that purpose?

4.7.2 Document 1: The Concept Outline

Step 1: Set up the template. On my website, http://www.anncudworthprojects.com/, is a sample text-based game design document (Game-Based Sim Design Document) that you can use as a blank template for this section [14]. As you can see, the document is an outline format that you can fill in. The key categories are

1. Introduction
2. Design/overall style
3. Terrain design
4. Levels of game
5. Game flow
6. Game components
7. Graphics/textures
8. Sounds and music
9. Technical aspects
10. Schedule

Feel free to add to or reduce these categories as necessary.

Step 2: Choose your game type. You may decide that this is the perfect place to create a history exploration game, or a vehicle-based game, or perhaps a weapon-based game. Take your time and fill it out as best you can. You will find that by putting your thoughts and creative inspiration into an organized, easily accessible document that it will be easier to see how complete the project actually is and where you need more development. Also included in the template are "Inventory Item Prefix" tags, so that when you get around to creating your own game, you and your team will have a standard naming methodology for your avatar's inventories. This will save loads of time when you are in the thick of production and sharing content back and forth.

Step 3: Fill out the form. Take your time with this step, and let the concept for your version of the game jell in your mind before it takes form on the page. You may even want to go back and refine it, once you have done the next three kinds of documents.

4.7.3 Document 2: The Image-Based/Storyboard Document: Planning Your Game

In Figures 4.5, 4.6, and 4.7 you saw some examples of how the image-based/storyboard document can be put together. These examples were made with SketchUp Pro and its sister program Layout. SketchUp is useful because it allows for fast 3D modeling, and Layout allows for you to create views of those models and annotate them with pictures you have grabbed from research and such.

In this part of the project, you will be making a "picture collection" of all the images that you think are important for research, ambiance, and inspiration during the design of your game-based sim. If you are using the content from this book, you will already have the environment, including the castle, village, fairy elements, and riddle clue props. To keep this beginning exercise in game documents to a reasonable size, let us suppose you have decided to add more objects to that preexisting content to create your version of a game for the sim.

Step 1: Organizing the lists. Based on the game you have decided to create in Section 4.7.2, generate a list of the new objects you would like to include in the game. If you are doing a weapons-based game, perhaps the list looks like this:

- Battle axe
- Long bow and arrows
- Trebuchet
- Buckets of boiling oil
- Slingshots
- Broadsword

Perhaps it is a mystery/history hunt, and you want to have objects like these:

- Magna Carta
- Rosetta Stone
- Dead Sea Scrolls
- Mask of Agamemnon
- Death Mask of Pharaoh Tutankhamun

And so forth, as you need to proceed with your game.

Step 2: Creating the story for each object by using pictures.

- Set up one page in your document software (Google Docs, MS Word, or SketchUp Layout, etc.) for each of these objects.
- Start to develop a "story" for each of these objects; think about what kind of interaction they can have with the player when these objects are touched inworld. Is it a magical sword for example?
- Describe with words and inserted pictures what each of these objects will look like. How big, what color, and so on.

Step 3: Develop a sense of how these objects will be used and what they can teach.

- Check out YouTube (https://www.youtube.com/) and other video sources to see how these weapons/props/other objects are used and maneuvered. Link those videos into your documents.
- Will this be a teaching moment? What can each object teach us?
- What rules can you add to promote the use of the object in your game?

4.7.4 Document 3: Paper or Digital Prototype of Game-Based Sim

Step 1: Layout a paper version of your game on the tabletop. Using the craft objects you collected from the list in Section 4.5.2.1, set up a ground plan of your game sim and locate all the objects you worked with in Section 4.7.3.

Step 2: Walk through the game with your playtesters.

- Check the rules. Do they make sense, and do they move the game forward?
- Testplay the game and refine your rules.

Step 3: Keep track of everything.

- Record your refined rules, and take snapshots of your tabletop layout.

4.7.5 Document 4: Building a Project Wiki for Your Game-Based Sim

Step 1: Set up a Google Account if you do not already have one.

Step 2: Use the Google Template to set up a wiki project page (https://sites.google.com/site/projectwikitemplate_en/).

Step 3: Edit the Wiki Project Template to fit your game project development.

- Perhaps it is your class or a group of friends; someone should be designated to manage the wiki and keep it up to date.

4.7.6 Project Conclusion

By now you should be very comfortable with some of the basic game documentation necessary for organizing a game-based sim. Feel free to utilize whatever software helps you get this done. As a game designer myself, I utilized game sites like Gamasutra.com and Massively.com for advice on game planning and documentation, and do-it-yourself sites like Makeuseof.com and Instructables.com for software advice. Feel free to “mod” the templates in any way you desire; game documents are personal things, they cradle our nascent ideas and dreams, as we work to build our game-based environments.

4.8 CHAPTER SUMMARY AND FINAL THOUGHTS

A lot of important concepts have been covered in this chapter. You have been introduced to designing for a game-based sim or virtual environment, which varies from the design for a board game, video game, or card game. As you discovered with the MDA framework, there are certain qualities common to all games. However, few games allow for the huge range in the social contribution, object creation, and storytelling factors that a game-based virtual world does. You learned about the game player types and how emergent play can happen. Creating a game-based virtual environment is a lot to manage and organize, so learning to develop and utilize game documentation that helps you do that is a fundamental skill for any designer. Take these tools and design your ideas!

REFERENCES

1. Mike Azzara, “Virtual Worlds, Second Life & Live Events: A White Paper Describing the Immediate Value Proposition for B2B Media Companies,” accessed November 20, 2014, http://contentmarketingtoday.com/wp-content/uploads/2007/11/azzara-virtual-worlds-and-second-life.pdf.
2. James Paul Gee, “Learning Principles,” (excerpts from *What Video Games Have to Teach Us about Learning and Literacy*), accessed on May 13, 2015, http://mason.gmu.edu/~lsmithg/jamespaulgee2.
3. *Wikipedia*, s.v. “Gameplay,” accessed July 19, 2014, http://en.wikipedia.org/w/index.php?title=Gameplay&oldid=635468803.
4. “The First Video Game?,” *Brookhaven National Laboratory*, accessed July 18, 2014, http://www.bnl.gov/about/history/firstvideo.php.
5. Robin Hunicke, Marc LeBlanc, and Robert Zubek, “MDA: A Formal Approach to Game Design and Game Research,” developed at Game Design and Tuning Workshop at the Game Developers Conference, San Jose, California, 2001–2004, accessed August 16, 2014, http://www.cs.northwestern.edu/~hunicke/pubs/MDA.pdf.
6. Bryn Oh, “Singularity of Kumiko (trailer),” YouTube video, 3:08, posted by Bryn Oh, January 30, 2014, accessed July 17, 2014, http://youtu.be/JONF4tgTh34.
7. Maya Paris, “Sauce,” *Maya Paris* (blog), April 3, 2014, accessed July 17, 2014, http://mayaparisbluestocking.blogspot.com/2014/04/seaside-sauce.html.
8. Richard A. Bartle, *Designing Virtual Worlds* (New Riders Publishing, 2004), 145.
9. Gamer Discovery Quizzes, “Bartle Test,” accessed August 17, 2014, http://www.gamerdna.com/quizzes/.
10. Nick Yee, “Motivations of Play in MMORPGs: Results from a Factor Analytic Approach,” accessed August 17, 2014, http://www.nickyee.com/daedalus/motivations.pdf.
11. Nick Yee, “The Virtual Skinner Box,” accessed August 18, 2014, http://www.nickyee.com/eqt/skinner.html.
12. “Master Class: Stone Librande; A Workshop in Game Pre-Production and Design Prep,” accessed July 12, 2014, http://gamecenter.nyu.edu/event/master-class-stone-librande/.
13. *Wikipedia*, s.v. “Comparison of Wiki Software,” accessed August 21, 2014, http://en.wikipedia.org/w/index.php?title=Comparison_of_wiki_software&oldid=639614514.
14. Adapted from the Google Design Document, accessed June 25, 2014, https://docs.google.com/document/d/1ct5-qyUZC9cAKn-iLUgtOczDkERmPzNNwPLDfT9Hgjs/preview.

5 Designing for Social Interaction in Virtual Spaces

The whole enterprise around spheres and networks—which superficially looks like a reduction, a limitation, to tiny local scenes—is in effect a search for space, for a vastly more comfortably inhabitable space.

Bruno Latour [1]

5.1 OVERVIEW OF DESIGNING FOR SOCIAL INTERACTION IN VIRTUAL SPACES

The purchase of the virtual reality (VR) headset company Oculus Rift by the social network Facebook in early 2014 is indicative of social media's interest in providing a fully immersive, inhabitable virtual space with naturalistic avatar interactions [2]. Planning the design of your game-based virtual environment for real-time social interaction will take on even more meaning and prominence in your workflow as we progress toward the full integration of head-mounted display and facial/body tracking with the on-screen representation of avatars in the virtual space. If you are familiar with Facebook, you probably use the "friend" and "like" functions built into the pages and you know that these functions create ripples in a vast social net that interconnects with and influences others. To be a virtual world designer requires an understanding of how social media works and how to build support for it into your 3D designs. Here are the related concepts we will examine in this chapter, as we seek to deepen this understanding.

- Virtual social spaces, like the bubbles in foam, are full of existential spheres containing our subjective overview positioned in a social context.
- Designs that support the sensation of eye contact and an understanding of body language within the virtual environment are very powerful tools for the designer.
- Facial tracking, vocal shifting, and voice chat functions contribute significantly to the aspects of social interaction and the social media that develops from it in a virtual environment.
- A social network is significantly enhanced by utilizing ways to connect to the 3D-based VR worlds.
- Design that includes public, private, and exchange (marketplace) spaces will enhance the social interaction in a virtual environment.

In the project at the end of this chapter, you will start the development of a brand or graphic identity for your game-based sim and the ongoing process of collecting and curating visual content relating to it. As you finish the other projects within the chapters of this book, you will accumulate more related images and content necessary for this. Keep track of your progress by returning to this project and finishing each step as you develop your game-based sim.

5.1.1 Going beyond the Rhizome, the Vizome, and into Spheres and Foam

In Chapter 3, Section 3.2 of this book, we discussed *A Thousand Plateaus: Capitalism and Schizophrenia*. This book by Gilles Deleuze and Félix Guattari has laid a significant foundation for building an understanding, a holistic perception of interconnections and the networks of our modern media and communication. We co-opted the term *rhizome* to create *vizome*, a new word to help us define the "interconnectedness" experience in a virtual environment. Let's advance this conceptualization to another level by incorporating the thinking of Dr. Peter Sloterdijk, a German philosopher and cultural theorist who also works as a media correspondent. Sloterdijk has written a trilogy called *Spheres*, which explores the philosophy of existence in our hyperconnected and wickedly complex 21st century world. Some of the major ideas he has written about are seeping into the definitions and understanding about our "place" and relationship to the virtual world. Think visually for a moment. If you have ever looked at a virtual environment in wireframe, you will see that the sky is created on a huge sphere that surrounds the scene. Each of us is enclosed in that sphere, and within that are other spheres or containers that provide us with experiences we perceive visually and aurally in the virtual environment. These containers may be your virtual home in a sky box, or the boundaries of your virtual land region. It is important to realize that what we are experiencing is not completely defined by the math that creates the geometrical structures in the virtual environment or by the range of our visual field. What we are experiencing is also defined by our identities, our existence in the space that we create with our mind's eye. In the short video "Spheres – Existential Space," Sloterdijk says:

> The term "sphere," from the late 18th century on became a common European concept. I think that observation holds true. When authors like Goethe and Shiller or Kant and Hegel use the term "sphere," they usually don't think about the fact that during two thousand years of philosophical history it represented a very sophisticated cosmological idea according to which the Earth was the center of a system of seven or ten or twelve spheres, one on top of the other, generally made up of ether, that reach ever higher upwards, until the highest sphere of all, the crystal heaven or Empyrean is reached, where you're already quite close to God. This traditional cosmological and spatial-theoretical concept got humanized from the late 18th century on. I think that observation holds true. It was the work of linguistic history. All the modern spherologist does is repeat this shift in theory. The sphere as we talk about it today is an existential space. It is a space as physicists can't talk about it, as mathematicians or geometricians can't talk about it, but also as architects normally can't talk about it, nor registrars or clerks at the land registry office. Because these all work with pictures that describe an impersonal, abstract, geometrical or mathematical space. A space that has no "I." The moment I or an "I" enters such a space, "space" turns into a "sphere." [3]

In that short commentary, Sloterdijk gets to the subjective truth; we experience and create the space or sphere in our minds simultaneously. So, how can a designer leverage the concept of Sloterdijk's spheres for virtual environments? Here are some questions for you to ponder:

- When you design a virtual environment, how can you work with the "mind's eye" of the visitor?
- What elements in your design will evoke a sense of deep involvement and identification with the visitor?
- What interactive scripting and arrangement of structures in a virtual environment supports the formation of a sphere containing the "I," or a sensation of existence in the virtual environment?

Now, let us take the next step and relate the sphere concept to social interactions and social structures like cities and online communities. Think of blowing bubbles when you were a child. Do you remember how sometimes the bubbles would multiply and create a foam-like mass? In *Sphären III – Schäume* (*Sphere III: Foams*) Sloterdijk describes the structure of our society like this: "The co-isolated foam of a society conditioned to individualism is not simply an agglomeration of neighboring (partition-sharing) inert massive bodies, but rather multiplicities of loosely touching cells of life-worlds, each of which, by reason of its own inner expanse, deserves the status of a separate universe" [4]. We are, each one of us, a separate universe, sharing touch points with many other individuals via social media, virtual worlds, or the real-world architecture we live in. We are starting to see virtual representations of this idea in the architectural or system structures of the Metaverse. The OpenSim grid (http://www.osgrid.org) is like a foam structure, comprised of many privately owned servers linked together, each one holding its owner's virtual world. High Fidelity (https://highfidelity.io), a virtual world architecture being developed by Philip Rosedale, Ryan Karpf, Chris Collins, and others, embodies the social foam concept on several levels. They are leveraging social media to involve part-time developers, who bid for jobs on the work list, and open source development collaborators, who work alongside the full-time developers employed by High Fidelity. This allows High Fidelity to tap a multiplicity of creative sources, and keep the contributions organized and flowing. Social collaboration is inherent in the overall structure of its system architecture too. High Fidelity plans to create a system architecture that will let it scale to its audience size and the content it needs by allowing many people to set up virtual servers and interconnect them into a globally based structure. As you can see from the diagram in Figure 5.1, High Fidelity is creating a system structure comprised of five major groups: (1) global services, (2) internetworked virtual worlds, (3) clients, (4) contributed devices, and (5) agents and objects. This diagram was created by Greyson Stebbins, and used with permission from High Fidelity.

5.2 DESIGNING FOR THE "I," "YOU," AND "US" IN A VIRTUAL ENVIRONMENT

Let us suppose you log into a busy grid like Second Life (SL), and for one reason or another you get relocated to a "Safe Hub" or "Info Hub" instead of your home position. Suddenly you are surrounded by a crowd of strange avatars of all sorts and sizes. Find a seat, and take a few minutes to observe their behavior. Notice how groups who are chatting with one another in public or ("nearby chat") will often form a circle, just wide enough for their cameras to see a full-body image of each other's avatars. Typically, there is a fair amount of posturing and status challenges being issued to the group in nearby chat such as "I αм Δ∂σℓρн Ħιτℓєя!" and "elementary versus rudimentary, I'm more British than you know." The avatars are almost in continuous movement, driven by their looping animations. Sometimes racist and sexual comments are made in voice chat to shock and provoke those within hearing range. Some people remain mute, others chat, and some adjust their avatar's attachments or clothing. It is like a combination of an agora, a public bath, and a school yard.

5.2.1 Supporting Eye Contact and Body Language

While you may not be interested in hearing or seeing what everyone at the Safe Hub has to say, you will notice how these patterns of avatar positioning and chat repeat no matter where you are visiting in a virtual world. If you are going to use a virtual environment for something like a business meeting, brainstorming session, or a presentation, you will find that supporting visual and aural communication is crucial to the design of a virtual environment. Albert Mehrabian, a pioneer researcher of body language in the 1950s, found that the total impact of a message is about 7 percent verbal (words only), 38 percent vocal (including tone of voice, inflection, and other sounds), and 55 percent nonverbal [5]. By this observation (assuming you are not using any body, hand, or face tracking), we are only transmitting 45 percent of our meaning to the group participants in a virtual environment as it is configured right now. As a designer you should be aware of these limitations and what new devices will be added to move past them as virtual environments join large social media sites like Facebook. Right now, as this book is being written, we are on the threshold of another wave of technical advancements that will have direct impact on how you will design a

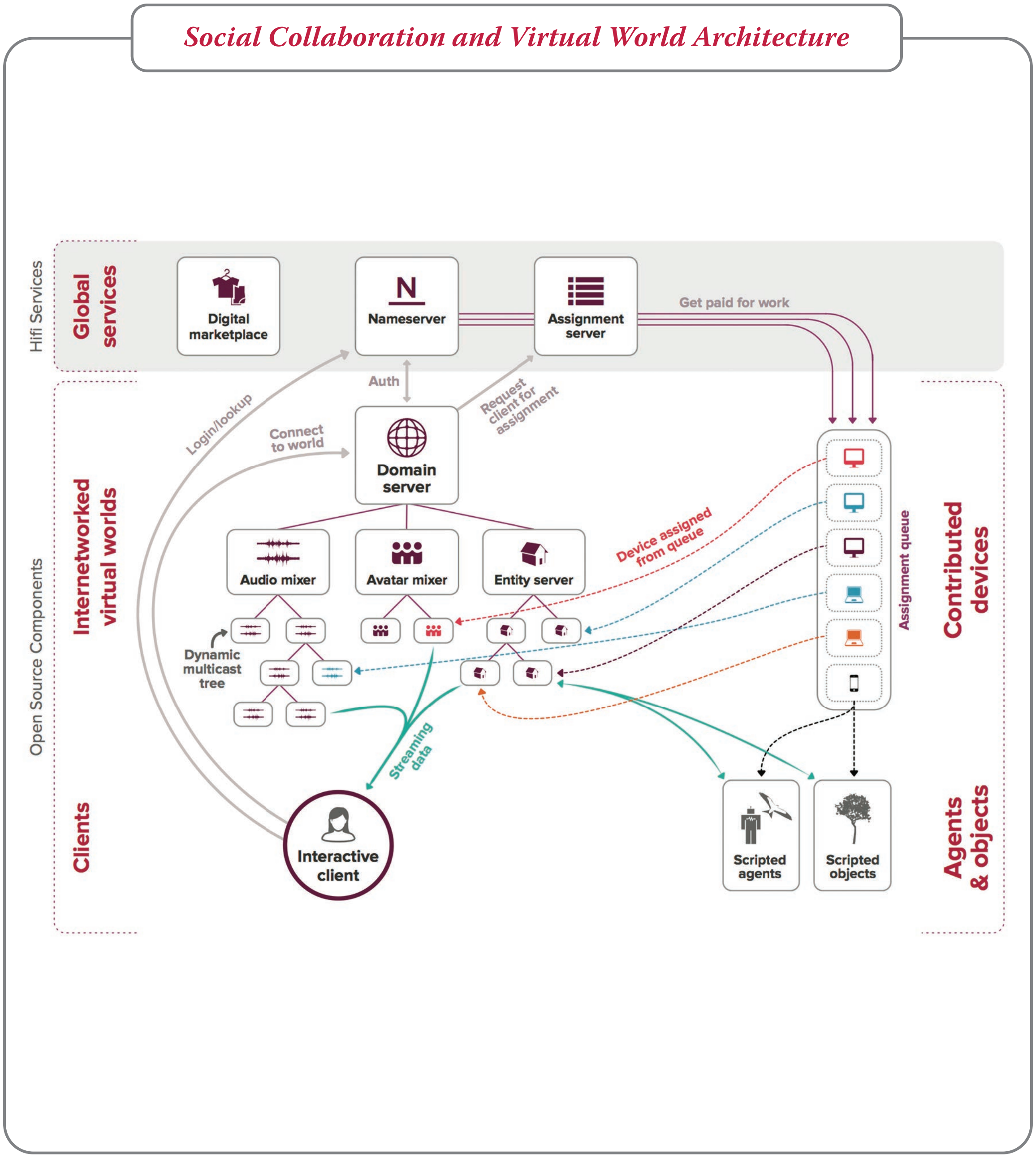


FIGURE 5.1 Diagram illustrating the interconnected groups that comprise the High Fidelity system architecture.

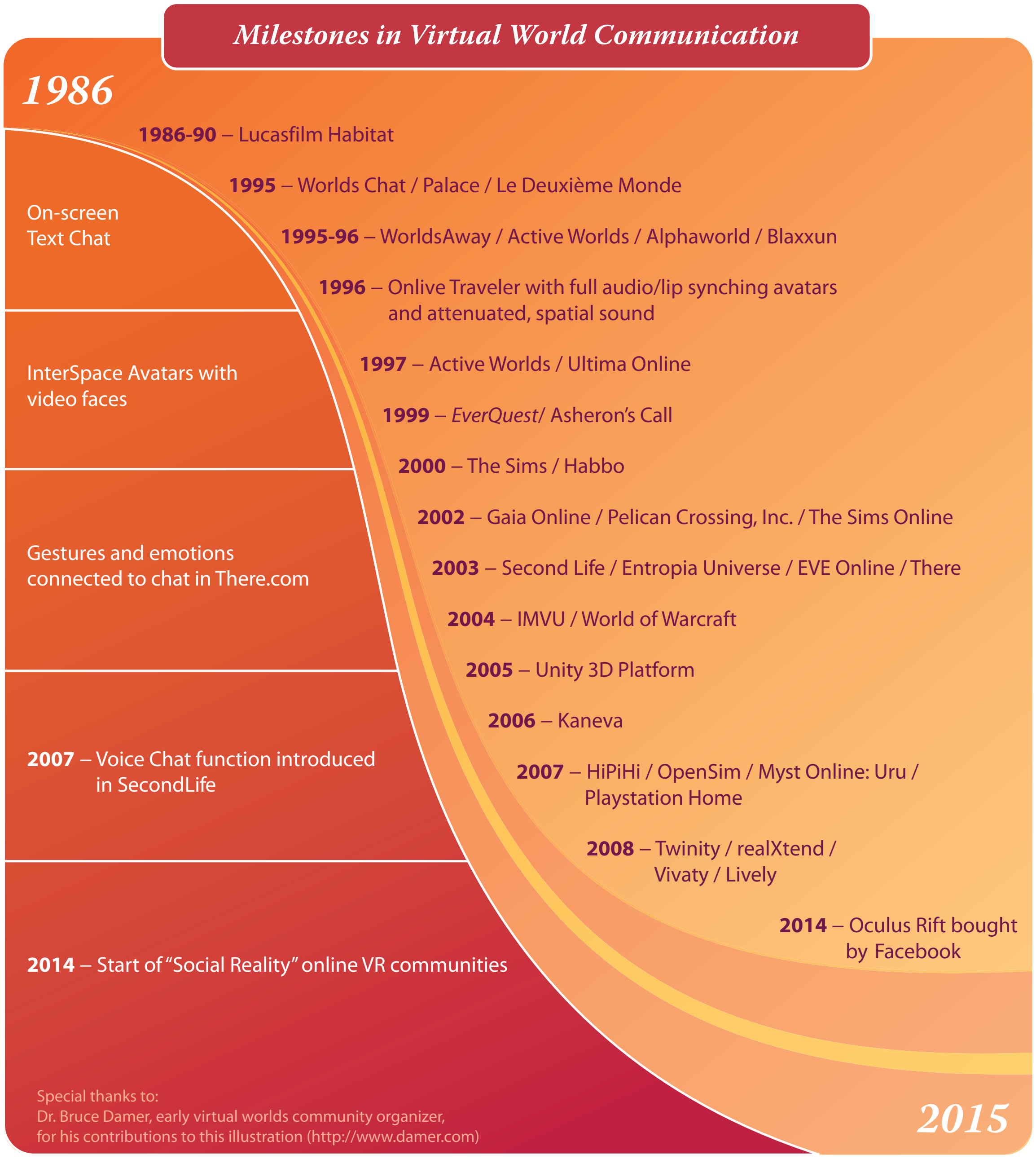

Milestones in Virtual World Communication
1986
1986-90 – Lucasfilm Habitat
1995 – Worlds Chat / Palace / Le Deuxième Monde
1995-96 – WorldsAway / Active Worlds / Alphaworld / Blaxxun
1996 – Onlive Traveler with full audio/lip synching avatars and attenuated, spatial sound
1997 – Active Worlds / Ultima Online
1999 – EverQuest/ Asheron's Call
2000 – The Sims / Habbo
2002 – Gaia Online / Pelican Crossing, Inc. / The Sims Online
2003 – Second Life / Entropia Universe / EVE Online / There
2004 – IMVU / World of Warcraft
2005 – Unity 3D Platform
2006 – Kaneva
2007 – HiPiHi / OpenSim / Myst Online: Uru / Playstation Home
2008 – Twinity / realXtend / Vivaty / Lively
2014 – Oculus Rift bought by Facebook
On-screen Text Chat
InterSpace Avatars with video faces
Gestures and emotions connected to chat in There.com
2007 – Voice Chat function introduced in SecondLife
2014 – Start of "Social Reality" online VR communities
Special thanks to:
Dr. Bruce Damer, early virtual worlds community organizer,
for his contributions to this illustration (http://www.damer.com)
2015

FIGURE 5.2 Timeline of the development and implementation of communication forms used in virtual world environments.

virtual environment to enhance the communication in it. In Figure 5.2 is a timeline illustration about how communication in virtual environments has evolved since the year 1990. (Many thanks to Bruce Damer for his insight and assistance with this diagram.) It is highly likely that you will soon have to consider new things such as the acoustics of a virtual space and hyperrealistic avatars that are animated in real time by face and body capture data.

Just as in a real-world situation, there is the *individual* we will call "I," the *other* we will refer to as "You," and the *social group* called "Us" to be considered in your design planning. All of these categories will be in the "social foam" as our individual worlds touch and communicate. Our virtual bodies, our avatars, may allow us to interact with other people globally, but they still limit us with their current lack of haptic and other sensory feedback. Your virtual environments should be designed to accommodate and facilitate interaction in spite of these barriers. Here is a checklist of what to consider as you design the space to support eye contact and body language in a virtual environment:

- The design of the space is clearly laid out and encourages grouping through the arrangement of seating or open areas.
- A moderately sized group of people (6–10 avatars) should have the space to get close enough to engage in vocal and local text chat when they want to.
- The avatar's voice can be clearly heard and its text chat can be clearly seen.

5.3 THE FACE, THE VOICE, AND SOCIAL INTERACTION

From very early childhood, human beings look at and interpret the meaning of human facial expressions. Every day, we use our observations of facial movement and tone of voice to guide us in our social interactions with other people [6]. Our brains utilize several information-processing areas to build a composite understanding of the individual features, the arrangement and movement of these facial features, and the emotional meaning of this facial composition. At this time, in late 2014, the most commonly available way to view a person's actual facial expressions and body language over distance is with videoconferencing or chat. People use online services like Skype or the embedded video chat channels in social media like Facebook or Google Plus, but not very comfortably or intuitively. Would we be more comfortable and able to communicate better if we do it through avatars that track our facial expressions and body language? That remains to be seen and demonstrated with social media sites that will use face tracking software on avatars in their video chat.

5.3.1 Design and Facial Tracking

In 2012, Sony Online Entertainment introduced the "SOEmote" module (https://www.soe.com/soemote) to its *EverQuest* II game environment. This module set up your webcam for facial tracking and mapped your expressions onto your *EverQuest* II avatar. Although it did not blossom into widespread use, this early addition of face tracking to a game platform preceded the development of software like Face Rig (https://facerig.com/), which lets you map facial tracking to an avatar and show that animated image in real-time over video chat systems like Skype and Google Plus. Visage (http://www.visagetechnologies.com/) is the face tracking software that powers Face Rig, and it is also utilized in the Eyeware Try-on app (for iPad), an augmented reality app that lets you try on sunglasses virtually. Face Shift (http://www.faceshift.com/) is a face tracking program for software developers that can be used to record and apply animation to a computer generated character. All of these advancements in face tracking, real-time animation capture, and augmented reality are great new tools for the designer of virtual environments and the creator of content it contains. Here is a short list of ways you could use face capture in your designs.

- Animate your avatar in real-time.
- Record animation for non-player characters (NPCs) and animated objects in your environment.
- Create augmented reality apps for your virtual world.
- Develop assistive devices for universal access to your virtual environment.
- Scan for 3D printing and prototyping of character's facial expression.
- Generate an animated previsualization for your designs.

5.3.2 Design and Vocal Shifting

In *I Know That Voice* (http://iknowthatvoice.com/), a documentary about vocal actors for animation, there is a moment when Nancy Cartwright who does the voices of Bart Simpson, Ralph Wiggum, and Nelson Muntz on The *Simpsons* animated show (Fox Broadcasting Company) switches back and forth from one character to another. She demonstrates three things in a few moments of screen time: (1) her understanding of a characters vocal tempo, (2) the great versatility and range her vocal cords have, and (3) how she can bring this musical tempo sensibility together with her acting talent to make these memorable, even iconic characters for this long-running television show. This is live, unassisted vocal shifting, and it is an amazing thing to listen to. As a designer, doing this kind of research and observation builds a solid foundation for your understanding of how to create the right voice for your avatar character and will guide you when you utilize vocal shifting technologies available in Second Life or via your favorite voice chat provider to create a great sound for your avatars or non-player characters.

5.3.3 Designing with the Built-In Audible Communication Formats

Since the overall sound design of a virtual environment is the purview of designers, they would be well served to consider the built-in voice chat formats as part of that design scheme. If you are trying to create a sense of serene peacefulness across your virtual environment, you probably do not want to listen to other people's avatars engaged in voice chats. Like the annoying cell phone conversations in a restaurant or on a bus, these audible conversations in the nearby voice chat inadvertently include all of the people within a 60 to 110 meter range depending on the avatars "listen from" settings. Rather than post signage in multiple languages asking avatars to use instant messaging (IM) or group voice chat, you have the option of disabling voice chat on the whole region or to restrict it to the parcels you choose. Second Life uses Vivox (http://www.vivox.com/) for its voice streaming services, and the Second Life viewer controls the settings for configuration, display, and control of each sort of voice option. In OpenSim, there are more options, and utilization varies depending on your capacity to run a separate voice server. The Freeswitch Module (https://confluence.freeswitch.org/) and Mumble/Murmur (http://wiki.mumble.info/) are open source voice chat software used by a number of OpenSim-based grids in the Metaverse that choose to run their own voice servers. There is information about installing them in the OpenSim wiki (http://opensimulator.org/wiki/). For other voice chat users who do not mind running another program to power their verbal communication, Skype (http://www.skype.com/) is a common alternative, as well as TeamSpeak (http://www.teamspeak.com/). Finally, Vivox offers both free and paid service, so that OpenSim grid owners can access the same voice chat software that Second Life does and smooth the transition for avatars exploring the Metaverse together.

5.4 DESIGNING FOR SOCIAL MEDIA INTERACTION

Many kinds of designers work in the social media world. There are the user experience (UX) designers who concern themselves with how a product/website is experienced or "feels" to a new user, user interface (UI) designers who work with the usability and layout of each screen or scene in the product/website; visual designers (graphics designers) who create the beautiful images and graphics that go on the product/website; interaction (motion) designers who create the interactive animation or movement on the interface that responds to your touch; and product designers who work with the creative teams, conduct user research, and interface with all the other design categories that were just described [7]. There is no doubt that in the near future we will be adding more designer categories to this list. These will be designers who concern themselves with the UX or UI of 3D virtual space, who work with the textures/graphics, and who create interactive responses from content in the game-based virtual environment.

5.4.1 Three Design Challenges in Social Spaces

Oftentimes, virtual worlds have been described as 3D chatrooms. Observation of my avatar's behavior reveals that I create shifting circles of social contact in a virtual world. If I am busy, I will log into my home position, stand there, and chat via IM to my inworld friends, behavior that could be illustrated as a single dot (me) at the center of a circle. The circle around my avatar shifts as I reach out via IM to various points on the grid, and it divides into two or three separate circles as I receive incoming messages from other friends and carry on more than one conversation simultaneously. When I am in need of more visible social contact, I may teleport over to my favorite sandbox for a group chat in text or voice, which could be represented graphically by a cluster of dots (me and the people I am chatting with) in the center of a fixed circle. If I want some variety in my visual experience, I may head off for some "sim hopping" with my friends or alone. Teleporting across a virtual landscape while chatting to various contacts is not really something that can be experienced collectively with 2D social media. You can describe your location changes with words and pictures to your social contacts in 2D social media, but unless they get into the virtual world, they cannot join you. In this collective experience is where the design challenges of 3D social media exist. In the following short sections, we will highlight three of the most important kinds of virtual environments for successful social media interaction: (1) public spaces, (2) personal spaces, and (3) exchange (or marketplace) spaces.

5.4.2 Public Space Design

On March 25, 2014, Facebook purchased the Oculus Rift company for $2 billion. On that day, Mark Zuckerberg, CEO of Facebook, posted these comments on his Facebook page: "This is really a new communication platform. By feeling truly present, you can share unbounded spaces and experiences with the people in your life. Imagine sharing not just moments with your friends online, but entire experiences and adventures. … One day, we believe this kind of immersive, augmented reality will become a part of daily life for billions of people."

Is it possible for a social media website to actually become a virtual environment? Even if only half of Facebook's 900 million members [8] decided to make a 3D personal space or hangout from their page, you would end up with 450 million virtual environments ranging in scope from the simplest box to a complex, multipurpose "identity space" dedicated, as websites are, to the display of the owner's personality and interests. Inevitably, the need to create some kind of visual and audible infrastructure will arise. What kind of grid or connecting system that allows people to navigate easily to their destinations in this vast 3D social media construct will Facebook build? Try to imagine what this virtual global public space or virtual terminal will look like. If you have read *Snow Crash* by Neal Stephenson, you are familiar with his description of the Metaverse.

> Hiro is approaching the Street. It is the Broadway, the Champs Élysées of the Metaverse. It is the brilliantly lit boulevard that can be seen, miniaturized and backward, reflected in the lenses of his goggles. It does not really exist. But right now, millions of people are walking up and down it. … The sky and the ground are black, like a computer screen that hasn't had anything drawn into it yet; it is always nighttime in the Metaverse, and the Street is always garish and brilliant, like Las Vegas freed from constraints of physics and finance. But people in Hiro's neighborhood are very good programmers, so it's tasteful. The houses look like real houses. There are a couple of Frank Lloyd Wright reproductions and some fancy Victoriana. So it's always a shock to step out onto the Street, where everything seems to be a mile high. This is Downtown, the most heavily developed area. If you go a couple of hundred kilometers in either direction, the development will taper down to almost nothing, just a thin chain of streetlights casting white pools on the black velvet ground. But Downtown is a dozen Manhattans, embroidered with neon and stacked on top of each other. [9]

How would you design a massive public space in a virtual environment? Let's look at a great historical building for inspiration. The main concourse of Grand Central Terminal (known to most folks as Grand Central Station) on 42nd Street in New York City is monumental in scale. The barrel vault ceiling, painted with celestial decorations, covers a space that is 275 ft (84 m) long, 120 ft (37 m) wide, and 125 ft (38 m) high [10]. People enter from all sides and levels, often meeting in the center at the information booth, which is surmounted by a four-sided clock. Overall, the experience energizes the visitor with the visual patterns of people's movements, the activities along the galleries on both sides, and the hum of a thousand conversations. As you look at the top picture in Figure 5.3, you can see how the spatial volumes, entry points, and architectural details all work toward providing the visitors on the east and west balconies with a fascinating view.

It's easy to see how effective this kind of space would be for a public meeting space in the Metaverse. And if you look closely, you can see that people are also making it into their private spaces, by clustering into groups, heading into corners or isolating themselves along the walls. Now compare this to the bottom image in Figure 5.3, which shows the central hub of the London, England, UK, region in Second Life (www.virtuallondons.com). Notice the similarity in how the avatars and people in these two pictures group themselves, how they cluster for conversation and visual communication, even though in a virtual world they could just as easily chat via the IM system.

5.4.3 Personal Space Design

Personal space in the real world is created and influenced by a multitude of factors. We all have invisible concentric spheres around us, of varying sizes, that define the borders of our personal comfort zones, and the proximity we allow strangers, friends, and lovers to our bodies. We are also forced by our environments to bend and twist these comfort zones, to accommodate the necessity of riding public transportation together, attend a crowded party, or take an elevator. In our social media behavior we also define our personal space, with the choice of our friends or contacts, what they get to see on our home pages, and what we let them post there. This kind of personal spatial control manifests itself in virtual environments too. We can invite others to visit our sim or ban them. We can set up invitation lists or make the site available to the public. We can also create interactive devices that will only reveal information to a chosen few. It does not matter if the personal space is the size of Grand Central Station or a shipping container; what matters is how we control it, that is what makes it a personal space. As you can see in Figure 5.4, people use positioning and body language to create their private space inside of a real-world public space.

Social Grouping in Public Spaces

Grand Central Terminal, New York City, NY, USA
Latitude 40.7528, Longitude -73.9772 DDD

Hub and Park, London England, UK

http://maps.secondlife.com/secondlife/London%20England%20UK/234/14/21

Special thanks to Debbie Butler (Debs Regent in SL) for the images of London in Second Life (www.virtuallondons.com)

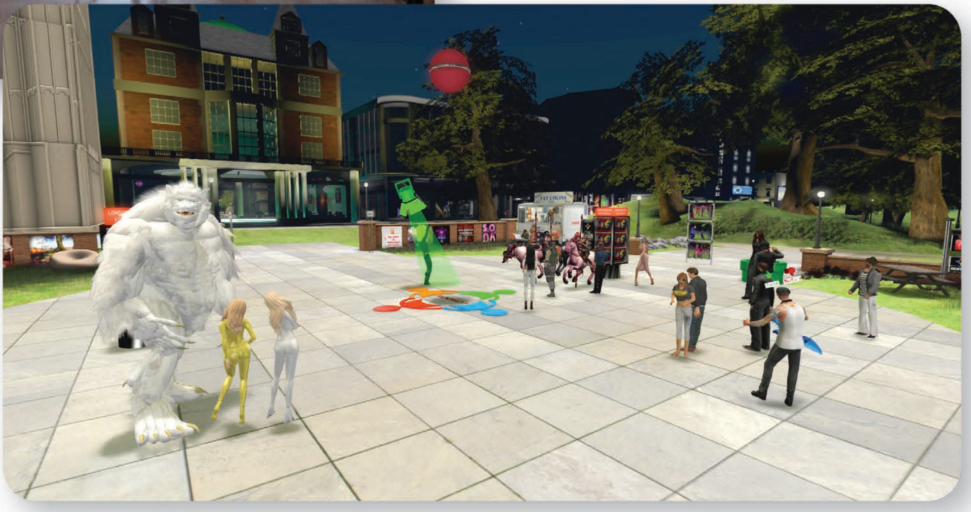
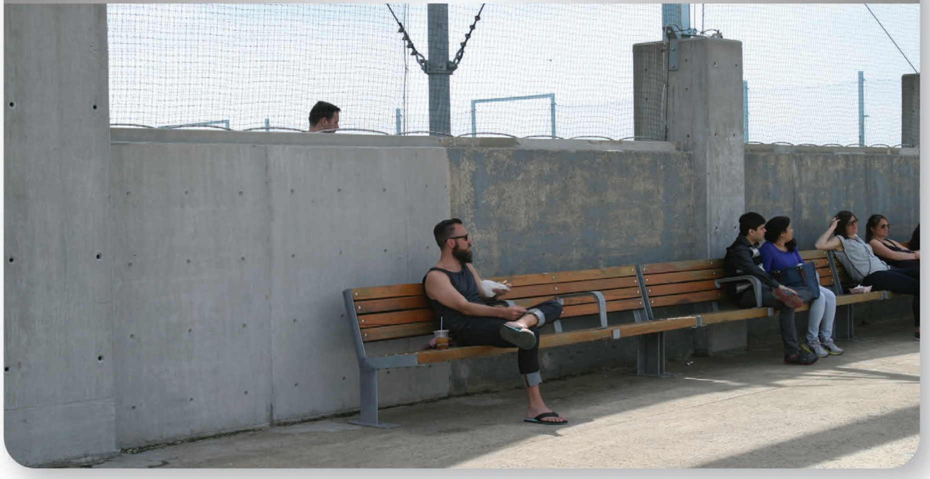


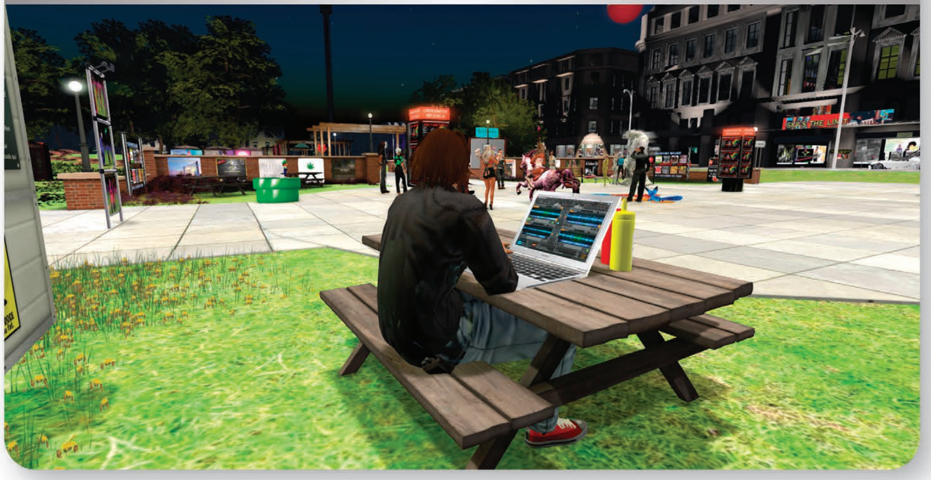
FIGURE 5.3 Social grouping compared: The real world space of Grand Central Terminal, New York (top) compared to virtual world space of Second Life/London England UK hub (bottom).

Finding Private Space

Finding private space in a public park in Brooklyn, New York



Finding private space in a virtual public park in Second Life



Special thanks to Debbie Butler (Debs Regent in SL) for the images of London in Second Life (www.virtuallondons.com)

FIGURE 5.4 Creating personal space: The real world (top) compared to a virtual world (bottom).

As a virtual environment designer, you will have to consider the following questions when you design a private space.

- What are the borders of the space?
- What kinds of privacies are offered? Visual? Audible?
- What kind of access should you design into the structures?

5.4.4 Exchange (Marketplace) Space Design

Exchange or marketplace spaces are often part of public spaces, but are set up with a different purpose and create a different kind of social interaction. Here in the marketplace things slow down, people take time to look around, and engage in conversations with the vendors. It is the place for the exchange of ideas, currency, goods, and services. Visibility is very important, as is a sense of location. All vendors want to make sure customers can find them, and all customers want to be able to relocate vendors when they return to the marketplace. What lifts this experience out of the real world and makes it part of the Metaverse is the capacity to post information about your purchases or services onto a social media site. How many times have we seen someone taking a picture of the meal they are about to eat, so they can post it on their favorite social site? We also share social information and our ratings about goods and services on the webpages we shop online from. How will this be translated into the 3D virtual environment? Will we be able to touch a virtual sofa in our favorite Metaverse shop and get an instant rating based on what the previous purchaser added into its database? Some customer feedback is posted on the pages of virtual marketplaces like the Second Life Marketplace (https://marketplace.secondlife.com/) or the Kitely Market (http://www.kitely.com/market), but it does not track consumer goods popularity in real time. Eventually, the sofa itself may tell us who is buying it and downgrade its rating as it starts to become less popular. This kind of data display could also be given out by the vendor's 3D space, so the customer knows what content is the most popular and how much of a trendsetter the vendor is. As you can see in Figure 5.5, marketplaces can exist within a large public space and often seek to do so. Visual indicators as to the popularity of products in the real world are things like customer lines and crowded interiors. These manifestations do not happen in a virtual environment as they are used now. Crowded virtual environments can get laggy and make shopping for virtual goods a frustrating experience as you wait for the server to respond to your actions. Until then, the 3D designer needs to find other ways to indicate a vendor's popularity to the passerby in the marketplace. As you can see in the lower image of Figure 5.5, the store owner has added a "window shopping" non-player character (the woman on the far right looking in the window of the shop) to give the illusion of some activity and interest around the store.

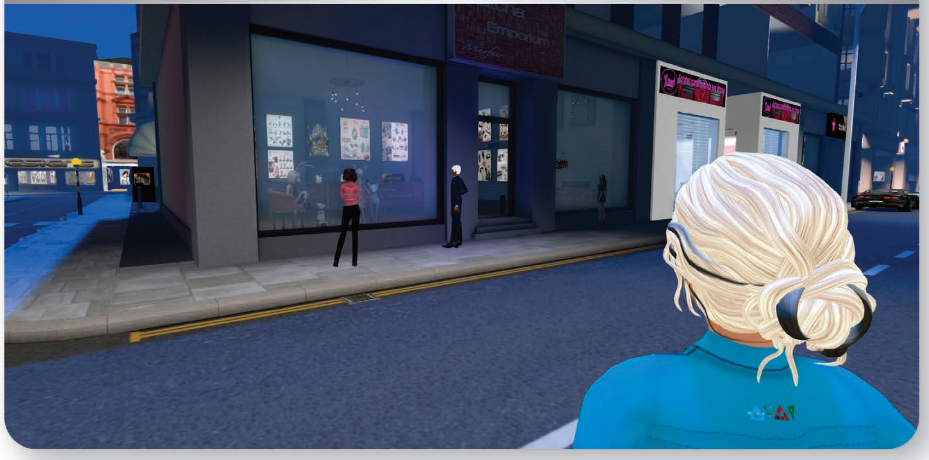
5.5 DESIGNING FOR SOCIAL COLLABORATION AND COMPETITION

"Immersive gaming will be the first, and Oculus already has big plans here that won't be changing and we hope to accelerate. The Rift is highly anticipated by the gaming community, and there's a lot of interest from developers in building for this platform. We're going to focus on helping Oculus build out their product and develop partnerships to support more games." These are some more of the comments by Mark Zuckerberg on the day Facebook bought Oculus Rift. Facebook, as we are all aware, has provided games such as *Farmville*, *Candy Crush Saga*, and *Words with Friends* for many years, so the social aspects augmented by the collaboration and competition of games is completely within their purview. In fact most of the online games provide for many forms of social contact and player rankings. As the designer of 3D

Signaling Popularity

Crowd lining up for barbecued ribs at the Smorgasburg Food Fair in Brooklyn, New York (www.smorgasburg.com)

PatriciaAnne Daviau and Art Blue doing some window shopping in Mayfair, London England, UK- in Second Life



Special thanks to Debbie Butler (Debs Regent in SL) for the images of London in Second Life (www.virtuallondons.com)

FIGURE 5.5 Vendor popularity compared: Real world food queue (top) and a virtual world shopping crowd (bottom).

virtual environments, especially the game-based sim, you should plan for the addition of these three elements into your design:

- Scorekeeping—A system or series of systems that records players' scores in your game
- Leaderboard/recognition signage—A display of players' rankings or accomplishments in a public venue
- Goals/levels—A series of goals or levels for players to attain while in the game

In Figure 5.6, you can see a screengrab from Second Life showing the scorekeeping system for an inworld archery game created by Norton Burns. As the arrows are shot, the scoreboard updates itself, and maintains a record of the competitor's scores.

Leaderboards are always popular; almost all players want to see their names on them. In Figure 5.7 is the "honors board" graphic/texture used for the Inland: Search for the Sy game-based sim. This was updated daily to reflect who had finished the entire game.

Evolution of the avatar was one of the goals to attain during the gameplay of Inland: Search for the Sy. As the player progressed through the puzzle levels, they evolved from a human form to an alien "Sy" form. As you can see from the screengrabs in Figure 5.8, their avatars became insubstantial and alien once they solved the second level puzzle of the game. Once they had become Sy aliens, they could go onto the third level, and if they solved that puzzle, they were transported to "Sy Space," the alien grid made from fractals, where they could hunt for their final prize—the Sy Finder Trophy.

5.6 PROJECT: BUILDING A COLLECTION OF RELATED CONTENT FOR YOUR GAME-BASED SIM

OK, let us move onto this chapter's project about how to build a collection of related content for your game-based sim. One of the key concepts from my first book *Virtual World Design* (Chapter 3, pages 21–37) was how to organize your workflow so that all content created for a project could be shared in an organized way by the creation team. Much of what you create on projects like these—the images you make in a 2D graphics program, the 3D content you build, the sounds and video you record—can all be used in multiple ways. In Appendix A is one example of how the content for a game can be defined and categorized by specific prefixes so that it will self-organize in files of everyone's computer or avatar inventory. As you can see there (Section A6.5), specific prefixes are used, such as BLDG, to indicate what sort of category this content should go into.

5.6.1 Creating "Content in Common"

In Table 5.1 is an overview of the "content in common" that may occur in your game-based sim project. Use this table to identify the content for the game-based sim project in Appendix A; find what will be shared across multiple platforms and make a list for each category. Note: If you have been developing your own game-based sim concept, then utilize that document.

5.6.2 Your Game-Based "Logo"

One of the most important elements in a game-based sim design and its related games is the logo. A logo is one of the most challenging things to design, because it has to distill and refine the underlying mythos of a project into a graphical image. In Table 5.2 is a checklist for your use in this part of the project. Take these questions and apply them to the information in Appendix A, or apply them to your own game-based sim project. If your class is creating a game-based sim and doing this project as classwork, try developing your logo concepts with teams.

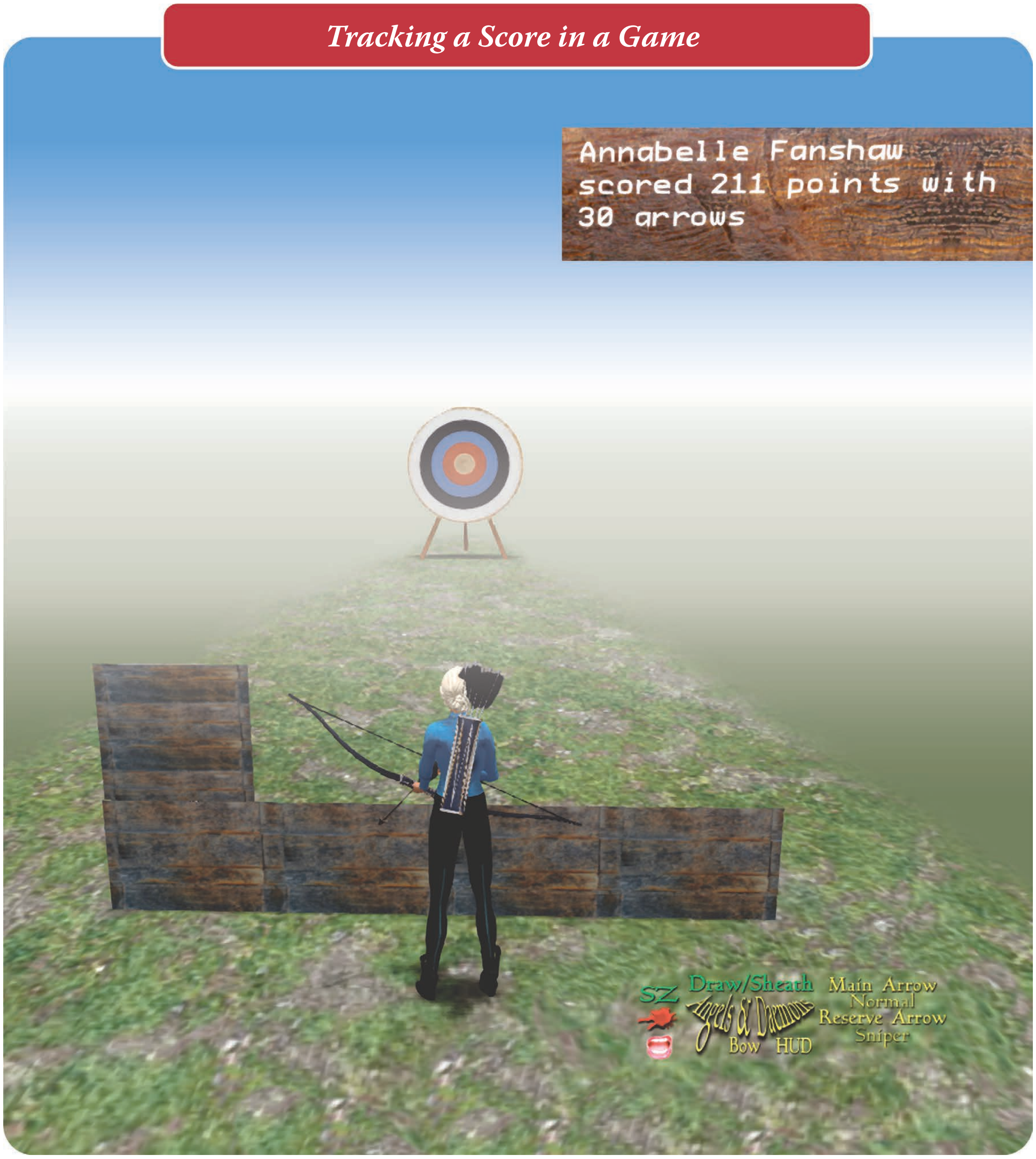
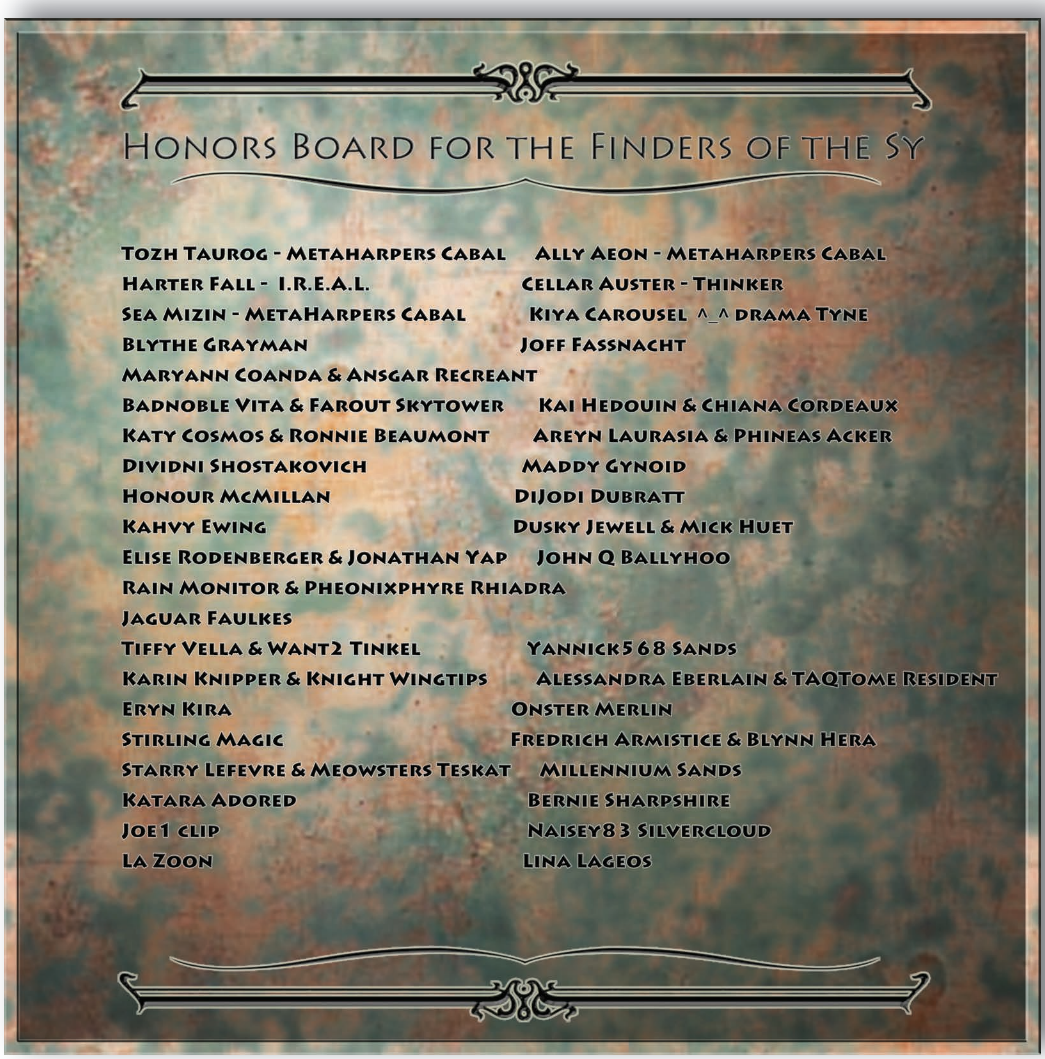

Tracking a Score in a Game
Annabelle Fanshaw scored 211 points with 30 arrows
Draw/Sheath
Angels & Daemons
Bow HUD
Main Arrow
Normal
Reserve Arrow
Sniper

FIGURE 5.6 Archery score keeping system devised by Norton Burns for an archery game in a virtual environment.

Indicating Rank in a Game-Based Sim



Honors Board for the Finders of the Sy

Tozh Taurog - MetaHarpers Cabal	Ally Aeon - MetaHarpers Cabal
Harter Fall - I.R.E.A.L.	Cellar Auster - Thinker
Sea Mizin - MetaHarpers Cabal	Kiya Carousel ^_^ drama Tyne
Blythe Grayman	Joff Fassnacht
Maryann Coanda & Ansgar Recreant	
Badnoble Vita & Farout Skytower	Kai Hedouin & Chiana Cordeaux
Katy Cosmos & Ronnie Beaumont	Areyn Laurasia & Phineas Acker
Dividni Shostakovich	Maddy Gynoid
Honour McMillan	DiJodi Dubratt
Kahvy Ewing	Dusky Jewell & Mick Huet
Elise Rodenberger & Jonathan Yap	John Q Ballyhoo
Rain Monitor & Pheonixphyre Rhiadra	
Jaguar Faulkes	
Tiffy Vella & Want2 Tinkel	Yannick568 Sands
Karin Knipper & Knight Wingtips	Alessandra Eberlain & TAQTome Resident
Eryn Kira	Onster Merlin
Stirling Magic	Fredrich Armistice & Blynn Hera
Starry Lefevre & Meowsters Teskat	Millennium Sands
Katara Adored	Bernie Sharpshire
Joe1 clip	Naisey83 Silvercloud
La Zoon	Lina Lageos

FIGURE 5.7 Leaderboard graphic/texture used for Inland: Search for the Sy.

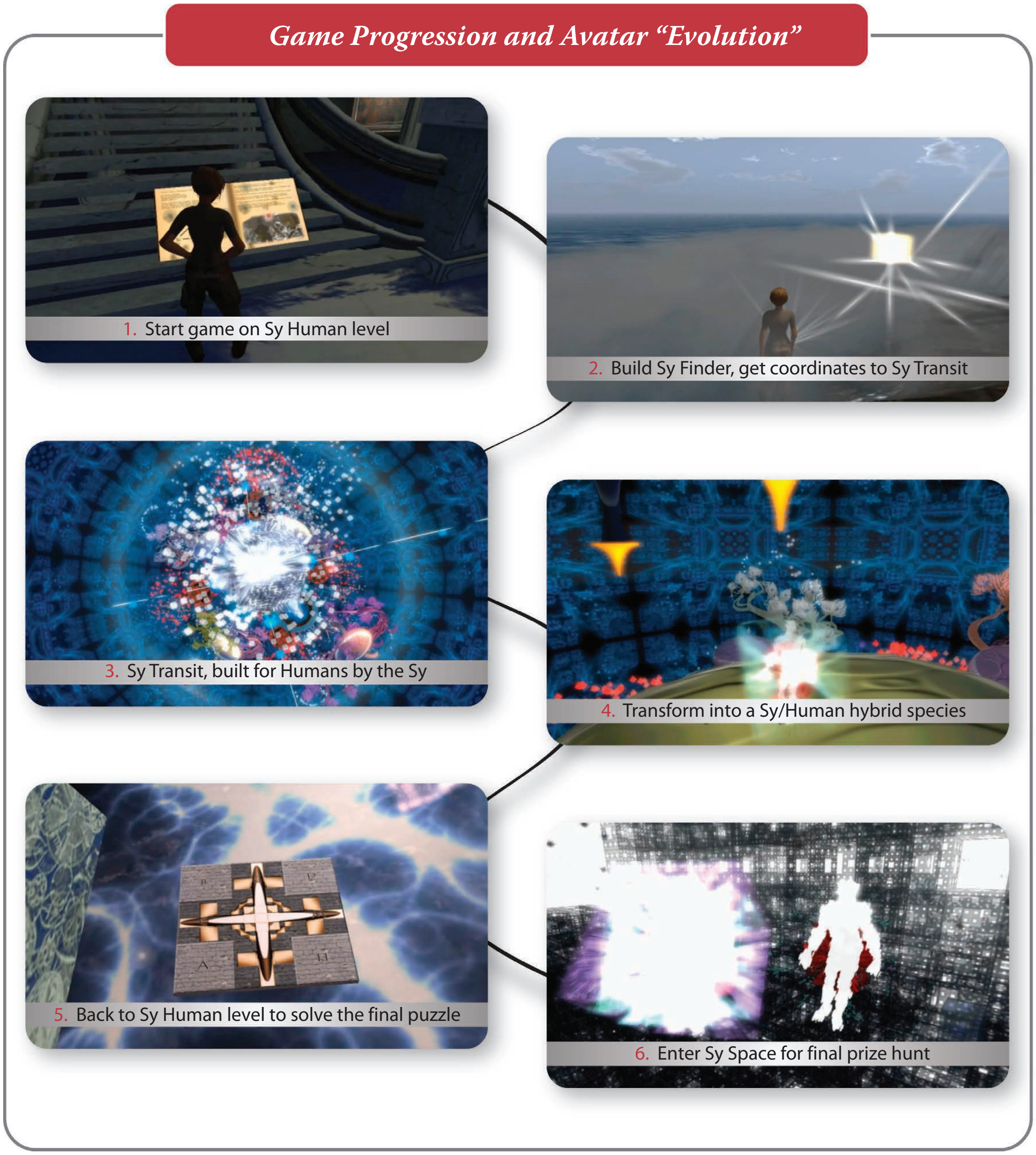

Game Progression and Avatar "Evolution"
1. Start game on Sy Human level
2. Build Sy Finder, get coordinates to Sy Transit
3. Sy Transit, built for Humans by the Sy
4. Transform into a Sy/Human hybrid species
5. Back to Sy Human level to solve the final puzzle
6. Enter Sy Space for final prize hunt

FIGURE 5.8 Evolution of human to "Sy" alien life form during Inland: Search for the Sy game play.

TABLE 5.1
Content in Common and Sources

Type of Content	Source of Content	Purpose of Content
Elemental content such as graphics for virtual environment	Created from user-owned images and graphics software	Used to create the environment both in the virtual world and in the website space
Signage content such as directional graphics and logos	Created from user-owned images and graphics software	Used to create movement and give information in the virtual world and in the website space
3D content such as models, architecture, and interactive elements	Created from user-owned graphics and 3D modeling software	Used to create the game-based sim as well as elements for 3D printing and augmented reality versions of the game
Sound-based content such as background music, interactive sounds, and ambiance tracks	Created from user-owned sounds and sound editing software	Used to a source of interactive guidance and energetic ambiance in the virtual space and on the website
Coded content such as scripts and applications	Created by user or code writer	Used as a source of interactivity within the game-based sim and related platforms such as mobile devices

TABLE 5.2
Checklist for Logo Development

Questions to Ask about the Logo	Related Elements	Notes for Action Items
Who is this logo for?	Elements that reflect someone's personal taste	
Is the logo for a specific game?	Name of the game spelled out in appropriate text	
Is the logo for a specific event on a game-based sim?	Name of the sim spelled out in appropriate text	
What is the overall tone that needs to be supported in the logo (i.e., medieval madness, pastoral pleasantness, or super sci-fi)?	Additional side images or background that illustrate the overall tone	
What is the overall pallet range and does it support the tone?	Color samples that harmonize or underscore the overall tone and range of experience	
What are the shapes and typography that are seen to support the tone and relate to the virtual environment?	The right font, modified as necessary	
How many forms of reproduction will be used?	Square shape will fit anywhere, rectangular may be best second choice for all kinds of reproduction, such as web, video, print, augmented reality, and 3D printing	

5.6.3 Setting Up Your Channels

Mastery of social media and the venues it provides are essential for the promotion of your game-based sim. Through 2D and 3D channels you can engage the audience and bring more visitors to your virtual environment. In the next sections, you will review four types of channels that can be accessed for the promotion of your game-based sim, and set up ways to promote your content.

Skip ahead and do the projects in Chapters 9 to 12 first if you want to utilize the content and projects in this book for your channels.

5.6.3.1 Making a Channel for 2D Still Media

Now that you have a virtual environment to make images from, you should set up accounts for your game-based sim with 2D social media sites like Flickr, Instagram, and Pinterest. If you want to reach a broad age range, then set up Facebook and LinkedIn pages for your virtual environment and seed it with the images you posted on the other social media sites. Make sure you have captioned the image, giving location and date it was taken. In Figure 5.9, is a screengrab of my Flickr page gallery showing images from this project.

5.6.3.2 Making a Channel for Video Streaming Media

The next level of social promotion is streaming video, live or edited. Set up an account for your game-based sim on YouTube or Vimeo for your edited promotional videos. You can add a live streaming account on Livestream (http://original.livestream.com) if you have an event to cover. In Figure 5.10 is a screengrab showing the YouTube channel for Alchemy Sims, the go-to place for all the promotional videos we have created.

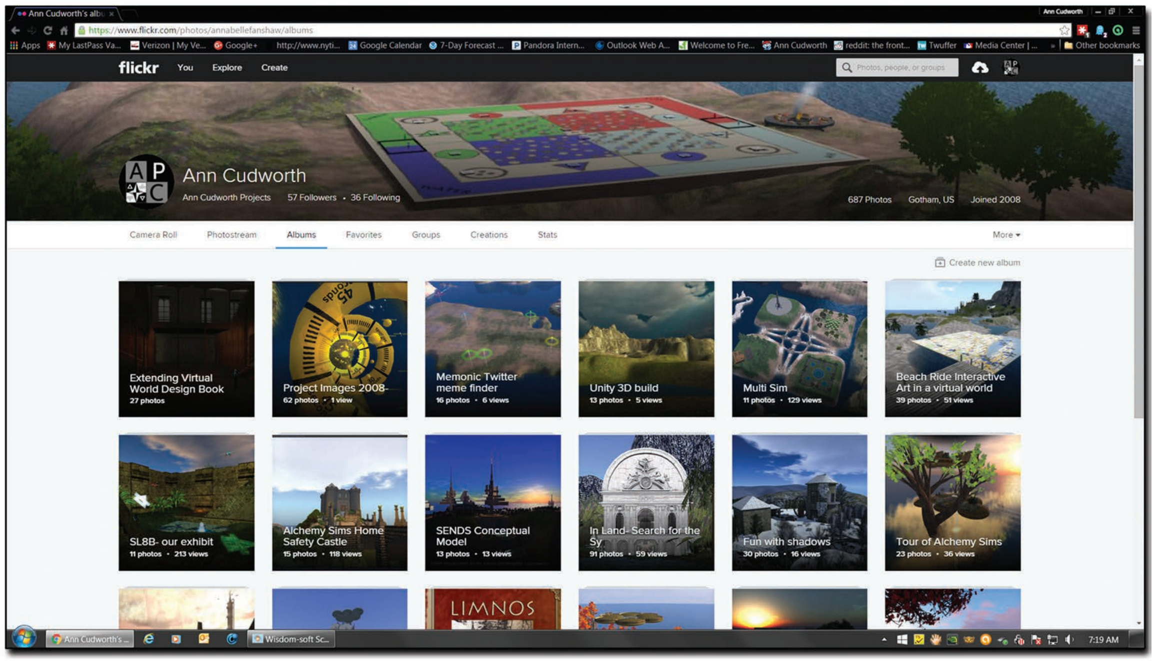


FIGURE 5.9 Image of a 2D channel. (From Ann Cudworth Projects/Alchemy Sims, Flickr page.)

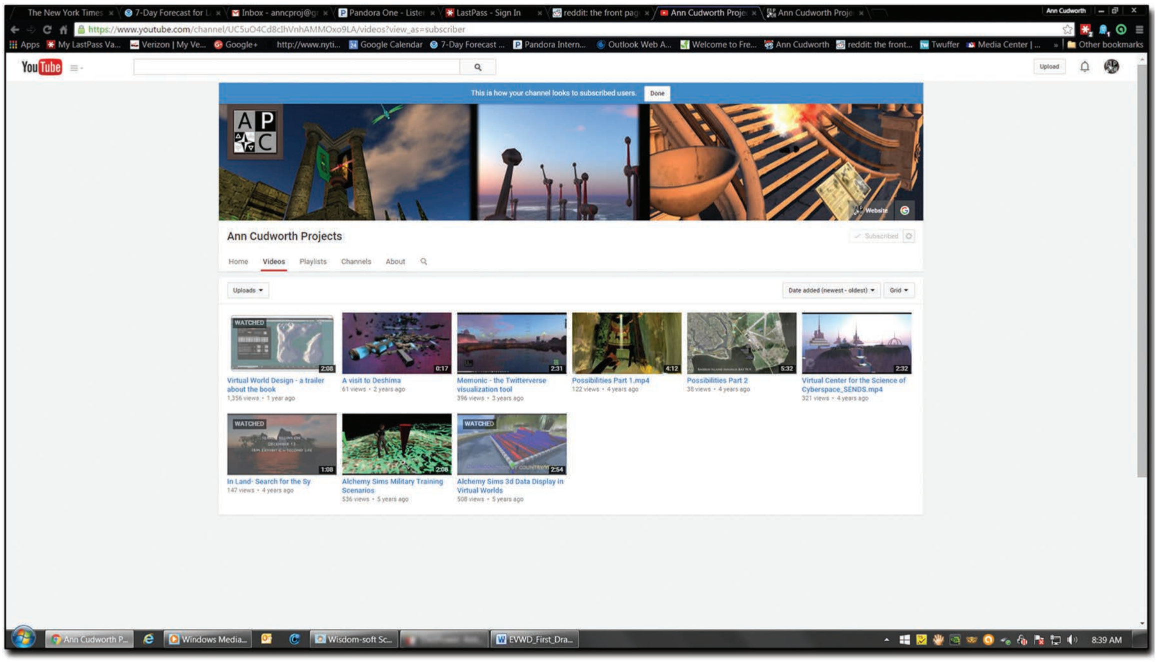


FIGURE 5.10 Image of video channel. (From Ann Cudworth Projects/Alchemy Sims, YouTube.)

5.6.3.3 Making a Channel for 3D Content Exchange

If you want to enter the real 3D world with your virtual objects, the possibilities are endless. Try uploading and printing a 3D model file from an online site like Shapeways or Sculpteo. Perhaps you have your own printer and can generate your own 3D models. There are many ways that this real 3D content can be used. You can make character player pieces, or give special content models away as prizes or awards.

5.7 CHAPTER SUMMARY AND FINAL THOUGHTS

John Carmack, chief technology officer (CTO) of Oculus and generally considered to be an apex developer of game engines, made some comments recently that conveyed his understanding of the importance of social interaction in the virtual game space. He said, “Getting that multiuser experience is our most important user-visible feature. It’s surprising that it hasn’t been done so far. As the basics mature, I expect more of that to be going on. … When you see grandparents using iPads to look at pictures of their kids. It’s going to be the same with VR. Everyone in the world will find something to do with it” [11]. As you read this chapter, which skipped lightly over the massive amount of new technologies and systems that will be linked to virtual reality and social media in the coming years, it probably occurred to you that there is one concept that binds it all together. Essentially, no matter how many bells and whistles they hang on the user interface for a virtual reality experience inside of social media, all that we will care about at the end of the day is how well we can communicate with each other. All the face tracking and voice shifting, the design of personal and private spaces, and the understanding of how to work with the mind’s eye of the visitor must support that goal.

REFERENCES

1. Bruno Latour, lecture at Harvard University Graduate School of Design, February 17, 2009, immediately after the lecture by Peter Sloterdijk, accessed October 1, 2014, http://www.weskline.com/115-SPACE-HARVARD-GB.pdf.
2. Nick Wingfield and Vindu Goel, "Facebook in $2 Billion Deal for Virtual Reality Company," *New York Times*, March 25, 2014, accessed November 22, 2014, http://www.nytimes.com/2014/03/26/technology/facebook-to-buy-oculus-vr-maker-of-virtual-reality-headset.html.
3. Peter Sloterdijk, "Sloterdijk: Spheres – Existential Space," Vimeo video, accessed September 18, 2014, http://vimeo.com/album/211206/video/11554213.
4. Peter Sloterdijk, "Sloterdijk's Foam City," WordPress, posted March 24, 2012, accessed September 18, 2014, http://architecturecriticism.wordpress.com/2012/03/24/sloterdijks-foam-city/.
5. Allan and Barbara Pease, "First Chapter: The Definitive Book of Body Language; Understanding the Basics," *New York Times*, accessed September 23, 2014, http://www.nytimes.com/2006/09/24/books/chapters/0924-1st-peas.html.
6. *Wikipedia*, s.v. "Face Perception," accessed September 24, 2014, http://en.wikipedia.org/w/index.php?title=Face_perception&oldid=639475418.
7. Lo Min Ming, "UI, UX: Who Does What? A Designer's Guide to the Tech Industry," Fast Company, accessed September 29, 2014, http://www.fastcodesign.com/3032719/ui-ux-who-does-what-a-designers-guide-to-the-tech-industry.
8. *Wikipedia*, s.v. "Facebook," accessed September 25, 2014, http://en.wikipedia.org/w/index.php?title=Facebook&oldid=640232181.
9. Neal Stephenson, *Snow Crash* (Random House), Kindle edition, 24.
10. *Wikipedia*, s.v. "Grand Central Terminal," accessed September 29, 2014, http://en.wikipedia.org/w/index.php?title=Grand_Central_Terminal&oldid=639158577.
11. Emanuel Maiberg, "Oculus CEO Says the Heart of VR Will Always Be Gaming," *Gamespot*, September 27, 2014, accessed October 1, 2014, http://www.gamespot.com/articles/oculus-ceo-says-the-heart-of-vr-will-always-be-gam/1100-6422603/.

6 Developing the Designer–Scripter Collaboration

Every collaboration helps you grow. With Bowie, it's different every time. I know how to create settings, unusual aural environments. That inspires him. He's very quick.

Brian Eno

6.1 OVERVIEW OF DEVELOPING THE DESIGNER–SCRIPTER COLLABORATION

Any game-based virtual environment will contain a number of scripts written to provide player interactivity, vehicular mobility, scenic changes, and other dynamic elements. While there are some individuals who are both master designers and scripters, most people tend to focus their creative efforts in one area or the other. This division of labor creates the need for a common understanding of each other's skillsets and job duties. With this common understanding, forming an alliance to become a build team is very rewarding for both the designer and the scripter. Farther along in this chapter we will examine this relationship, learn how to build a good foundation in designer and scripter vocabularies and examine two case studies of scripted game-based environments created by designer Ann Cudworth (Annabelle Fanshaw in the Metaverse) and her scripter associate Vicki Brandenburg.

In this chapter we will work with the following concepts:

- The organizational structure of the team can be variable and scalable.
- Good creative teams understand each other's vocabularies.
- The designer and scripter must work together to create dynamically functioning game-based virtual environments. [1]

At the end of this chapter, you will play a game called "Script Roulette" with a build partner to develop core communication skills and enhance your creative thinking.

6.2 COLLABORATING GUIDELINES FOR WORKING TOGETHER ON AN ADVANCED DESIGN

Chapter 3 of *Virtual World Design* (published 2014 by CRC Press) illustrated and discussed the organization of small and large virtual design offices (Sections 3.3.4 and 3.3.5). Let us review them and relate it to the collaborative structure that develops between a virtual environment designer and the scripter they are working with. As you can see from Figures 6.1 and 6.2, virtual design offices can become a complex network of many groups who all contribute to their part of the project.

Let us focus on the special relationship between those who design and those who write scripts for virtual environments. Typically, people who create game-based virtual environments as a hobby tend to be multitaskers. It is not unusual to find a scripter creating 3D models and a designer writing LSL scripts. This is all well and good in a small project, but when things start to scale up, division of labor becomes necessary.

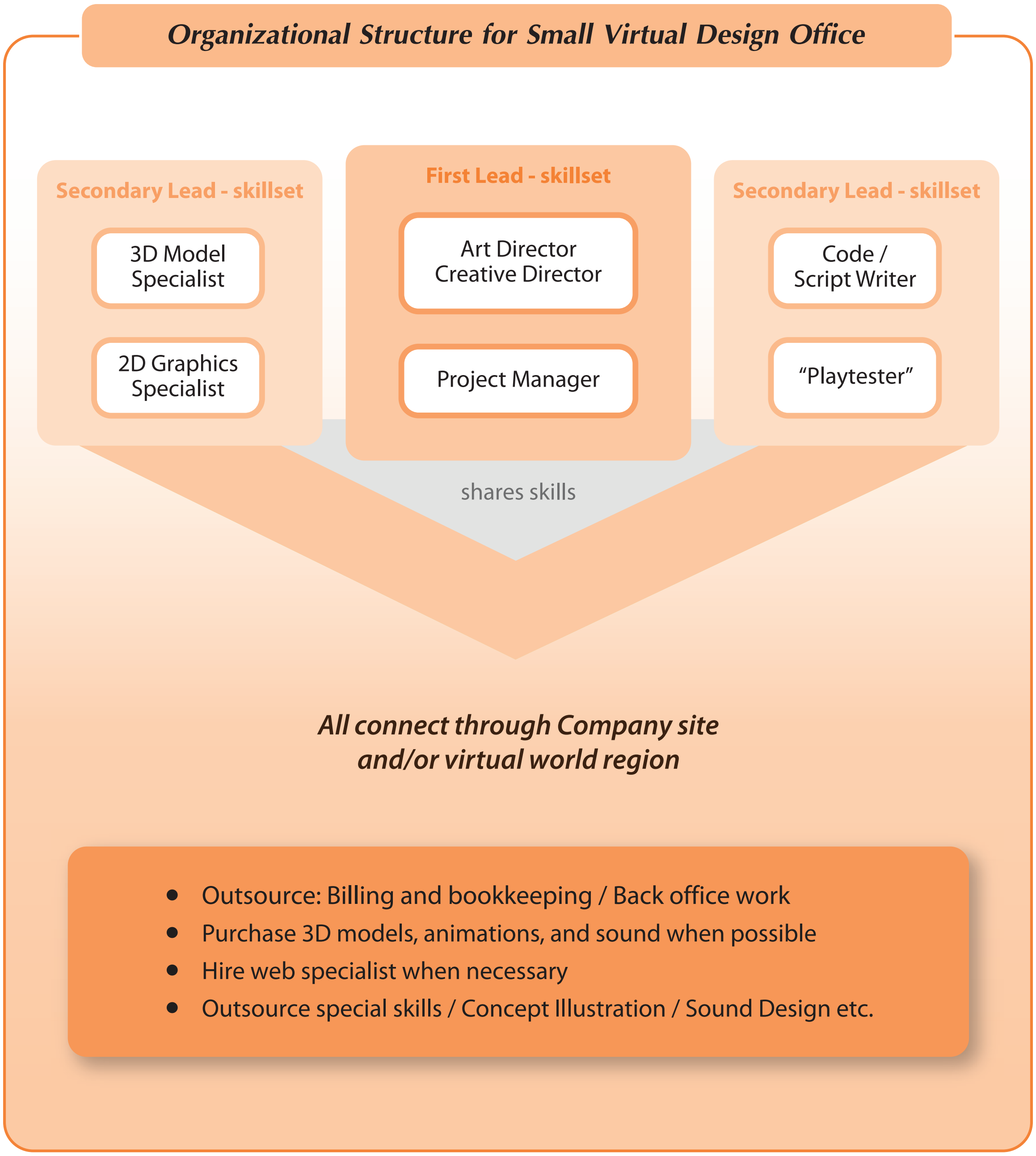

Organizational Structure for Small Virtual Design Office
Secondary Lead - skillset
3D Model Specialist
2D Graphics Specialist
First Lead - skillset
Art Director Creative Director
Project Manager
Secondary Lead - skillset
Code / Script Writer
"Playtester"
shares skills
All connect through Company site and/or virtual world region
• Outsource: Billing and bookkeeping / Back office work
• Purchase 3D models, animations, and sound when possible
• Hire web specialist when necessary
• Outsource special skills / Concept Illustration / Sound Design etc.

FIGURE 6.1 The organizational structure for a small virtual design office.

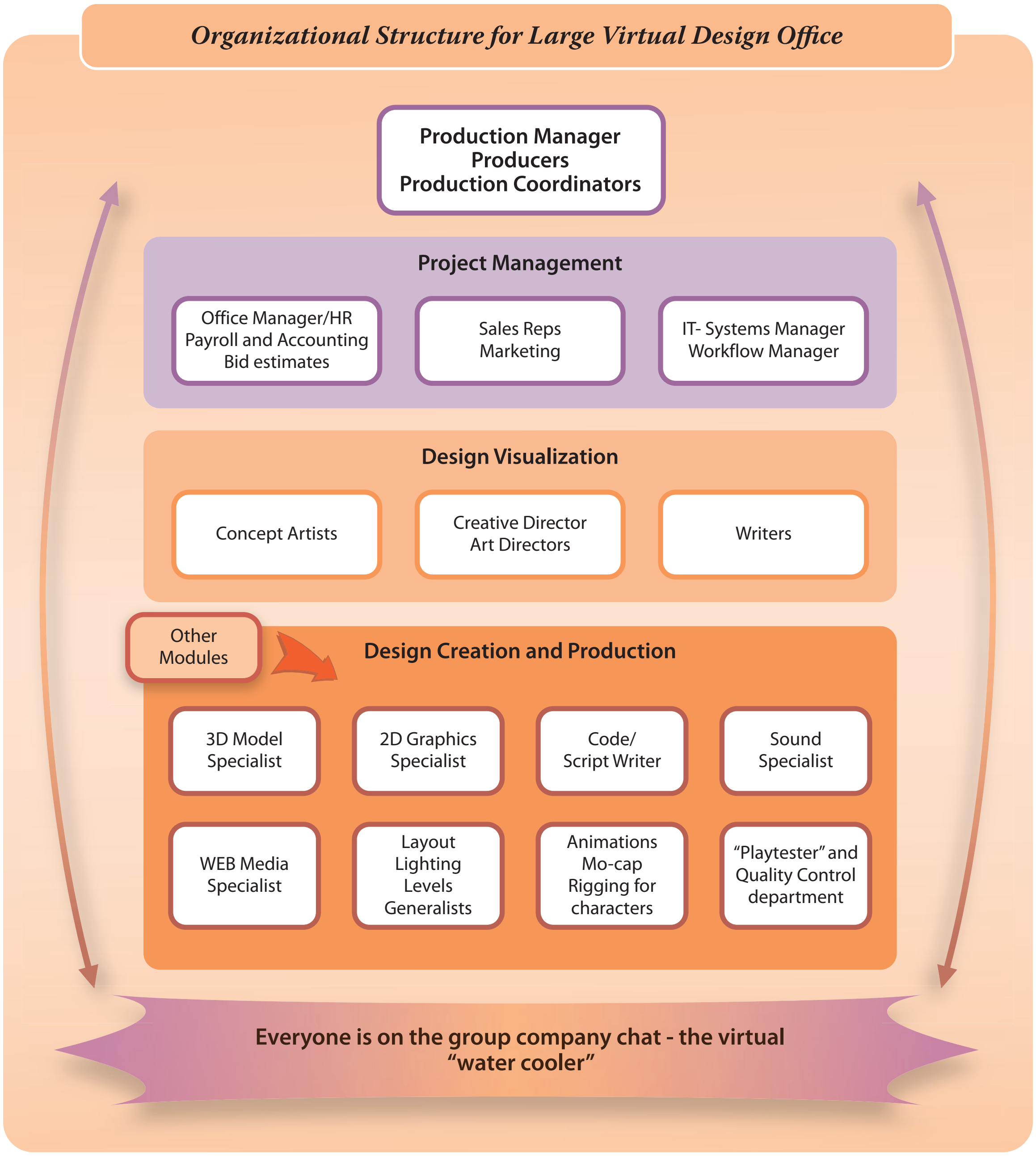


FIGURE 6.2 The organizational structure for a large virtual design office.

TABLE 6.1
Defining Designers' and Scripters' Tasks and the Creation of Collaborative Elements in a Game-Based Sim

Designers, Modelers, Illustrators, Graphic Artists	Collaborative Elements in a Game-Based Sim	Scripters, Coders, Programmers
Things they do:	Overall game flow and transitional aspects of the player experience within the environment*	**Things they do:**
Concept illustration*	Script-based environmental factors like weather; atmospheric displays; and terrain effects like volcanoes, earthquakes, floods and tsunamis	Experiential planning* and scripting
Terrain design	Script-based texture changing elements like animated limbs and wings, season-changing foliage, touch panels and buttons, and flashing signage	Interactivity planning* and scripting
Graphic/texture creation	Script-activated 3D models such as vehicles and avatar attachments (styling menus for hair, shoes, etc.)	Script development and testing
3D models	3D objects that are scripted to be components in the game, such as clue-giving elements, interactive characters, interactive vehicles, weapons, scoring systems, and reflexive building elements that respond to avatar presence during the game	System efficiency testing and balance
Avatar visual design	Also includes access "keys," prizes, and badges that will influence the gameplay	
Sound design		
Dialog and narrative*		

* These items may also be the purview of the storyteller or writer on the project.

Knowledge of what the other groups contribute to the project is essential, even if you are not doing that task. Furthermore, the project manager needs to know what a designer and a scripter do, and how to get them communicating effectively. In Table 6.1 is a partial list of the collaborative elements that scripters and designers might work on while building a game-based virtual world.

As you can see, there are a significant number of items in a game-based sim that involve the creative efforts of both parties, and these items usually have the most impact on the player's or visitor's experience. In fact, the seamless quality of a player's experience is so important that in 2013, Linden Labs (for Second Life) started a project to develop "Experience Tools" (also known as "Experience Keys") to facilitate smooth functioning of game-based elements inworld [2]. In the Second Life viewer there are screen menus that allow for the premium player to store "permission keys" for things like instantaneous teleport and weapons-based animations. With this system they only need to grant permission once when visiting the game-based sim, and then they are free to enjoy the game flow without any other permission pop-up menus on their screen. If you have tried to build or even visited a game-based sim before, you can see the advantage of this streamlining. This is a boon for designers and scripters alike [3].

6.2.1 Designer–Scripter Collaboration: Understanding Each Other's Job

Table 6.1 has provided you with an introduction (or perhaps an affirmation) as to the complexities of collaboration with a team on a game-based sim. In Figures 6.1 and 6.2 you saw the ways small and large virtual design offices may function. By now, you have probably surmised that it is crucial that you understand the scope of everyone's job, especially if you are in a project manager's role. Not only does this help you see where there may be bottlenecks in the creation flow, you can also explain to the client why some tasks take longer than expected. This is especially true in the designer–scripter collaborative process.

Here are some key factors to keep in mind while you are collaborating:

- The designer and scripter should share a common goal that is clear and simple. To do this may require that the project be broken into a series of smaller steps and displayed on a project timeline.
- Keep the lines of communication flowing between you with e-mail and frequent, brief face-to-face (or over Skype) meetings.
- The designer is responsible for a clear presentation of what the game-based sim and the elements will look like, and how they are expected to behave interactively.
- The scripter is responsible for a clear organized approach to the creation of all scripts necessary to enliven the elements in a game-based environment that satisfy the interactive needs of the project and conform to the server's capacity for that region.
- The scripter is also responsible for communicating to the designer or project manager that the scripting requirements are infeasible, of immense difficulty, or are otherwise likely to require a larger budget than allocated for time or money.

6.2.2 Learn the Key Vocabularies and Symbols for Each Other's Work

Every designer has experienced the moment when they present a set of drawings to a client and discovered that the only thing that the client can relate to is the sketch of the completed project. All the drafting and construction drawings are just lines on a page to the client, who has had no training in how to read the signs and symbols on a set of plans. Every scripter has a similar experience when the client (who could very well be a designer) does not have enough fundamental knowledge of scripting for an intelligent conversation about the parameters of the script they just commissioned. If you are a designer who has been working in the gaming world, you may already know about scripting; perhaps you use JavaScript for Unity or perhaps Lua for another game you have worked on [4]. In Table 6.2 is a short comparative list of the kinds of vocabularies that each position requires, so designers who do not program and scripters who do not design have a place to start understanding each other's approach. How many terms on both sides are you familiar with? If your answer is not too many, here are some ways to fill in the gaps in your knowledge.

6.2.2.1 Learning a Designer's Vocabulary

Designers base their visually descriptive vocabulary in the history of art and architecture. A fun way to start gathering your design vocabulary is through the fantastic museum apps and websites that have appeared over the last few years. For example, the Museum of Modern Art (MoMA) has a good learning-based site (http://www.moma.org/learn/moma_learning) that allows you to browse by theme and artist, and the Tate Galleries site offers a comprehensive glossary of art terms (http://www.tate.org.uk/learn/online-resources/glossary). Museums that house architectural and archeological treasures, like the British Museum (http://www.britishmuseum.org/) and Metropolitan Museum of Art (http://www.metmuseum.org/) are also recommended viewing, since many virtual environments a designer creates will be founded in that kind of visual research.

6.2.2.2 Learning a Scripter's Vocabulary for Linden Scripting Language (LSL)

There are many books for learning virtual-world-specific programming. *Scripting Your World* by Dana Moore, Michael Thome, and Dr. Karen Zita Haigh, is one of the classics and will provide you with a comprehensive foundation in scripting for Second Life and OpenSim. There are more books mentioned in the

TABLE 6.2
Comparing the Vocabularies of the Designer and Scripter

Designer's Vocabulary	Scripter's Vocabulary
3D compositional and perspective terms:	Basic terms in scripting:
Atmospheric perspective	Boolean
Bas relief	Event
Contrast and similarity	Function
Forced perspective	Operator
Six-point perspective	String
Visual hierarchy	Variable
Art and architectural style terms:	Computer architecture terms:
Neoclassical	System architecture
Postmodern	Virtual machine
Pre-Raphaelite	Syntax
Virtual world design terms:	Virtual world scripting terms and names:
Chim	Grey Goo
Coalesced Object	Indra
Fat Pack	Ogg Vorbis
Sculpty	Project Snowstorm

Source: "Second Life Glossary," *Second Life Wiki*, accessed February 6, 2016, http://wiki.secondlife.com/wiki/Second_Life_Glossary.

Bibliography at the end of this book. Essentially there are eight basic terms that a designer needs to know about in LSL scripting [5]. If you think of these eight terms as a team of superheroes, each with a specific power, it may be easier to remember them. Let us meet the team:

1. Functions—This LSL hero is identified by its red text in the Script Editors window and always starts with the "`ll`" symbol.
2. Events—This LSL hero is identified by its blue text in the Script Editors window and is often followed by (), which specifies additional parameters the event relates to.
3. True/False—This double-sided LSL hero is a Boolean variable: the False side of its identity is equal to the integer 0, and the True side of its identity is equal to the integer 1.
4. String—The String is an LSL superhero that contains alphanumeric characters surrounded by quotes, such as "String the LSL Superhero."
5. Integer—This LSL superhero dislikes fractions and is always represented as a whole number in the range of −2,147,483,648 to +2,147,483,647, including 0.
6. Float—Close friends with Integer, this superhero loves to use decimal fractions and can be represented with a high degree of precision, such as 2.234567, or with less precision, such as 2.2.
7. Vector—This superhero is made from three Floats (a team within the team!), and you see them represented with the float numbers enclosed in pointy brackets, such as <0.1, 1.0, 0.2>. This configuration can represent numbers from RGB (color values) or XYZ (location coordinates).
8. Key—This LSL superhero is a randomly generated individual with a 36-character name called a Universally Unique Identifier (UUID). It looks like this: ad9e1cce-0532-11e5-a6c0-1697f925ec7b, and is used to identify a specific object, graphic/texture, sound, and so forth to the rest of the LSL team.

If you are a designer, have you been thinking about what this team looks like and what kind of superhero outfits they would have? If you are a scripter, do you feel the power of these superheroes and what they can create for you?

6.2.3 Learning to Think Alike and Getting Organized on a Project

Learning about scripting and designing vocabularies is only part of the equation. Once you have done that, you need to work on developing a common vision of the project with your collaborator. As Michael Thome (Vex Streeter in the Metaverse) comments, “I think that an important part is to describe the intent of requirements (the who, what, why) and avoid the how’s in high-level requirements. At best, it is likely to be distracting detail. It is, of course, important to do this work as well, but I think too many projects end up with suboptimal implementations because of overspecification of requirements.” Who is involved? What are they doing? Why are they doing it? These are all important questions that must be answered early on by all the parties involved in the project.

The book *Virtual World Design* (page 288) explored the parts of a basic script and what a designer should consider when hiring a scripter to help with LSL scripting on their projects. This knowledge will come in handy when you sit down with a scripter to design a complex script. Let us suppose you have a fancy apartment building in mind, and all the doors will be scripted to open at their owners’ verbal command. Also, you would like the apartment owners to have the option of adding a guest’s name to the door when the apartment is being lent. In Table 6.3 we can look at the views that a designer and a scripter might have as to the aspects of this scripted door project. Reading this table should have shown you the following two things:

1. This kind of door script is probably beyond the abilities of a novice scripter.
2. There are many factors that can go into creating even the simplest things in a virtual world, especially if we want them to be interactive. It is not typical to think of a virtual door as interactive, but in fact, doors most often are to some extent. A good scripter will make a clean, readable door script with understandable comments on it. A great scripter will look for and suggest ways the door can be made more interactive and streamline the script to help it run faster. LSL does not have much documentation about best practices, but an experienced scripter will be familiar with that structure from other scripting languages they use. Now let us consider the question of designer–scripter

TABLE 6.3
Two Views of an LSL Script: The Designer’s and the Scripter’s Views

What the Designer Wants the Door to Do	How a Scripter Thinks about That
The door has to swing inward into the room.	The key prim will be the rotation point.
The door will open and close on a touch.	The door has two states in its script besides the default; they are Open and Shut.
The door knows who the owner is.	The script looks for the owner’s identity, called Key or UUID.
The door lets the owner update a list of people with access to the owner’s place so they can enter.	The script updates a list of the keys of the allowed avatars.
The door needs to open on the owner’s verbal command.	The script has a listener function.
The door needs to shut by itself after some time or when the owner or guest touches it again.	The script has a timer function and a touch function.
The door makes a nice closing and opening sound.	The script plays a sound clip on state change.

collaboration from both sides on a much larger project. Suppose you want to create a highly interactive game-based sim, which has non-player character (NPC) avatars and a complex architectural layout on it, and your time frame is 6 months. A big, multilayered project can be a tough one to grapple with and get organized in a very short amount of time, so the whole team really needs to get on the same page immediately. As you learned in Chapter 4, game documents are the first place to start, and that information can be carried over to the project wiki to help keep the whole team on track. Now you need to get into the details about the scripts you will need and how they will have to function. This is where the understanding of the designer and scripter vocabulary provides you with

TABLE 6.4
A Project Script Version Control System Checklist for the Designer and Scripter

Script Prefix for Inventory	Script Name Example	Script Function	Affiliation
SCR	Scr_Screen1	Level transition on browser	Graphic image for screen
HUD	HUD_Part1	Interconnected HUD parts that combine to make final rebus puzzle	Puzzle parts that are "worn" as HUD
CTRL	CTRL_Puzzlebox1	Game mechanics controls to direct players	Puzzle box object (different item on each level)
LVL	LVL_Level1	Announces level to player	Hidden listener object
INT	INT_Door	Opens door	Handle of door
PHY	PHY_Boat	Moves boat	Boat Key prim
AVA	AVA_Avatar name	Animates Avatar name	Invisible wearable that has Animation Override in it
NPC	NPC_Magical Character	Animates and lets Magical character speak	Magical character tour notecard script inside the NPC
ATT	ATT_Magic_Ring	Allows for avatar access to a special level and kills avatars that are not wearing the ring	"No-kill" script inside the ring
ANIM	ANIM_Sitting	Activates sit animation	Various chairs and sofas
SND-INT	SND-INT_Door_Bell	Plays doorbell sound	Door
SND-PHY	SND-PHY_Boat_Bell	Plays boat bell sound	Boat
SND-NPC	SND-NPC_Magical character_Hello	Plays Magical character hello sound on avatar approach	In or attached to Magical character
SND-BLDG	SND-BLDG_Dungeon_Whispers	Plays Dungeon Whispers sound on avatar approach	Hidden prim in Dungeon
SND-LAND	SND-LAND_Birdies	Loops bird call sounds on landscape	Hidden prims on landscape
SND-PART	SND-PART_Magic_Spell_Dust	Plays Magic Spell sound with emission of magic dust particles	Inside of various magical objects
MUSIC-AMB	MUSIC-AMB_Radio	Loops sound from radio	Hidden prim, source of radio sound is not seen
MUS-LVL	MUS-LVL_Puzzle_Correct	Plays sound when player has solved the puzzle on that level of the game	Sound comes from HUD puzzle piece

a secure foundation for your mutual creativity. As you settle in and the project starts to progress, the designer and scripter should each have a place to store iterations of their content as they develop. For the designer, that could be a cloud-based site like Dropbox or Google Drive. The scripter can use *git* (http://git-scm.com/) repositories to store and organize the iterations on the scripts. It is crucial that the scripter and designer agree upon the version names and enumeration, so that they can work with an understandable version control system. In Table 6.4 is a checklist for designers and scripters, so that each script needed for a project can be placed in an organized list, named for its type and identified by its function. This particular table is filled with examples of scripts that will go into one version of the game-based sim, a puzzle hunt game.

6.3 CASE STUDY: SCRIPTED TREES AND EMERGENT PLAY ON SUNDIAL, THE SEASON CHANGING SIM

Designing and creating a game-based sim oftentimes involves some simple ideas that combine to create unexpected complexity and interactive interest for the visitor. Such was our experience with the first region we built in Second Life, "Sundial–The Season Changing Sim" (2008, Second Life). In the next section we will discuss this project as a case study. In Figure 6.3 is a schematic overview of the interactive elements on Sundial and how they connected to create emergent play.

6.3.1 The Concept That Started with a Few Trees

In mid-2008, after a few months of exploration in Second Life and creating a blog about it at http://travelswithchip.blogspot.com/, I decided to build some interactive season-changing trees for a small skybox garden I had designed. Because I had no experience with programming or using Linden Scripting Language (LSL), I asked my friend Vicki Brandenburg, a skilled LSL scripter, to help me out. I explained to her that I wanted the trees to change seasons, just as they do in my native climate in the northern hemisphere. Vicki took these as my "specifications" for the tree and asked more questions about how it might work. Then I created sculpt map/graphics to make organic-looking tree trunk shapes, and a "wardrobe" of graphic/textures for the tree branches and bark that would reflect the seasonal changes, which she could utilize in her season-changing tree script. Vicki created the first seasonal tree script in August 2008. Here is her script commentary on the top of the script:

```
//SEASONAL TEXTURE CHANGING SCRIPT
//by Vicki Brandenburg 8/2008
//Responds to link message
//string msg = autumn, spring, summer, winter
//textures in inventory should be alphabetized in that order
//integer color = RGB color code (24bit hexadecimal numbers used in this script)
//includes code to remotely change prim color as well as texture
//script changes texture on all faces of the prim on link message command
//textures are read from prim inventory in alphabetical order
```

Since this was our first collaborative effort, we needed to test every component, making sure that my graphic/texture names matched her naming conventions in the script and that the tree parts were linked in the proper configuration. As you can see from her comments, the first season-changing tree script was looking for a string of word triggers (autumn, spring, summer, winter). These trees would listen for and change

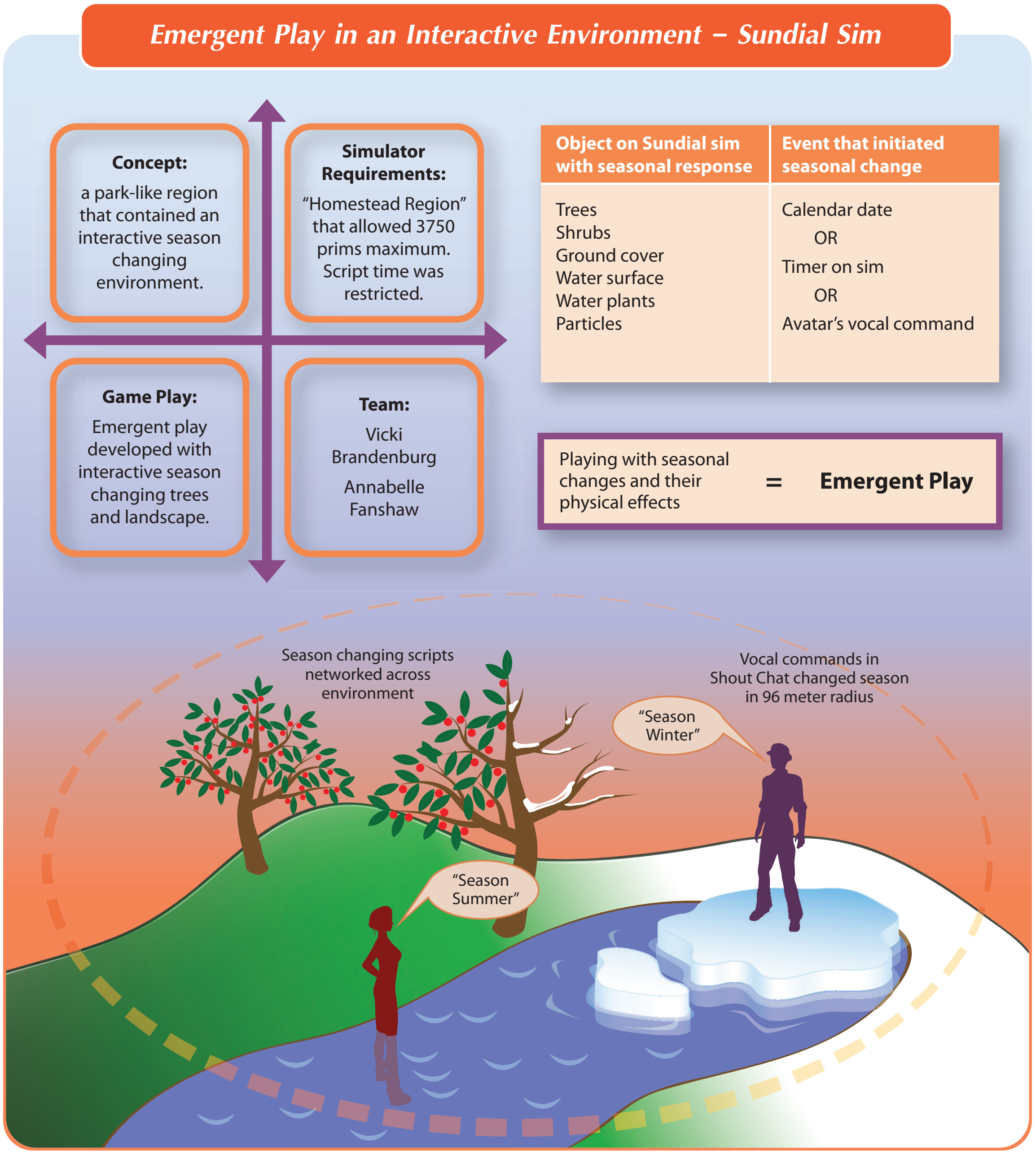

Emergent Play in an Interactive Environment – Sundial Sim
Concept: a park-like region that contained an interactive season changing environment.
Simulator Requirements: "Homestead Region" that allowed 3750 prims maximum. Script time was restricted.
Game Play: Emergent play developed with interactive season changing trees and landscape.
Team: Vicki Brandenburg Annabelle Fanshaw
Object on Sundial sim with seasonal response
Trees
Shrubs
Ground cover
Water surface
Water plants
Particles
Event that initiated seasonal change
Calendar date OR Timer on sim OR Avatar's vocal command
Playing with seasonal changes and their physical effects = Emergent Play
Season changing scripts networked across environment
Vocal commands in Shout Chat changed season in 96 meter radius
"Season Winter"
"Season Summer"

FIGURE 6.3 Schematic of interactivity and emergent play on Sundial—The Season Changing Sim.

with a local chat command from the avatar. Some of the early trees we made had a feature that would make them glow for a few seconds when activated, but that was dropped when we started to make forests of them, and the compound glows became too intense. Vicki started with my primary request that the trees change to four different seasonal looks, and built up the script from that foundation. Eventually she had a script built for many contingencies. The final version of the first generation of season-changing trees would respond to vocal commands from an avatar using local chat, listen for the timer script that instigated seasonal changes in the entire region, and automatically respond to the calendar date, changing on the equinox. In hindsight, it seems that we really only needed the timer event and the vocal command response to make this tree script contribute effectively to an interactive landscape. This project was the beginning of a great collaboration that continues to this day.

6.3.2 How It Grew into an Entire Environment That Worked under Limited Conditions

Eventually, a skybox garden was just not big enough for our developing tree franchise, so we took on a project to develop a "season changing" sim, which was called "Sundial." This was a "Homestead" region in Second Life, so it was limited in terms of the number of primitives it would support (3750 total prims) and had an avatar limit that was set to 20. We were asked by the owner to keep our script load to a minimum, and that led us to designing a "network" of trees across the terrain. Ten to 15 trees were clustered around and linked to a common "brain," containing the "listener" scripts that would activate the seasonal changes. By using this system, we reduced the script load on the server considerably, and could create a complete seasonal change across the landscape with many fewer scripts.

6.3.3 Our Discovery of Emergent Play

Previously mentioned in Chapter 1, Section 1.3.2, we soon discovered the possibilities for emergent play behavior in an interactive environment, and Sundial inadvertently became our first game-based sim. Not a game design per se, but we did provide an environment that encouraged game creation and game playing. A happy accident, you might say, that has led us on a path of discovery and learning since that event. In Figure 6.4 are images from the creation of our initial trees, the skybox garden, and Sundial, illustrating the process of content development as we collaborated on the interactive scripting.

Our collaboration renews itself from time to time, as Vicki and I discuss new ways to use these trees, remix the script, and add new kinds of responding vegetation to our growing list of content. Vicki has written her scripts as numbered versions (i.e., tree script v1, v2, v3, etc.), so we can always return to a previous version, and she always adds many comments at the top about how and why we made that particular script. Lately, she has developed a leaner tree script for a grove of trees that uses only one script, and the performance is very fast.

6.4 CASE STUDY: CREATING A GAME-BASED SIM CALLED "INLAND: SEARCH FOR THE SY"

Early on, IBM recognized the value of providing a means for showcasing the virtual content built for Second Life and other virtual worlds. From 2006 through 2010, IBM maintained a number of sandbox and exhibit regions in Second Life and on Reaction Grid (http://www.reactiongrid.com/). Andrew Sempere (Tezcatlipoca Bisiani in SL), a research designer at IBM's center for Social Software, curated the exhibits, which ran continuously. Although the regions are closed now, their legacy lives on in the Metaverse.

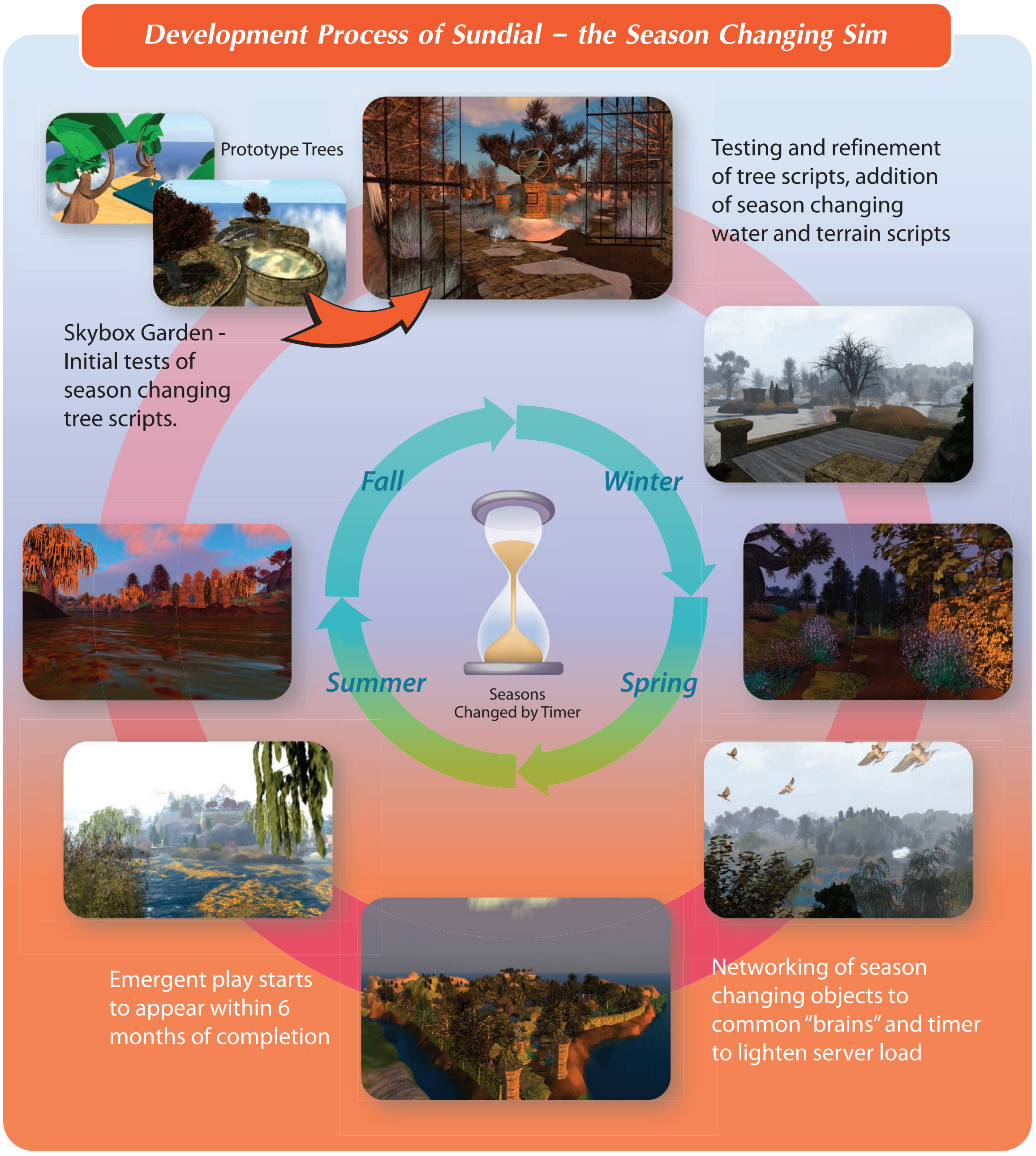

Development Process of Sundial – the Season Changing Sim
Prototype Trees
Testing and refinement of tree scripts, addition of season changing water and terrain scripts
Skybox Garden - Initial tests of season changing tree scripts.
Fall
Winter
Summer
Spring
Seasons Changed by Timer
Emergent play starts to appear within 6 months of completion
Networking of season changing objects to common "brains" and timer to lighten server load

FIGURE 6.4 The design process and emergent game play development on Sundial—The Season Changing Sim.

6.4.1 Building "Inland: Search for the Sy" for the IBM Exhibit Program

Imagine the virtual world without avatars. Imagine the decay of the boundary between a virtual world and the illusion of the real world. Imagine the sensation of being in 2 worlds at once. This virtual landscape embraces that ephemeral sense of reality and plays with the notion of the death of illusion. … What is the mirror of a virtual world reflecting? What is the goal here? What is the game?

Excerpt from original proposal, "Table Top World," to the IBM Exhibit Program, June 2010

In early 2010, my build group (Alchemy Sims) was invited by the IBM Exhibit program to create a virtual environment for one of its regions. We initially called this project "Table Top World." These first concepts were inspired by the book *The World without Us* by Alan Weisman and readings from HG Wells's *The Time Machine*, especially where these sources related to new visions of the world after the extinction of man on Earth. As you can see in Figure 6.5, the process of developing the concept, building the environment, and creating the game-based experience was a complex, multilayered project.

Once again, we were confronted with the problem of how to make the 3D virtual environment become more engaging and interactive. Table Top World was more like a diorama at the Museum of Natural History than an interactive sim. To counter that effect, we developed a narrative story in the form of a science journal diary written by the fictional character Dr. Aubrey Wynn (see Appendix B for the most recent version of this). Now, the intertwined themes of the "prehistoric shadow biosphere" and "computer based silicon life forms" could coexist. Large, prehistoric plants would engulf the remains of architectural structures, huge insects and strangely mutated sea creatures would inherit the world, and the mysteries described in Dr. Wynn's diary would provide reasons to explore the sim. We invented the Sy, who are silicon-based creatures living in the chips of our computers. Inspired by the fairy folk called "Sidhe" (pronounced "shee") or the "sí" in modern Irish, they resemble living fractal structures. Unknown to humankind, the Sy home world inside our computers was being threatened by the encroachment of virtual worlds. To defend their existence, the Sy are changing our human-based avatars into Sy–Human hybrids, and leading them into Sy Space for eternity. After the backstory and narrative was finished, we began on our content fabrication by building and scripting objects for the project. Simultaneously, Layton Destiny (3D modeler), Vicki Brandenburg (LSL scripter), and I started to create the rough prototypes of what would become the building structures, interactive scripts, creatures, vehicles, and landscapes of "Inland: Search for the Sy," as the project was now officially known.

6.4.2 Collaborative Creation on a Game-Based Sim

The preliminary press release about our newly renamed project included this copy:

"Inland: Search for the Sy" is a game-based sim that encourages play by creating a 3D real-time persistent space that has an interesting concept/backstory with believable characters. "Inland" engages the visitor in play by providing the opportunity to have fun while solving the central mystery of the story through the mastery of puzzles and object construction. "Inland: Search for the Sy," also provides a fertile ground for the visitors to create their own scenarios and unscripted game play within the space by providing game-based avatars, vehicles and nested environments with props that can be manipulated. "Inland," is also proof of concept, that game play can be prototyped rapidly in a virtual world space like Second Life. How is this different? It's not a "role play" sim. This sim is based on pure imagination, an original story that has been developed in 3D.

The real collaborative work between Vicki and me started when we began to create the scripted objects needed by the visitor to play the mystery hunt game on Inland. As you can see from the overview of the game flow in Figure 6.6, we came up with a complex, multistep approach to the puzzles included in the

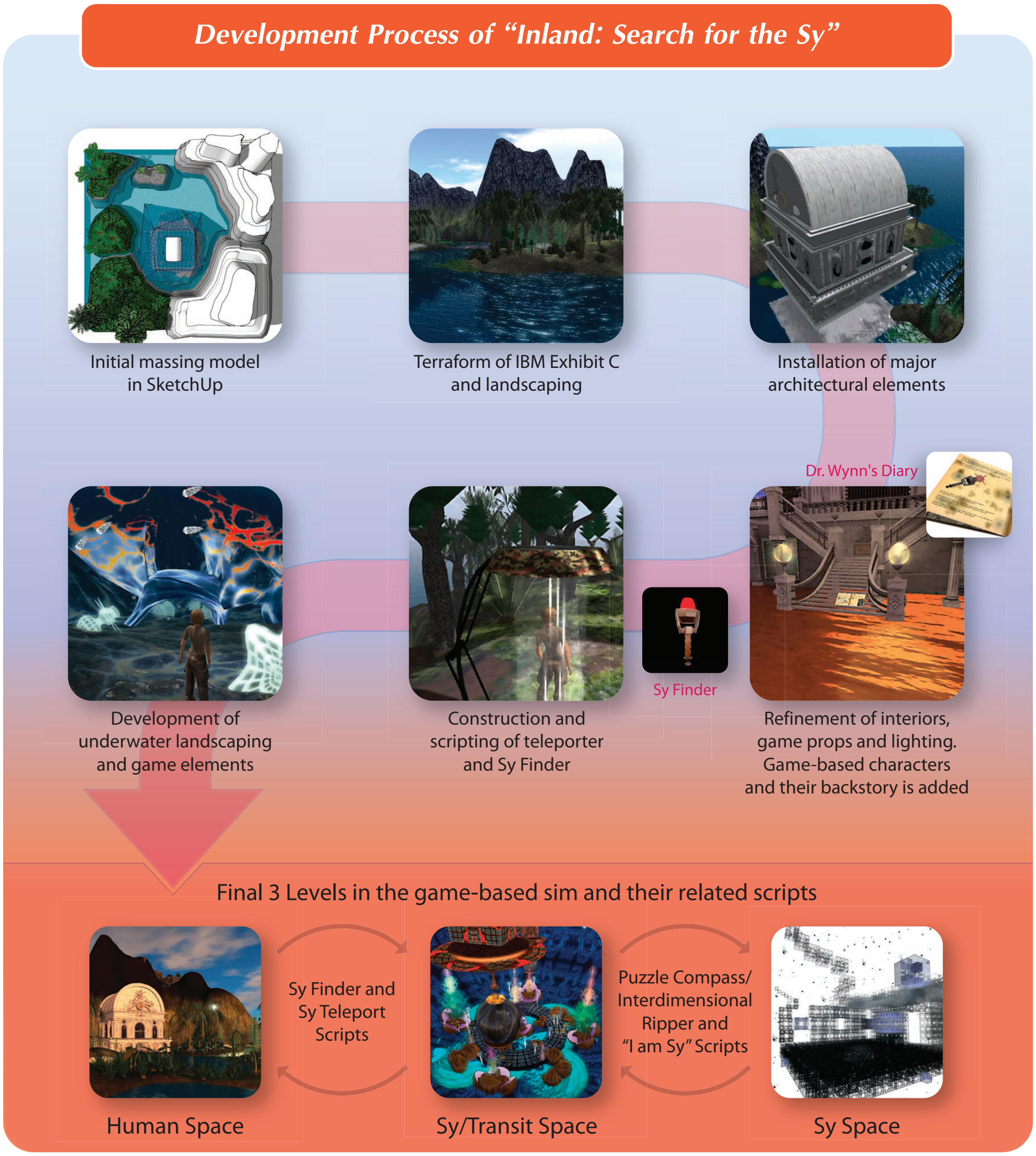

Development Process of "Inland: Search for the Sy"
Initial massing model in SketchUp
Terraform of IBM Exhibit C and landscaping
Installation of major architectural elements
Dr. Wynn's Diary
Refinement of interiors, game props and lighting. Game-based characters and their backstory is added
Sy Finder
Construction and scripting of teleporter and Sy Finder
Development of underwater landscaping and game elements
Final 3 Levels in the game-based sim and their related scripts
Sy Finder and Sy Teleport Scripts
Puzzle Compass/ Interdimensional Ripper and "I am Sy" Scripts
Human Space
Sy/Transit Space
Sy Space

FIGURE 6.5 The developmental design and scripting process from "Inland: Search for the Sy."

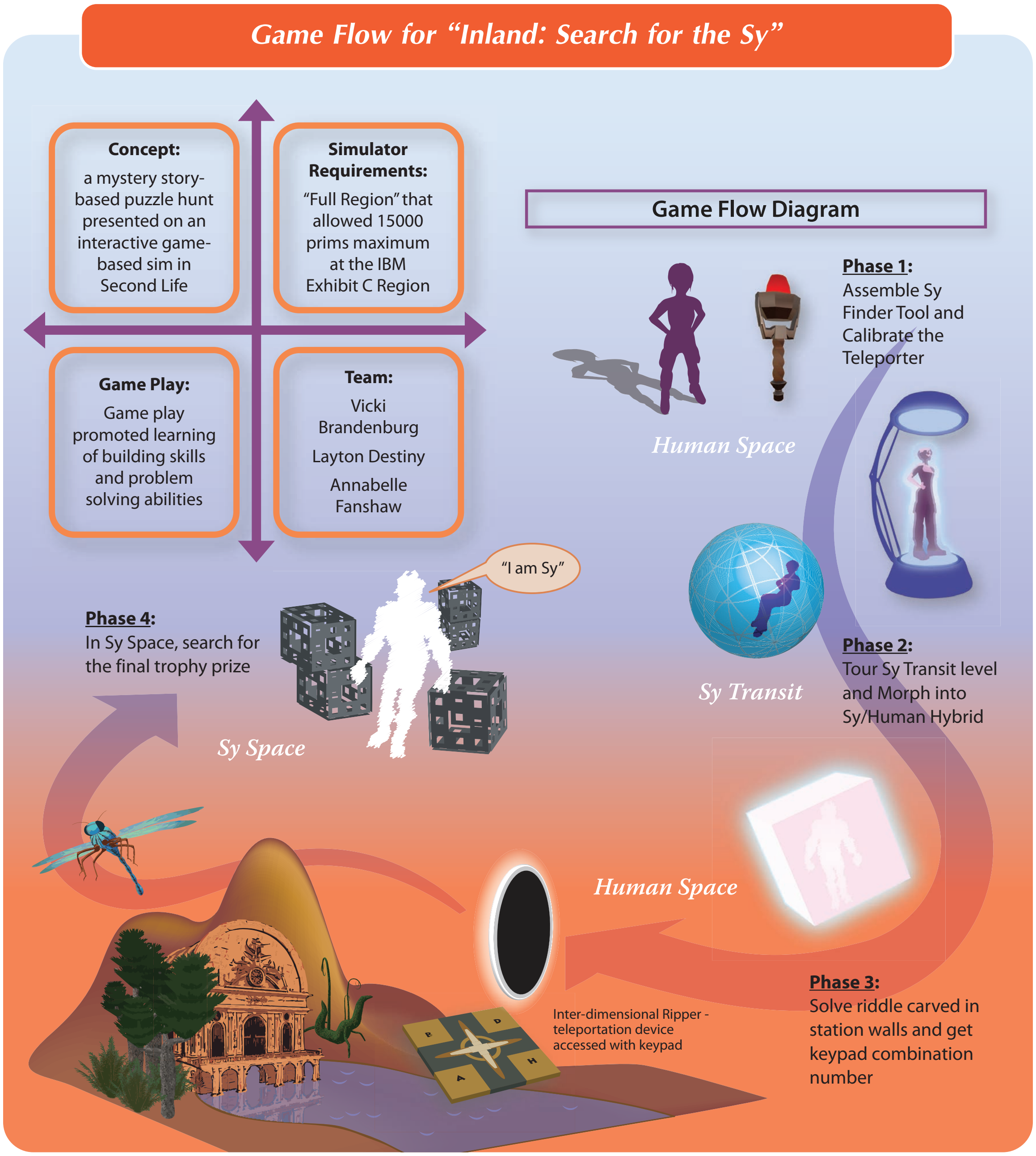

Game Flow for "Inland: Search for the Sy"
Concept: a mystery story-based puzzle hunt presented on an interactive game-based sim in Second Life
Simulator Requirements: "Full Region" that allowed 15000 prims maximum at the IBM Exhibit C Region
Game Play: Game play promoted learning of building skills and problem solving abilities
Team: Vicki Brandenburg Layton Destiny Annabelle Fanshaw
Game Flow Diagram
Phase 1: Assemble Sy Finder Tool and Calibrate the Teleporter
Human Space
Sy Transit
Phase 2: Tour Sy Transit level and Morph into Sy/Human Hybrid
"I am Sy"
Phase 4: In Sy Space, search for the final trophy prize
Sy Space
Human Space
Phase 3: Solve riddle carved in station walls and get keypad combination number
Inter-dimensional Ripper - teleportation device accessed with keypad

FIGURE 6.6 Game mechanic and flow schematic for "Inland: Search for the Sy."

environment. After the visitor landed on the "Human Space" level of the sim for their initial visit and made the decision to play the game by solving the puzzles, they had five basic tasks: (1) assemble the "Sy Finder," which provided them with the means to teleport up to the next level, which we called Sy Transit; (2) evolve into a Sy–Human hybrid so they could survive in Sy Space; (3) find and solve the verbal puzzle in the main station area on the Human Space level; (4) enter the encrypted code from that puzzle onto the interdimensional ripper keypad below the main station to enter Sy Space; and (5) search and find their trophy prize in Sy Space. Providing the visitors with these scripted objects that allowed them access into two other virtual environments located high above in giant hollow spheres really opened the game dimensionally, giving it a greater sense of travel and transition. Obviously, a three-tiered environment controlled by scripted artifacts and teleportation devices took some time to create. To embed interactivity within the Sy Finder and other parts of the environment, Vicki created three complex scripts:

1. Sy Finder/Teleportation script—This interactive script coached the player along as they built their own version of a crystal oscillator, known as the Sy Finder tool, from bits they found lying on the terrain. When they had completed this step and were standing in a specific place on the terrain, the Sy Finder would activate a special visual effect and give them a chat channel number so they could use the nearby teleport to go up to the Sy Transit level.
2. Puzzle Compass/Inter-dimensional Ripper—This interactive script responded to touches on the quadrants of a touch-pad object. When the player touched the quads in the right sequence, an inter-dimensional rip appeared in front of them, allowing access to the top level, or Sy Space.
3. "I am Sy"/Prize Giver—This script functioned primarily for the third-stage game mechanics. Concealed in the Sy Avatar worn by the player, it prevented other players from jumping into Sy Space or visiting the area without playing through the game. Any player who was not wearing it, would be "killed" immediately and sent back to their home base. It also prevented players from taking more than one prize, and recorded the names of players that had won.

Once we worked out the mechanics of the game, how the flow would go, and what the goals were, the designer–scripter collaboration was very fluid. As the designer, I would rough out the kind of artifact or object I was thinking about, and Vicki would give me a preliminary version of the script to test it with. Sometimes she would ask me to tweak the design, so the interactivity would work better. This was the case with the interdimensional ripper keypad, which held a script she had set up to recognize touches in various locations on its surface. As often the case with long-term collaborators, we knew what each other's styles of working were, and could anticipate and provide for needs ahead of the schedule.

6.4.3 Putting It All Together Onsite

In November 2010, the team began the initial installation at the IBM "Exhibit C" region. When the time arrived to load it all in, Layton and I set up the architecture and landscaping, and placed the artifacts around for Vicki so she could install the scripts. Key objects like the interactive scientific journal (Dr. Wynn's diary) were left in prominent places so the visitor would be introduced to the game very quickly. This journal contained the mystery hunt backstory and introduced the visitor to the overall purpose for the sim. We also included objects that were red herrings to provide misleading information for the puzzle solvers, hidden traps, as well as clue-giving objects that would change from time to time over the course of the exhibition. As you can see in Figure 6.6, the flow of the game was getting quite complex.

6.4.4 Playtesting and Learning about Gamers

When the build phase was completed, we enlisted the help of the MetaHarpers Cabal, an art-based group in Second Life, to do some playtesting for us. It immediately became apparent to us through this process which aspects of the game were too easy and which aspects needed more clarification and description for the player to understand the rules. When the game opened to the public, I spent a significant amount of time observing the players and occasionally helping them in their quest for the Sy. We discovered that the scope of player abilities is vast, from the "pros" that could run through the game in an hour to the beginners who did not even know how to link two primitives together. In fact, one of our beginner players went so far as to remove the scripts from the objects that they were supposed to assemble into the Sy Finder, effectively breaking it. The experience of watching people play your game, to talk with them as they struggle with the problems in it, and to share their triumphs as they finish is one of the most satisfying experiences I have ever had. In Figure 6.7, you will see some images from the final game playing areas for Inland: Search for the Sy.

6.5 TOP 10 LIST OF SCRIPTS THAT YOU SHOULD HAVE

You do not have to be building content very long in a virtual environment to realize that you need some specific scripts in your inventory. In Table 6.5 is a Top 10 list of the most commonly used scripts in my inventory. While your needs may not be so extensive, it is a good idea to find sources for these scripts, test them, and customize them for your needs. For instance, you could tailor a basic particle script to be "touch activated" (using the `touch _ start` and `touch _ end` event triggers in your code), or you could add the timer event function `llSetTimerEvent` to create a timer in the script, as we did with the season changing trees on Sundial. On the right side of Table 6.5 is a partial list of webpages, books, and locations where you can get these basic types of LSL scripts. Make sure that the scripts you are using are from Creative Commons or "free to use" with no licenses attached to them. Respect the efforts of other content creators.

6.6 OTHER SCRIPTING LANGUAGES USED IN VIRTUAL WORLDS

As with most globally based human endeavors, the use of scripting languages in virtual worlds is diverse. Linden Scripting Language (LSL) was created for Second Life and somewhat resembles the C programming language. For some folks, learning LSL is a good way to ease into learning other programming languages like C# and JavaScript [6].

6.6.1 The Big Picture

In Table 6.6, there is a partial list of virtual worlds and the scripting languages they use. As you can see, LSL is only prevalent in two of them, and JavaScript and C# are quite common in several instances. Some platforms like Unity and OpenSim offer the scripter several options. The future of virtual world scripting is unknown; we are not sure how many of them will take advantage of JavaScript or how many will create new kinds of scripting languages. Fortunately, there are lots of people who are willing to collaborate and teach others what they know in most virtual worlds.

6.6.2 What Scripting Language Should I Learn?

The question of what scripting language to learn has often crossed my mind. As a designer, most of my work is done through the use of tools like pencils, software and cameras. However, the power of knowing how to

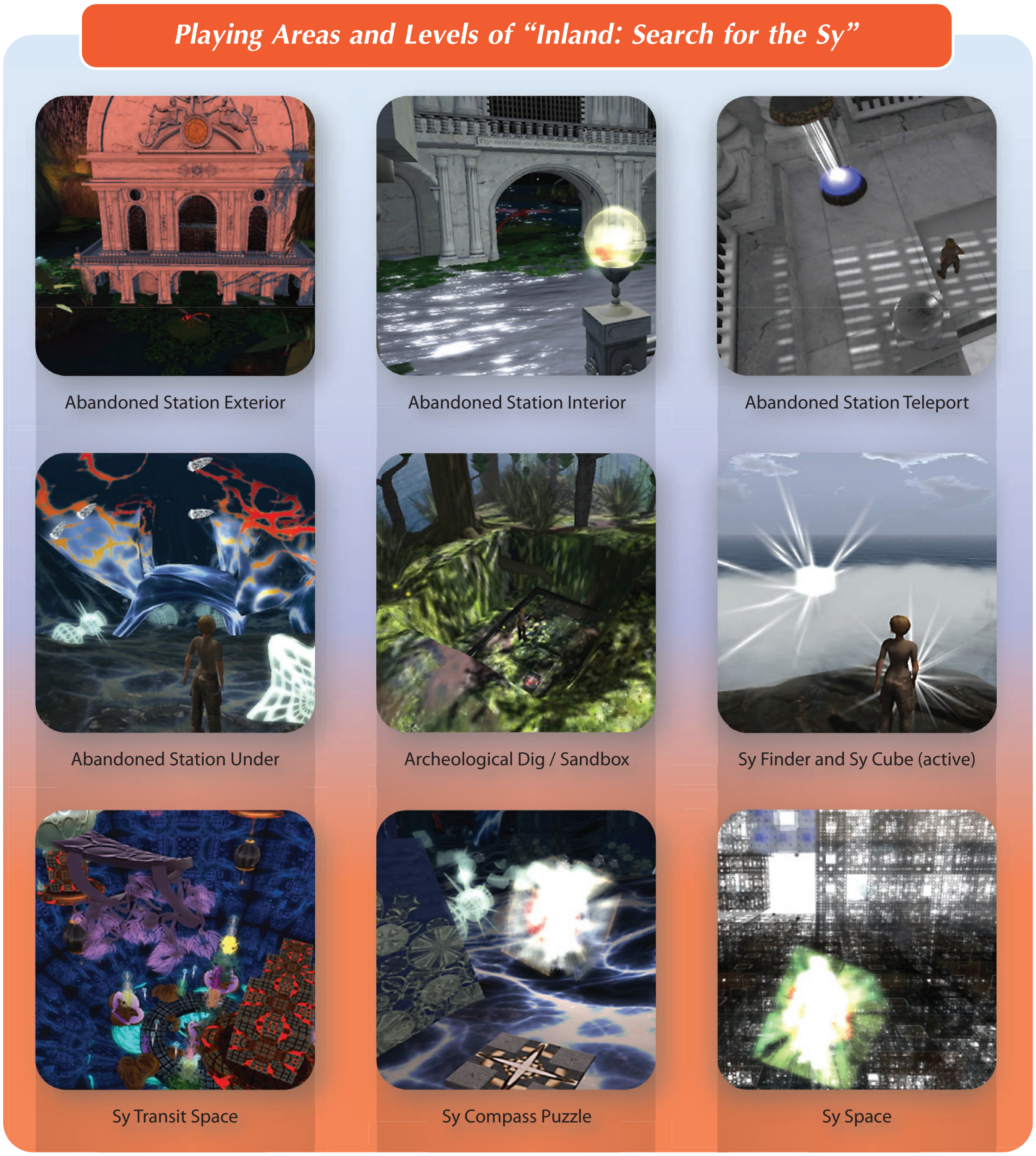

Playing Areas and Levels of "Inland: Search for the Sy"
Abandoned Station Exterior
Abandoned Station Interior
Abandoned Station Teleport
Abandoned Station Under
Archeological Dig / Sandbox
Sy Finder and Sy Cube (active)
Sy Transit Space
Sy Compass Puzzle
Sy Space

FIGURE 6.7 Play levels of the game-based sim "Inland: Search for the Sy."

TABLE 6.5
Top 10 Scripts That You Should Have

Type of Script	What It Does	Where to Get Them
Sit Position Script	Establishes sitting position of avatar on a prim, and allows for adjustment of position and choice of pose.	If you are not yet working with a scripter, and want a script to start with, there are several sites that offer free LSL scripts, some of them will work in both Second Life and OpenSim.
Basic Particle Script	Sets up a particle system that is easy to modify	**Second Life specific:** http://www.free-lsl-scripts.com/
Sound Ambiance Script	Allows for sounds to be combined into overall virtual environment ambiance	http://wiki.secondlife.com/wiki/LSL_Library http://lslwiki.net/lslwiki/wakka.php?wakka=ScriptLibrary http://www.hypergridbusiness.com/category/resources/scripts/
Basic Vehicle Script	Sets up modifiable framework for creating working vehicles	**OpenSim specific:** http://opensimulator.org/wiki/OSSL_Script_Library
Basic Teleport Script	Interactive teleport script that allows for destination choices	OpenSim Scripting Forum: http://forums.osgrid.org/viewforum.php?f=5
Graphic/Texture Animation Script	A suite of texture animation scripts that will move graphics across the surface of an object or primitive	**Places to visit in Second Life for getting specific scripts:** Particle Lab: http://maps.secondlife.com/secondlife/Teal/180/74/22
Basic Communication Script for Signage (includes settings for Proximity, Collision, and local/shout chat)	Provides the capacity to create hovertext and/or proximity-activated audible instructions for signage in the environment	Graphic/Texture Tutorials: http://maps.secondlife.com/secondlife/Livingtree/128/102/25 General Freebie Section of Builder's Brewery: http://maps.secondlife.com/secondlife/Builders%20Brewery/110/156/23 SL Location: http://maps.secondlife.com/secondlife/Hennepin/15/183/107
Visitor Sensor and Greeting Script	Watchdog script that tracks the visitors to your sim, and greets them	**Publications about Scripting with script resources:** *Scripting Your World* by Dana Moore, Michael Thome, and Dr. Karen Zita Haigh (Wiley Books, 2008)
Basic Vendor Script	Allows you to give away or sell items	*Scripting Recipes for Second Life* by Jeff Heaton (Heaton Research, 2007)
Scrubber Script	Allows you to remove all scripts and systems from the primitive, especially hover text and particles	

TABLE 6.6
Other Scripting Languages and the Virtual Worlds That Use Them

Virtual World	Scripting Language
Second Life 1.0	LSL
OpenSim	LSL, OSSL, C#
Unity	JavaScript, C#, Boo
High Fidelity	JavaScript
Advanced Distributed Learning (http://www.adlnet.gov/)	JavaScript

utilize a scripting language is quite useful, especially if you are designing and building game-based virtual worlds. Ultimately, JavaScript became the first language I chose to learn, but that decision is based on the need to create for the Unity platform as well as High Fidelity. If you are only using the current first version of Second Life and OpenSim, then perhaps LSL is where you should start. If you want to plan for future versions of Second Life, then learning C# would probably be a good idea.

6.7 PROJECT: "SCRIPT ROULETTE"—A COLLABORATIVE GAME OF CHANCE

This game of chance is designed to foster collaboration between a designer and a scripter, as they strive to compete with other teams in the creation of a randomly selected scripting project. The goal of the game is to create the most interactive object possible given the parameters set by the role of the dice.

Please note that if you are a teacher using this project for your class you may want to incorporate the following game mechanics:

1. A time limit that is set for doing this work in class or assigning the project for homework.
2. If you have lots of scripting novices, you may want to provide your class with a collection of seed scripts (from sources in Table 6.5) so there is something for them to start with.
3. If you have more experienced scripters, then perhaps you can provide scripts with errors in them so they get more practice with troubleshooting.

6.7.1 Setting Up to Play

You will need to gather a few things before you can play this game. Go out and find the following:

1. If you are a designer who does not use a scripting language, find someone who is experienced in LSL (or OSSL*, JavaScript, or C#) to be your partner.
2. If you are a scripter who does not create content or make 3D models, find a designer/builder who is experienced in these things.
3. A pair of six-sided common dice.
4. Paper and pencil for notes and doodles.
5. Script editor software on your laptop or desktop computer. Here is a link for an LSL editor: http://sourceforge.net/projects/lsleditor/.
6. A leader board to keep track of scores and team standings.

Please note: OSSL is not completely compatible with LSL. There are some differences that require consideration if you want this script to work in Second Life and OpenSim.

Now that you have found your partner, you can settle down to play the game. Hopefully there are enough players so you can play the game as a tournament. In Table 6.7 are three columns that list the following categories: Object, Script Type, and Interactivity. There are 11 options in each column, one for each possible number on the pair of dice (2–12).

6.7.2 Playing

The game starts with each team throwing their dice three times, noting the combined number on the dice, and choosing the corresponding item on the lists in the three columns of Table 6.7. For instance if they rolled a 3,

TABLE 6.7
"Script Roulette" Options Chart

Number on Dice	#1: Object to Script	#2: Script Type	#3: Event Interactivity Required
2	Chair	Adjust Pose	Touch button on Screen Menu
3	Planet	Particle system	Touch surface
4	Magical ring	Rezzer	Local Chat Command
5	Top Hat	Hover object	Land Collision
6	Snake	Teleport	Identifies presence of Owner
7	Land Vehicle	Spin/Rotate	Avatar Proximity
8	Sculpty Fruit	Animate Texture	On Rezz
9	Lamp	Hover Text	On Avatar Collision
10	Beach ball	Vehicular movement	Set Timer
11	Torch	Chat in Nearby	Specific Date on Calendar
12	Planter	Scale Change	On Attach

8, and 6, then they would choose these three items: Planet, Animate Texture, and Identifies presence of Owner. That team would then have to create a planet that animates its graphics/textures when it senses the presence of its owner, a friendly sort of planet, it seems. Think of what you could do with an animated smiley face graphic!

6.7.3 Scoring

If this is a time-limited game, then the first team that creates the functioning interactive object and demonstrates it in the virtual world will win 5 points. The team in second place will win 3 points, and the third place team gets 1 point. The winner is decided after three rounds of play.

If this is a homework assignment, then the group will vote on what it considers to be the best effort, and that team takes the prize for the week. Over the semester, the team that has won the prize most often is declared the winner of the game.

6.7.4 Adding Collaboration Incentives

To encourage collaboration and prevent one member of the team from doing it all, the scripter and designer should switch roles on every other turn. Neither should be considered to be the go-to person for only the design or scripting tasks.

6.8 CHAPTER SUMMARY AND FINAL THOUGHTS

Creating a culture of collaboration takes work, hard work. Everyone has to step up their game, because collaborating teams are only as strong as their weakest link. Collaborating teams are at full power when every member knows their skills and is sharing them completely with the whole group. There should be an overall attitude of willingness to learn from each other. As the project progresses, the entire team should be able to see the incremental changes in the designs and scripting by using a project wiki or common source of information. This kind of visibility will give team members a sense of accomplishment and allow them to see any problems early on. People do have natural skill proclivities and if someone is a gifted natural scripter, if they find a deep sense of satisfaction and flow while they work on code, then pave the way for them to actualize that talent. The same process applies to gifted artists and designers. However, this does not mean they should all be sitting on their individual mountains like so many hermits. Bring them together onto the same mountain, show them where the goals and skills overlap, and soon they will understand they can go further and fly higher together as a team.

REFERENCES

1. A sufficiently complex build might require a team of code writers, especially if there are multiple skillsets required (e.g., LSL scripting, PHP for web services, and MySQL for the database).
2. Linden Lab, "Check Out Experience Keys in … The Cornfield," *Second Life* (blog), July 14, 2014, accessed August 30, 2014, http://community.secondlife.com/t5/Featured-News/Check-Out-Experience-Keys-in-The-Cornfield/ba-p/2776400.
3. The Restrained Love Viewer (RLV), a favorite of Second Life's niche audiences, is another example of this. An RLV-activated viewer allows properly scripted objects to exert additional control over your viewer experience. The RLV-enabled viewer will also allow for control over other avatars who are wearing the scripted object. These additional controls can be leveraged to implement additional functions such as traps that disallow movement, teleports without asking permission, and objects that attach without user interaction. In many ways, the RLV foreshadowed what could be leveraged for development in the User Experience API.
4. "Lua (programming language)," *Wikipedia*, accessed August 31, 2014, http://en.wikipedia.org/w/index.php?title=Lua_(programming_language)&oldid=639283081.
5. James Benedek, "A Basic LSL Tutorial," *Second Life Wiki*, accessed September 2, 2014, http://wiki.secondlife.com/wiki/A_Basic_LSL_Tutorial.
6. "Does Anybody Think That Learning Linden Scripting Language Is Worth It?" *Stack Overflow* (blog), accessed September 10, 2014, http://stackoverflow.com/questions/575547/does-anybody-think-that-learning-linden-scripting-language-is-worth-it.

7 Designing with Level of Detail (LOD)

I like design, I like details, to me it is just another form of self-expression.

John Malkovich

In the real world you can get closer and closer to an object while observing its infinite detail. In a virtual world this experience is limited to the finite capacity of your graphics processor. As you move your avatar (or agent) around in a virtual world the client viewer and the virtual world servers are working hard to stream an image of consistently high detail onto your monitor screen. This level of detail (LOD) is influenced by several things including but not exclusive to (1) the amount of graphics processing power in the computer, (2) the complexity of the geometry and terrain you are viewing with your camera, (3) the draw distance settings on your camera, (4) the size and type of graphics/textures that are being used on the geometry, and (5) the type of LOD processing being utilized by the viewer. In Figure 7.1 is a general overview of how the LOD for the mesh models and their corresponding physics meshes are manipulated in the virtual environment. The three major influences on LOD display are (1) the player or person observing the virtual world, (2) the client viewer interface (mesh enabled) and (3) the server (or sim cache), which are linked in an interconnected feedback system that selects and displays the most appropriate LOD to the player depending on their graphics preferences and display settings in the client viewer.

As you can see on the bottom of Figure 7.1, the asset server (sim server or cache) is holding various options for the geometrical and physics LOD of a particular object. As the camera gets near to the object, a high LOD version of the model's geometry is displayed, and as the camera moves back the LOD steps down until it is at the lowest possible level for that object. In between the various levels of detail, there is a "switching zone" (utilizing alpha blending) that helps to ease the transition between the changes in geometrical complexity and to prevent the visual effect called "popping." All of this camera information is handled by the client viewer, and that is where the various levels of LOD are chosen. The client viewer is working hard to maintain the highest resolution possible on your screen, given where you have chosen to put your camera, the amount of detail available for the particular object you are looking at, and the amount of screen space the image of that object will take up. Note that the physics LOD is much simpler throughout many of these transitions. Designing to keep the physics shape as simple as possible at all levels of detail helps to speed up the processing done by the physics engine in the server.

7.1 OVERVIEW OF DESIGNING WITH LEVEL OF DETAIL (LOD)

Although we are several years away from totally realistic 3D worlds running in real-time on our displays, the capacity to achieve that goal increases with each new generation of graphics cards. In this chapter you will gain a deeper understanding of how to leverage the use of LOD, squeeze more performance out of the graphics card, and end up with a better looking virtual environment. Here are the topics we will explore:

- LOD fundamentals for the designer
- Designing for LOD optimization on the upload to a virtual world scene

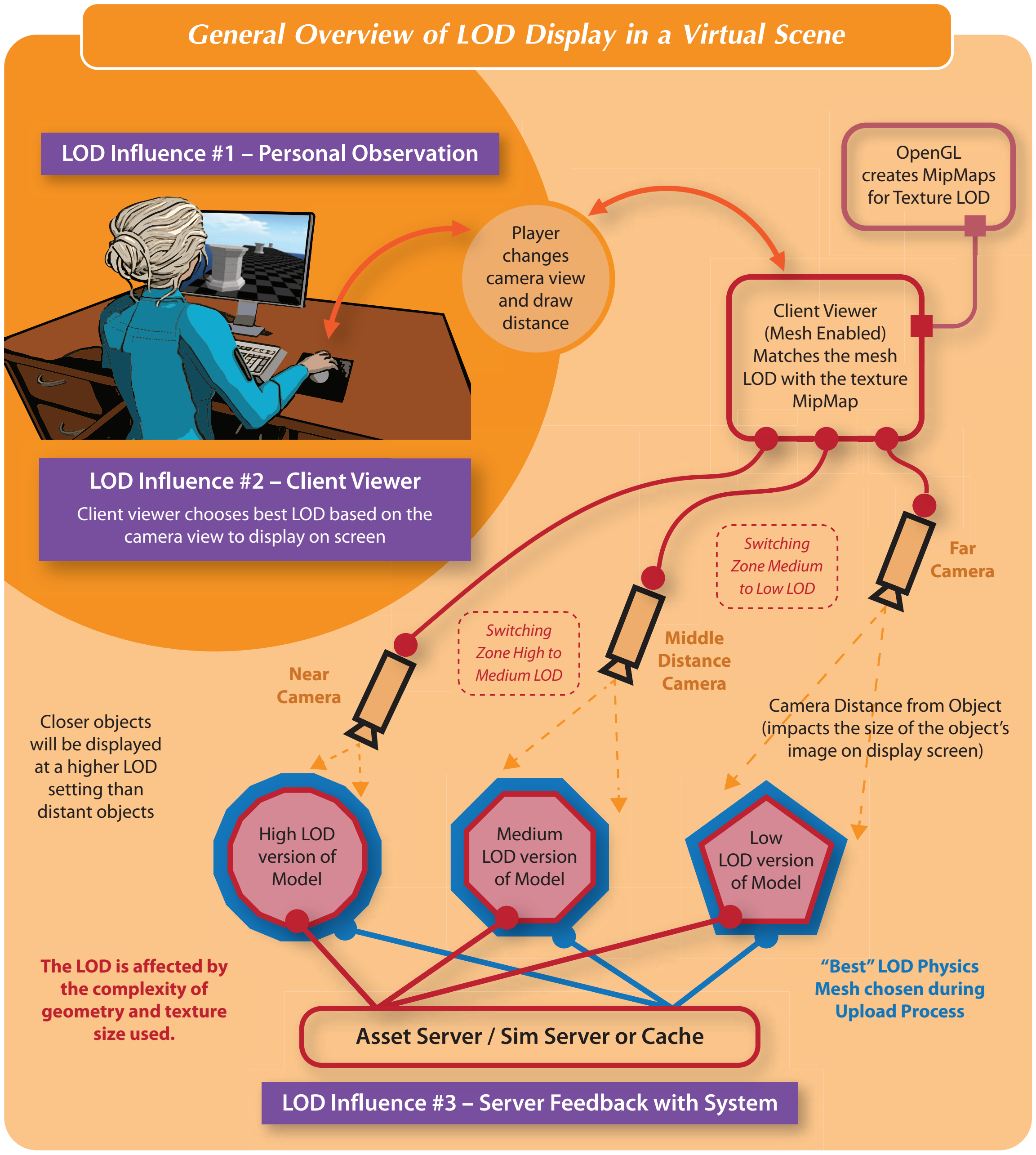

General Overview of LOD Display in a Virtual Scene
LOD Influence #1 – Personal Observation
Player changes camera view and draw distance
OpenGL creates MipMaps for Texture LOD
Client Viewer (Mesh Enabled) Matches the mesh LOD with the texture MipMap
LOD Influence #2 – Client Viewer
Client viewer chooses best LOD based on the camera view to display on screen
Switching Zone Medium to Low LOD
Far Camera
Switching Zone High to Medium LOD
Middle Distance Camera
Near Camera
Camera Distance from Object (impacts the size of the object's image on display screen)
Closer objects will be displayed at a higher LOD setting than distant objects
High LOD version of Model
Medium LOD version of Model
Low LOD version of Model
The LOD is affected by the complexity of geometry and texture size used.
"Best" LOD Physics Mesh chosen during Upload Process
Asset Server / Sim Server or Cache
LOD Influence #3 – Server Feedback with System

FIGURE 7.1 Level of detail (LOD), a systems overview for the virtual world environment.

- How mesh geometry and graphic/texture size impact the success of an LOD system
- How LOD relates to the physics qualities of an object
- How a designer can get creative within the limitations of LOD

At the end of this chapter you will utilize some of the 3D content provided with this book to do your own LOD testing for the virtual environment we are building. We will explore how various LOD settings for physics and for rendering will impact how the content appears and interacts with the player in your game-based setting.

7.2 FUNDAMENTALS OF LOD SYSTEMS FOR DESIGNERS

Like the problem of rendering too many polygons in a scene, the topic of LOD systems is difficult to squeeze into one chapter. The computational math and psychological studies that support the current LOD processes in our virtual worlds are constantly under research and development. In this section, we will take a high-altitude overview regarding aspects of the LOD systems that impact your work as a designer of virtual environments and game-based sims. At the end of this book, you will find several listings in the bibliography containing supportive information in greater detail.

7.2.1 The Beginnings of LOD Systems

In 1976, Dr. James H. Clark (University of California at Santa Cruz) published an academic paper titled "Hierarchical Geometric Models for Visible Surface Algorithms" [1]. This is often cited as the unifying document that pulled together lots of previous concepts about computer graphics representation, which Clark calls "multiple levels of description" and created the initial concept that is the foundation for our current LOD systems. In his words:

> As an example of such a description, consider a model of the human body. When viewed at a very large distance, for example when the body covers only 3 or 4 display raster units, it is sufficient to model the body with a single rectangular polyhedron with appropriate color. Therefore the uppermost node, or "object," for this body represents this simple description. If the body is viewed from a closer distance—for example, if its topmost node's description covers 16 raster units—then this topmost description is no longer sufficient, and the next level of more refined description is needed. At this next level the body is now perhaps described as a collection of rectangular polyhedra appropriately attached to each other, for example using one polyhedron for each of the arms and legs, the head and the torso. Then so long as each of these "objects" covers only a few raster units of the display, their description is "sufficient." When the viewing distance decreases such that any of them covers a critical maximum area of the display, its more detailed sub-objects are used to replace its description. This process is carried out to whatever maximum level of detail will be needed. For example, a terminal level of description of the fingertip might be several surface patches (which could be implicitly structured even more finely using Catmull's algorithm). The body described is just one "object" of an environment, or larger hierarchy. There might be many such bodies, or other objects. The significant point, however, is that in a complex environment, the amount of information presented about the various objects in the environment varies according to the fraction of the field of view occupied by these objects. [1]

There it is. That is the definition of a foundational concept for organizing your screen graphics to maximize the utilization of the computer graphics display system.

7.2.2 A Brief Overview of the Kinds of LOD Used in Virtual Worlds

Graphics system developers, computer architecture designers, and programmers are always improving the capacity of a computer to display visual and audible detail. Clark's initial idea of a "visual hierarchy" and how an organized display of increasingly more detailed models could be guided by algorithms, which are essentially a set of "instructions" [2] for the computer to follow, has advanced into an entire science containing interrelated fields of study that includes optics, topology, psychology, geometry, and engineering. Algorithms have been created that squash, switch, and reconfigure the polygons of a 3D model as it changes position on the display screen. Algorithms that compress and expand the color information on the screen or shift the resolution of graphics/textures are also part of LOD science. Algorithms help us see increasing detail in the ground surface as we swoop down on it in a flight simulator or seamlessly transition from a flat silhouette of a distant avatar (sometimes called an "imposter") into the highly detailed representation of a virtual face. LOD manipulation also happens before the 3D content is uploaded and displayed in a virtual world. Content developers and 3D modeling software coders have devised techniques like "render to texture," "baking," and LOD tools for optimization to create a range of 3D models that represent that hierarchy of simplification necessary for the effective presentation of a highly detailed scene that will run in real-time.

According to David Luebke and colleagues, in their book *Level of Detail for 3D Graphics*, level of detail practices can start with such basic things as color rendition and the choice of what numerical system (fixed point vs. floating point) to use for the positioning of vertices.

> Some simplifications are so woven into the practice of computer graphics that we tend to forget about them, such as the quantization of color to 24 bits or the use of three-color channels (red, green, and blue) to represent the spectral response of the virtual scene. Even storing a polygonal mesh may involve simplification, since we clamp the precision of the coordinates and normals (typically to 32 bits each for X, Y, and Z). Although useful and long-used approximations, especially in interactive graphics, the quantization of color and geometric attributes is indeed simplification, and recognizing this can lead to many benefits. For example, game programmers once commonly optimized storage and computation by using fixed-point arithmetic and fewer bits of precision for vertex coordinates, and Michael Deering's seminal work on geometry compression introduced many ways of controlling precision to optimize storage and transmission of 3D models. [3]

Luebke et al. continue as they describe the various shading and mapping techniques that have been utilized in computer graphics for gaming. They go on to say,

> Shading and illumination form another spectrum of simplification in computer graphics. Many packages let the user control rendering performance with the familiar progression from wireframe to flat shading, to Gouraud shading, to texture mapping, and finally to per-pixel effects, such as Phong shading, bump mapping, and environment mapping. Lighting calculations range from simple local illumination techniques, such as the Phong model, to sophisticated global illumination methods such as radiosity and path tracing. Designers have long recognized the possibility of managing rendering performance by varying the complexity of shading and illumination applied to objects within the scene. The connection between texture mapping and level of detail deserves particular mention. One can think of MIP mapping and other texture filtering techniques as a form of LOD management. For example, Dumont et al. manage texture memory (and hence rendering performance) with a perceptually motivated metric by varying the maximum-resolution MIP map used. A more frequent technique, however, replaces geometric detail in LODs with texture maps. Termed imposters by Maciel and Shirley, these textured LODs are especially common in video games and visual simulation applications. [4]

7.2.2.1 LOD and Sound in a Virtual Environment

Visuals are not the only aspect of a game-based environment. The audio aspects of a virtual environment also make demands on the server and client viewer, and therefore can be considered eligible for LOD processes. In their review "3D Virtual Worlds and the Metaverse: Current Status and Future Possibilities," John David N. Dionisio, Will Burns, and Richard Gilbert comment on the rich and varied aural experience of virtual worlds and speculate on some of the audio design and manipulation systems that are under development. They say,

> In the same way that modern visuals cannot rely solely on predrawn textures for detail, one can say that the ambient sound environment should also explore the dynamic synthesis of sound effects—a computational foley artist, so to speak. Work on precisely this type of dynamic sound synthesis, based on factors ranging from rigid body interactions, such as collisions, fractures, scratching, or sliding, to adjustments based on the rendered texture of an object, exists in the literature. Synthesis of fluid sounds is also of particular interest, as bodies of water and other liquids are sometimes prominent components of a virtual environment. These techniques have not, however, seen much implementation in the virtual world milieu. Imagine the full effect of a virtual world audio environment where every subtle interaction of avatars and objects produces a reasonably unique and believable sound, which is then rendered, through real-time sound propagation algorithms and binaural synthesis, in a manner that accurately reflects its spatial relationship with the user's avatar. This research direction for realistic sound within the Metaverse has begun to produce results and constitutes a desirable endpoint for truly immersive audio. [5]

The use of LOD with sound is being explored with interactive "audiographic" objects, or objects producing sound and graphic effects. These are objects that share processing and are synched in time and space. They can be added into the virtual environment to create soundscapes or "topophonies" [6]. Diemo Schwarz, Roland Cahen, François Brument, Hui Ding, and Christian Jacquemin write in more detail about these methods of creating sound LOD in their 2011 paper "Sound Level of Detail in Interactive Audiographic 3D Scenes." They discuss how a bimodal framework that includes these audiographic objects and three defined levels of detail for sound (individual events, statistical texture, and background din) will be created and experienced throughout the virtual landscape. For a designer, being able to create with this kind of granular control over the sound in the virtual environment is a great step toward more immersive experiences with fewer gaps and stutters in the audio ambiance for the player.

7.2.2.2 LOD Systems for Polygons

Although not all models made for 3D graphics are made from polygons, the standard building component for 3D content in virtual worlds and gaming platforms is polygons and the triangles that create them. Since the initial concept of utilizing a progressive level of detail was posited by Dr. Clark in 1976, several kinds of LOD systems have been developed for use in computer graphics. Here are the most common types, a short description of how they work, and where they are most often utilized.

- Discrete (DLOD or just LOD) is an LOD system that creates detail transitions on the view screen in response to the camera distance from an object by shifting from one model resolution level to another greater or lesser model resolution. These model resolutions are premade with the 3D modeling software or generated as the model is loaded into the virtual environment. To prevent "popping" or a sudden shift when one resolution is changed out for the next, the system will utilize alpha blending or a transitional fade between two resolutions of the model as the camera distance changes the view.

- Continuous (CLOD) is an LOD system that constantly evaluates the scene and updates the level of detail based on a vertex database it creates for each object. The amount of detail rendered is undergoing constant transitions as the system extracts the level of detail data from the original model, collapsing or expanding the vertices from the mesh object's database to change the level of detail.
- View-dependent LOD is an LOD system that builds on the CLOD system by constantly changing the level of detail based on the database and the camera view. This system allows for multiple levels of detail in one scene, the nearby content will be highly detailed, and the distant content will be much less so. Large models such as terrain display benefit from this kind of LOD system.

When queried about the LOD system in Second Life, Scott Lawrence (Oz Linden) described it as such: "Second Life uses LOD (Level of Detail) and alpha blending to continuously provide a higher level of detail on the content as it streams to the viewer" [7].

7.2.2.3 LOD Systems for Graphics/Textures: MIP Mapping

Along with the polygons, the LOD system is processing your graphics to provide greater performance and quality in the onscreen images. Large graphics/textures use up considerable memory space, so a method called MIP mapping (or mipmapping) is employed by your computer to give your rendering engine some options. In the creation of a mipmap, the original graphic/texture will be used to create a whole "family" of graphic/textures, each image is a scaled down version of the initial one, as shown in Figure 7.2. The first part of the name "MIP map" comes from the Latin phrase *Multum in Parvo* ("much in little," or "much in a small space"), and as you can imagine from Figure 7.2, once the family of reduced images is created, they can be packed very tightly in memory [8]. Some of the initial work on the concept of mipmapping was done by Dr. Lance Williams at the New York Institute of Technology. In his 1983 paper "Pyramidal Parametrics" he describes the process of creating MIP maps and how they benefit graphics processing [9]. MIP maps are automatically generated by the OpenGL engine connected to the Graphics Processing Unit in your computer, and are used to help increase the rendering speed and frame rate of your game or virtual scene. Virtual worlds use the client viewer to set a "discard" level for these compressed images and only render the level of resolution needed for the camera's distance from the object being viewed. To quote Scott Lawrence again, who was kind enough to research this for me at Linden Labs, "LOD are selected by the viewer as the camera distance changes from the object. Low resolution levels are loaded first when you enter a scene, which is why you see hazy textures at times."

7.2.2.4 LOD Systems Using Culling, Draw Distance, and Fog

Other ways that the level of detail can be lowered and processing speed improved is by introducing culling and a draw distance limit for the camera. Culling will remove any object from the scene that is past a certain distance from the camera's position, and it will also remove any surfaces not visible to the camera from the rendering list. The draw distance can be set by the user, allowing them to remove visual eyesores from their view, as well as tailor the amount of detail their computer will have to calculate for each frame [10]. Fog is another means for reducing the amount of level of detail that will be rendered in a scene. Having the virtual world system create a foggy background will lower the demand on the graphic rendering system and client viewer. Although you see less and less of this as the overall level of graphics cards improves, this technique has often been used in games that have large open spaces [11].

7.3 PERCEPTION AND LOD: THE BASICS

The perception of detail by the human visual system is a complex process involving not only the actual detail of the object as it is being represented on the screen, but the ways in which our visual system handles

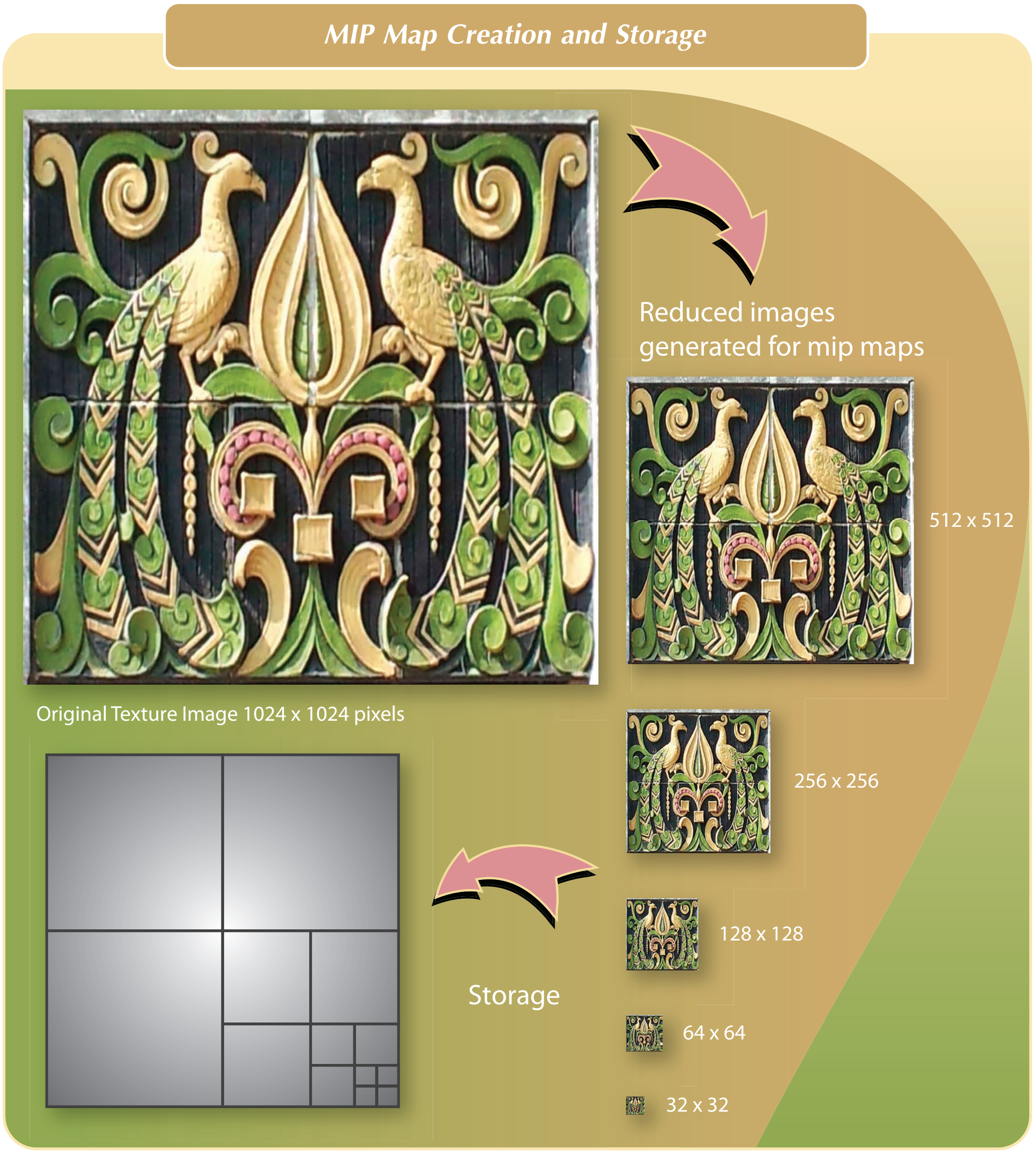

MIP Map Creation and Storage
Reduced images generated for mip maps
512 x 512
Original Texture Image 1024 x 1024 pixels
256 x 256
128 x 128
Storage
64 x 64
32 x 32

FIGURE 7.2 MIP map creation and storage structure in the OpenGL engine.

the information being gathered by our eyes, and how effectively our eyes can gather that information. In Figure 7.3 is an overview of the major factors that affect the perception of objects in our visual field. With an understanding of how images, our visual cortex and other parts of the visual system combine to create the sensation of depth and level of detail, you as a designer, can tailor your virtual environments to maximize the LOD effects. This diagram is based on notes taken from *Level of Detail for 3D Graphics* by Luebke et al. In *Level of Detail for 3D Graphics*, the authors explain how eye structure, contrast in the scene, and object movement affects our perception of level of detail, but for the purposes of this book we will just consider some of the major aspects of perception and how it relates to design and LOD.

7.3.1 General Aspects about Our Visual Perception and How We See LOD

Following are some of the key aspects about visual perception and how it pertains to the ways we perceive LOD. By raising your awareness of these factors, leveraging that knowledge, and implementing level of detail enhancements in the virtual environments you design, you can increase the level of detail that reaches observing eyes and deepen the experience of the visitor.

- In our eyes, the threshold of our visual system, we use the fovea of our eye to gather an image of the highest level of detail. The rest of the visual field is processing a less detailed image, and near the edges of our visual field, in our peripheral vision, we are more sensitive to motion than detail [12].
- Parallax or the movement of near and distant objects in our binocular visual field allows us to perceive depth in the scene. Movement of our point of view and the changing parallax it creates allows us to see how various objects are arranged in space [13].
- Depth of field (DOF) and the blurring of distant objects as we change the focus of our eyes on near and distant things is one of the fundamental aspects of our visual perception [14].
- Contrast sensitivity, or the perception of dark and light areas on the surface of an object, is another major aspect of LOD perception [15].

Many factors can affect how detail is seen in a virtual environment. Essentially these factors can be grouped into two major categories: environmental factors and individual factors. Environmental factors that affect our visual perception of LOD include (1) background illumination of the scene on the screen, (2) light adaption or how well our eyes have adjusted to changes in the surrounding illumination, (3) duration of the stimulus and its effect on the contrast we see in the surrounding scene, (4) display settings on the screen we are viewing, and (5) the interaction with other senses, such as sound, which can affect what we think we see. Individual factors that affect visual perception of LOD can include (1) the age of the visitor and their sensitivity to contrast in the visual field, (2) their color vision acuity, (3) their stereo vision acuity or depth perception acuity, (4) their actual visual acuity (pertaining to nearsightedness or a need for corrective lenses), (5) their emotional circumstances and the effect that has on their pupil size, and (6) their previous experience or memory of the object being viewed, which can enhance the quality of the image perceived [16]. As you can see in Figure 7.3, the relationship of these factors is organized to illustrate how interconnected our brain, visual cortex, visual memory, and proprioceptive senses are. Even though they may only be directly experiencing the virtual environment with their eyes and ears, there are many other factors that influence how they will visually process and remember the experience. On the most basic level, it is here where a designer can start making the informed decisions about what kind of environment best suits their needs and where the emphasis should be focused in the details.

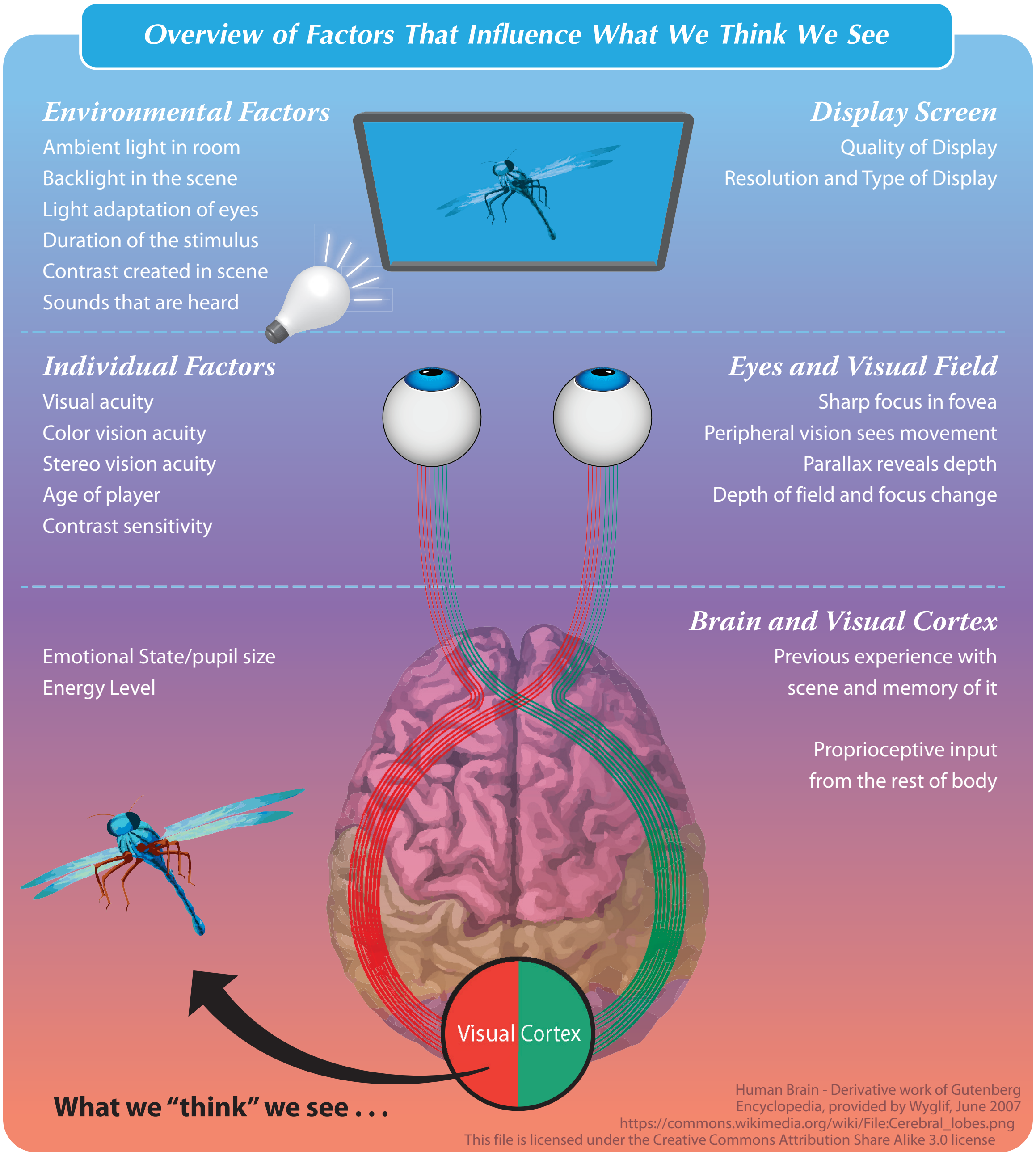

Overview of Factors That Influence What We Think We See
Environmental Factors
Ambient light in room
Backlight in the scene
Light adaptation of eyes
Duration of the stimulus
Contrast created in scene
Sounds that are heard
Display Screen
Quality of Display
Resolution and Type of Display
Individual Factors
Visual acuity
Color vision acuity
Stereo vision acuity
Age of player
Contrast sensitivity
Eyes and Visual Field
Sharp focus in fovea
Peripheral vision sees movement
Parallax reveals depth
Depth of field and focus change
Emotional State/pupil size
Energy Level
Brain and Visual Cortex
Previous experience with scene and memory of it
Proprioceptive input from the rest of body
Visual Cortex
What we "think" we see . . .
Human Brain - Derivative work of Gutenberg Encyclopedia, provided by Wyglif, June 2007
https://commons.wikimedia.org/wiki/File:Cerebral_lobes.png
This file is licensed under the Creative Commons Attribution Share Alike 3.0 license

FIGURE 7.3 Overview of human visual system and perception of level of detail.

7.4 MAKING LOD WORK FOR YOU

A good designer gathers information and tries to make wise choices about the project they are working on. In a perfect world where you would have unlimited time and resources to create a virtual environment, you would not have to decide how to balance the creation time with the quality of the visual output. But no designer has that luxury. Projects are always under tight deadlines and budgets. In this section, we will talk about how to make LOD work for you and give you information that will help you make the most advantageous decisions regarding your 3D models and the levels of detail you create with those models.

7.4.1 Geometry and Managing the Levels of Detail

Managing the levels of detail in your virtual world environment starts with three questions:

1. Where will the user's cameras most likely be located while they are visiting your virtual environment?
2. How fast will the user be traveling through your environment?
3. How many interactive areas or devices will be included in your virtual environment?

Let us look at the design challenge each of these questions implies.

For question 1, the most common answer would be that the user's camera is in the default position that is above and slightly behind the avatar's head, and is pointed slightly downward. As the designer of a virtual environment, you have the option to work within that parameter and locate most of your interesting details near the floor and midground areas, or you can include visual "guides" like vertical lines in the architecture, gradient color changes, or contrast changes that encourage the user to move their cameras up and look around, perhaps even enter into the "Mouselook" state.

Regarding question 2, if you are building a vehicle-based environment, full of users that will be driving or flying a vehicle, or chasing other users or objects within it, then the general level of detail in their surroundings may not need to be very high at all. In fact, they may not even be at any specific point long enough for the graphic/textures to load completely, so you will want to reduce the size and repeat them as often as possible to diminish the loading time overall.

And finally considering question 3, interactivity often needs a high level of detail. If you are using graphic/textures on the geometry that link up to external sources of information like a QR code pattern, or subdivide the side of a cube primitive into touch panel areas, then you need to provide for a larger resolution in the avatar's field of view. In this instance, the size of the graphic/texture must be balanced with the clarity of the image based on the avatar's camera position. You may need to do a few tests to find out what size graphic/texture will load quickly for most users, and still provide them with a clear image on their screen, thereby enhancing easy mouse usage while they click on the object.

If you are working with meshes and using an external modeler such as 3ds Max, Blender, or Maya, you will want these meshes to look their best in a virtual environment like Second Life or OpenSim. Find and learn about the LOD tools provided by the modeler you prefer. In Table 7.1 is a short list of the most common modelers used for creating meshes in virtual environments and some information about their LOD tools.

7.4.2 Graphic/Textures and LOD

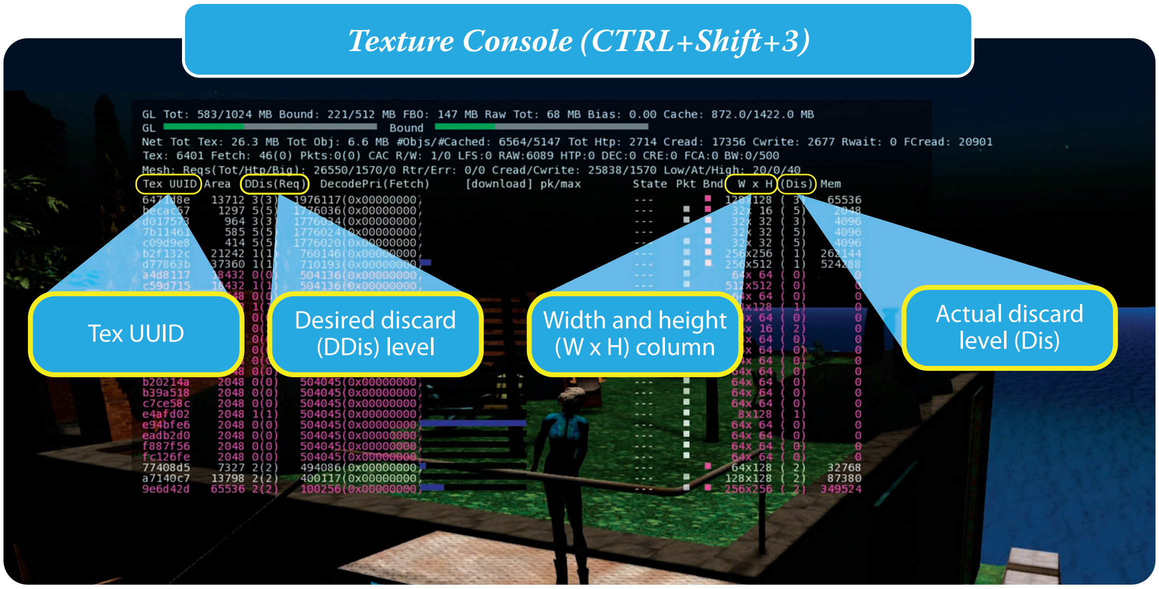
The first thing to understand is how hard the client viewer and server are working to present the highest resolution possible for each graphic/texture on your screen. If you activate the Texture Console (CTRL+Shift+3) in your client viewer, you will see a list of textures that are being displayed to create the scene. Take a look

TABLE 7.1
3D Platforms and Methods for Optimizing LOD with Suggested Links

3D Modeling Software or Platform	Method for Creating Models in LOD
3ds Max	Use segment modification to change LOD on the mesh; when used in combination with VRML LOD helper you can render out different images at various levels of detail
Blender	Use Blender LOD menu to create changes in segmentation in the Blender Game Renderer
Maya	Use Maya LOD groups menu to manage detail levels of an object
Unity	Use LOD optimization in Group Component menus
Mesh Enabled Client Viewer for Second Life and OpenSim	Use the mesh upload menu to set up how the model will be imported with various levels of LOD (see Section 7.5)

Source: "LOD or Level of Detail in 3dsMax Tutorial," YouTube video, 9:22, posted by Stereopixol, March 27, 2014, accessed October 30, 2014, https://www.youtube.com/watch?v=8G2sbjJ7c3k; "Blender/Game Engine/Tutorial - LOD (Level of Detail)," YouTube video, 3:37, posted by Blenderraiser, March 7, 2014, accessed October 30, 2014, http://youtu.be/zp9mA1gK8sQ; Nicholas H. Battjes, "How to Use LOD (Level of Detail) Groups in Maya," *Wonder How to.com*, 2010, accessed October 30, 2014, http://autodesk-maya.wonderhowto.com/how-to/use-lod-level-detail-groups-maya-376511/; "Level of Detail," *Unity Manual*, accessed October 30, 2014, http://docs.unity3d.com/Manual/LevelOfDetail.html.

at the line starting with "Tex UUID" just over the cascading list of textures. The proceeding headings over the texture console display are highlighted in Figure 7.4. As you read to the right, you will learn what is the desired discard (DDis) level for each specific texture. That is the highest level of resolution that will be loaded for that specific texture, and it is determined by how many pixels that texture is filling on your screen surface. A discard level of 0 indicates a texture that will be loaded at its highest resolution (let us assume 512 × 512),


Texture Console (CTRL+Shift+3)
Tex UUID
Desired discard (DDis) level
Width and height (W x H) column
Actual discard level (Dis)

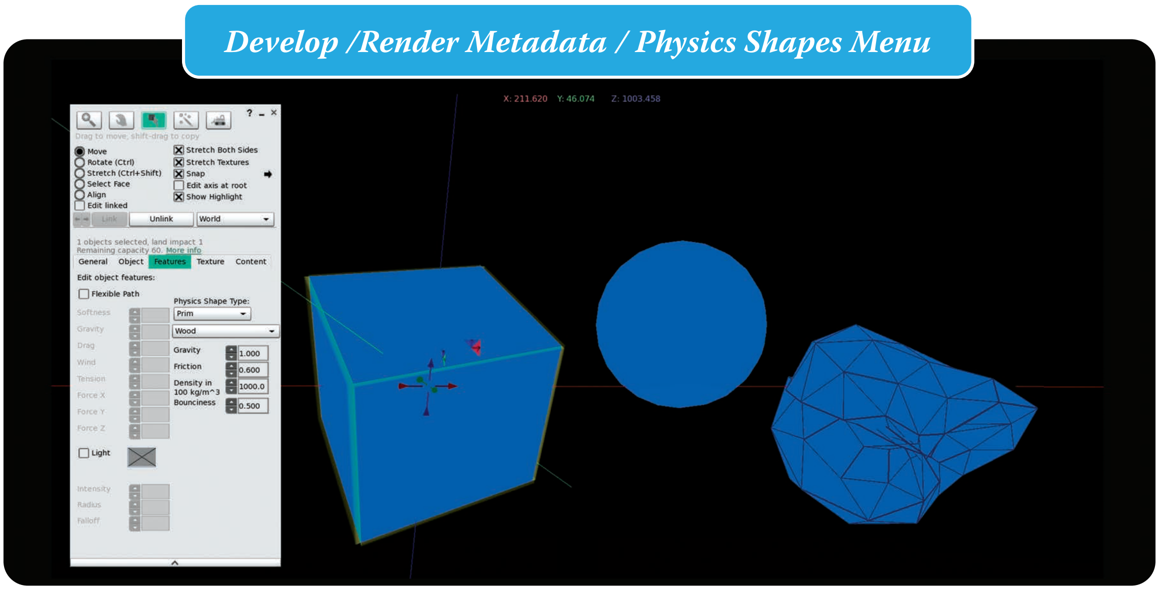
```
GL Tot: 583/1024 MB Bound: 221/512 MB FBO: 147 MB Raw Tot: 68 MB Bias: 0.00 Cache: 872.0/1422.0 MB
GL                                  Bound
Net Tot Tex: 26.3 MB Tot Obj: 6.6 MB #Objs/#Cached: 6564/5147 Tot Htp: 2714 Cread: 17356 Cwrite: 2677 Rwait: 0 FCread: 20901
Tex: 6401 Fetch: 46(0) Pkts:0(0) CAC R/W: 1/0 LFS:0 RAW:6089 HTP:0 DEC:0 CRE:0 FCA:0 BW:0/500
Mesh: Reqs(Tot/Htp/Big): 26550/1570/0 Rtr/Err: 0/0 Cread/Cwrite: 25838/1570 Low/At/High: 20/0/40
Tex UUID Area  DDis(Req)  DecodePri(Fetch)     [download] pk/max        State Pkt Bnd  W x H (Dis) Mem
6471d8e   13712 3(3)  1976117(0x00000000)                      ---             128x128 ( 3)   65536
becac57    1297 5(5)  1776036(0x00000000)                      ---              32x 16 ( 5)    2048
d017573     964 3(3)  1776034(0x00000000)                      ---              32x 32 ( 3)    4096
7b11461     585 5(5)  1776024(0x00000000)                      ---              32x 32 ( 5)    4096
c09d9e8     414 5(5)  1776020(0x00000000)                      ---              32x 32 ( 5)    4096
b2f132c   21242 1(1)   760146(0x00000000)                      ---             256x256 ( 1)  262144
d77863b   37360 1(1)   710193(0x00000000)                      ---             256x512 ( 1)  524288
a4d8117   18432 0(0)   504136(0x00000000)                      ---              64x 64 ( 0)       0
c59d715   18432 1(1)   504136(0x00000000)                      ---             512x512 ( 0)       0
b20214a    2048 0(0)   504045(0x00000000)                      ---              64x 64 ( 0)       0
b39a518    2048 0(0)   504045(0x00000000)                      ---              64x 64 ( 0)       0
c7ce58c    2048 0(0)   504045(0x00000000)                      ---              64x 64 ( 0)       0
e4afd02    2048 1(1)   504045(0x00000000)                      ---               8x128 ( 1)       0
e94bfe6    2048 0(0)   504045(0x00000000)                      ---              64x 64 ( 0)       0
eadb2d0    2048 0(0)   504045(0x00000000)                      ---              64x 64 ( 0)       0
f887f56    2048 0(0)   504045(0x00000000)                      ---              64x 64 ( 0)       0
fc126fe    2048 0(0)   504045(0x00000000)                      ---              64x 64 ( 0)       0
77408d5    7327 2(2)   494086(0x00000000)                      ---              64x128 ( 2)   32768
a7140c7   13798 2(2)   400117(0x00000000)                      ---             128x128 ( 2)   87380
9e6d42d   65536 2(2)   100256(0x00000000)                      ---             256x256 ( 2)  349524
```

FIGURE 7.4 Important informational columns in the texture console of the client viewer.

discard level 1 is the next level down in resolution or one half in each dimension (256 × 256), discard 2 is a smaller one, and so forth. If you continue reading to the right, you will see a width and height (W × H) column indicating the actual resolution of the image. This is a good place to look if you think the texture load in a scene is slowing things down. If that is the case, you may see a large number of textures that are 1024 × 1024 in dimension. If these are your textures, and you have a laggy response, try reducing as many textures as possible to 512 × 512. Next to the W × H column is the actual discard level (Dis). Eventually, as the client viewer loads in the appropriate resolution for each texture the camera is seeing, the numbers listed on this will match the desired discard numbers (DDis) in the left-side column. Once the client viewer has streamed the whole environment to you, on the screen you should see clear, nonfuzzy versions of each texture that are within your camera's draw distance. There is much more information about this texture console and the information it displays is available at http://wiki.secondlife.com/wiki/Texture_Console.

7.4.3 Physics and LOD

The virtual world would be little more than a display case of 3D models if we did not have the capacity to calculate physical reactions between avatars, objects, and the virtual environment they are in. It is essential to creating believable doorways, drivable vehicles, navigable streets, and so forth, and that job is done by the physics engine. In order to calculate the physical properties of each 3D object, a physics shape is applied to each object in the virtual environment. You can see them, if you activate "Physics shapes" under the Develop/Render Metadata menu. In Figure 7.5 is a screengrab showing the physics shapes generated for objects in a scene; note the black lines outlining the faces on each shape. It is very important for performance reasons to have the simplest physics shape possible surrounding your 3D models. For example, a wall without any openings that the avatars need to pass through can be covered with a simple box physics shape, but a torus with an opening in the middle that serves as the entrance to your hobbit house will need a



[● Move] Rotate (Ctrl) Stretch (Ctrl+Shift) Select Face Align
☐ Edit linked
☒ Stretch Both Sides
☒ Stretch Textures
☒ Snap
☐ Edit axis at root
☒ Show Highlight

1 objects selected, land impact 1
Remaining capacity 60. More info

Edit object features:
☐ Flexible Path
Physics Shape Type: Prim
Wood
Gravity 1.000
Friction 0.600
Density in 100 kg/m^3 1000.0
Bounciness 0.500
☐ Light

FIGURE 7.5 Screengrab of showing physics shape display, with a cone, box, and slightly twisted sphere.

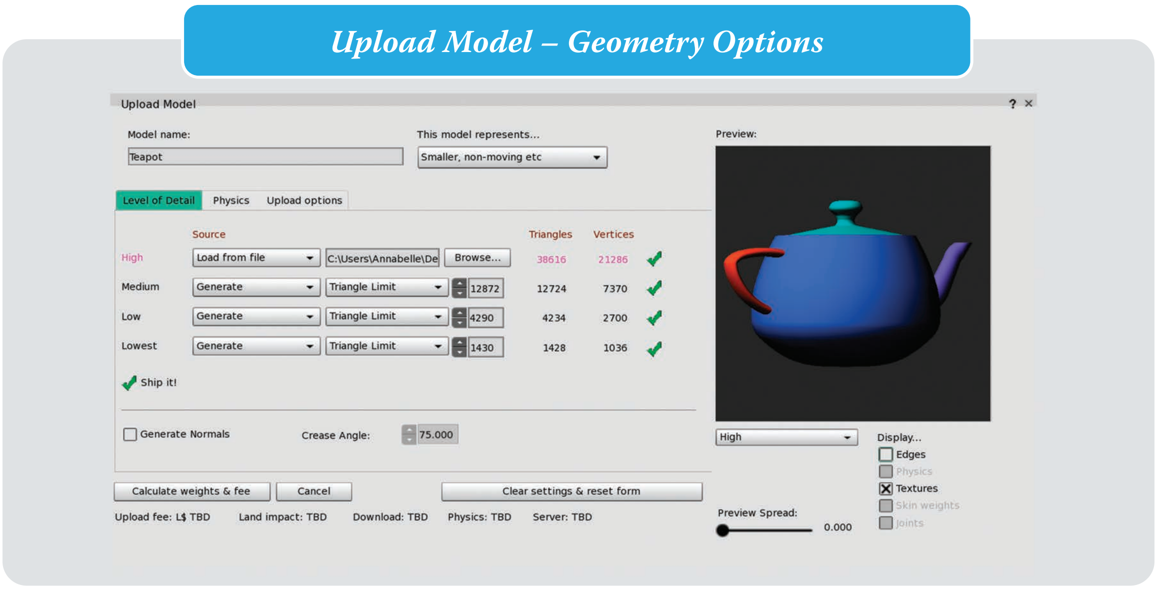
much more complex physics shape and demand more calculations from the physics engine. There are some ways to simplify the demands on your physics and increase the responsiveness of your regions. If you are in Second Life and working with the Havok engine you will find some tips for that at http://wiki.secondlife.com/wiki/Physics_Optimization. If you are in OpenSim and working with the Bullet Sim Physics engine, there are some guidelines as to how the engine is being implemented at http://opensimulator.org/wiki/BulletSim/Functionality. In general, you should be looking for ways to remove physics calculations from as many objects in the scene as possible. Objects that are not in contact or collided with, children in a linkset, or objects created from sculpties, can all be either removed from physics calculations or given a simpler box or cylinder physics shape. Finding the right physics shape for your mesh requires a bit of trial and error, but the results are worth it.

7.5 CLIENT VIEWER SETTINGS, IMPORTS, AND LOD

All of the projects in this book will be using 3D models created in Blender. Understanding how the LOD can be manipulated while they are imported into your game-based sim is crucial to providing yourself with the best performance possible in the regions. In the next section we will look at the model LOD and the physics LOD options that you have in the importation menu of the client viewer. This is where the knowledge about the LOD tools specific to your preferred 3D modeling software will pay off.

7.5.1 UPLOADING GEOMETRY WITH LOD

Let us assume that you have created a perfect 3D model of your favorite teapot and want to make a house out of it in your virtual environment. Using the current upload menu provided by the client viewer will present you with some interesting choices. In Figure 7.6, you will see the buttons for these choices highlighted on the



Model name: Teapot

This model represents... Smaller, non-moving etc

	Source			Triangles	Vertices	
High	Load from file	C:\Users\Annabelle\De		38616	21286	✓
Medium	Generate	Triangle Limit	12872	12724	7370	✓
Low	Generate	Triangle Limit	4290	4234	2700	✓
Lowest	Generate	Triangle Limit	1430	1428	1036	✓

✓ Ship it!

Generate Normals Crease Angle: 75.000

Upload fee: L$ TBD Land impact: TBD Download: TBD Physics: TBD Server: TBD

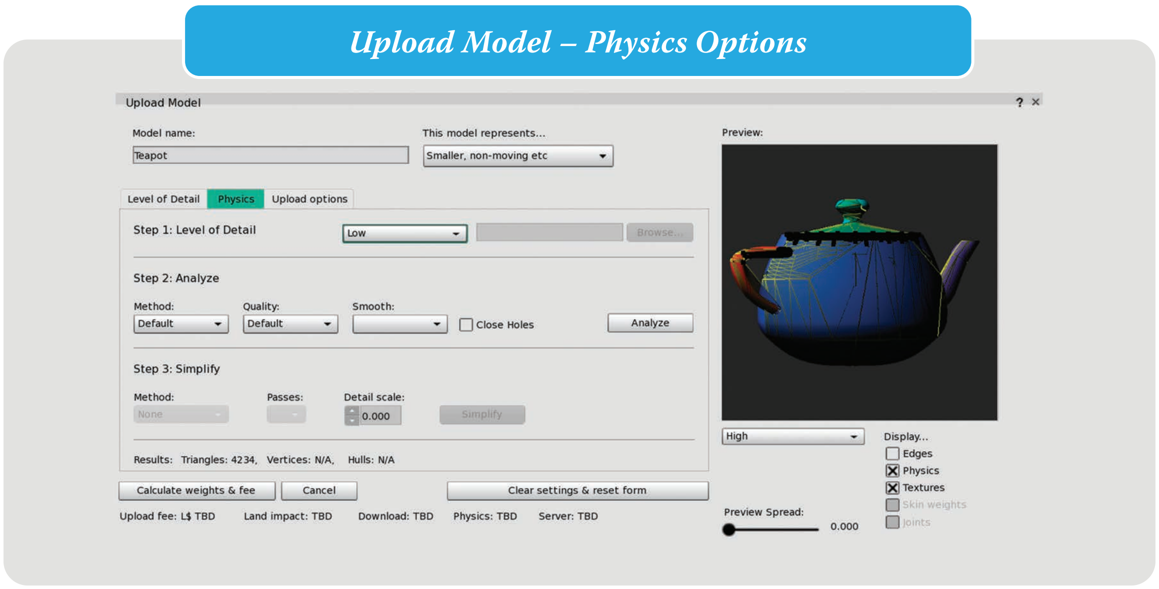
High Preview Spread: 0.000

FIGURE 7.6 Mesh model upload menu, geometry options.

first page of the upload menu. You can follow one of these two workflows: (1) you build the teapot model and create the levels of detail for it in your 3D modeling system (3ds Max, Blender, etc.), choosing Load from File in the drop down boxes as you upload the model; or (2) you make the perfect teapot in your 3D modeler, and use the Generate option on the client viewer upload menu and let the automatic algorithms create the levels of detail as you upload the model into the virtual environment. Either way is valid, and your personal preference combined with the needs of the project will dictate what you utilize. You may find that by using the Generate function in the upload menu, you will have a better sense of how to create LOD files with your 3D Modeler for export into a wide variety of environments. A good rule of thumb is to decrease the Triangle Limit settings in the Generate functions by 50% to 75% as you step down toward the lowest resolution model [17]. You can also smooth or sharpen the look of the teapot model by changing the Crease Angle on the normals as you upload it. Lower numbers will sharpen the edges, and higher numbers will smooth it.

7.5.2 Uploading 3D Models and Setting the Physics LOD

As you upload your teapot model, you also have some upload options for the physics level of detail. Bear in mind that the upload menu for the client viewer was created for the Havok engine in Second Life, and if you are working in OpenSim with the Bullet Sim physics engine, you will not have as many options for setting up the physics shape. In that case, you may want to create a physics shape from a very low resolution, low polygon model that you create in your 3D modeler. Once you have set up your geometry LOD hierarchy on the first tab, go to the next tab and look at the possibilities offered. In Figure 7.7, you will see that they offer you three steps for setting up the physics shape around your model. Generally, you should set up the



Model name: Teapot

This model represents... Smaller, non-moving etc

Step 1: Level of Detail: Low

Step 2: Analyze

Method: Default Quality: Default Smooth: ☐ Close Holes

Step 3: Simplify

Method: None Passes: Detail scale: 0.000

Results: Triangles: 4234, Vertices: N/A, Hulls: N/A

Upload fee: L$ TBD Land impact: TBD Download: TBD Physics: TBD Server: TBD

High

☐ Edges
☒ Physics
☒ Textures
☐ Skin weights
☐ Joints

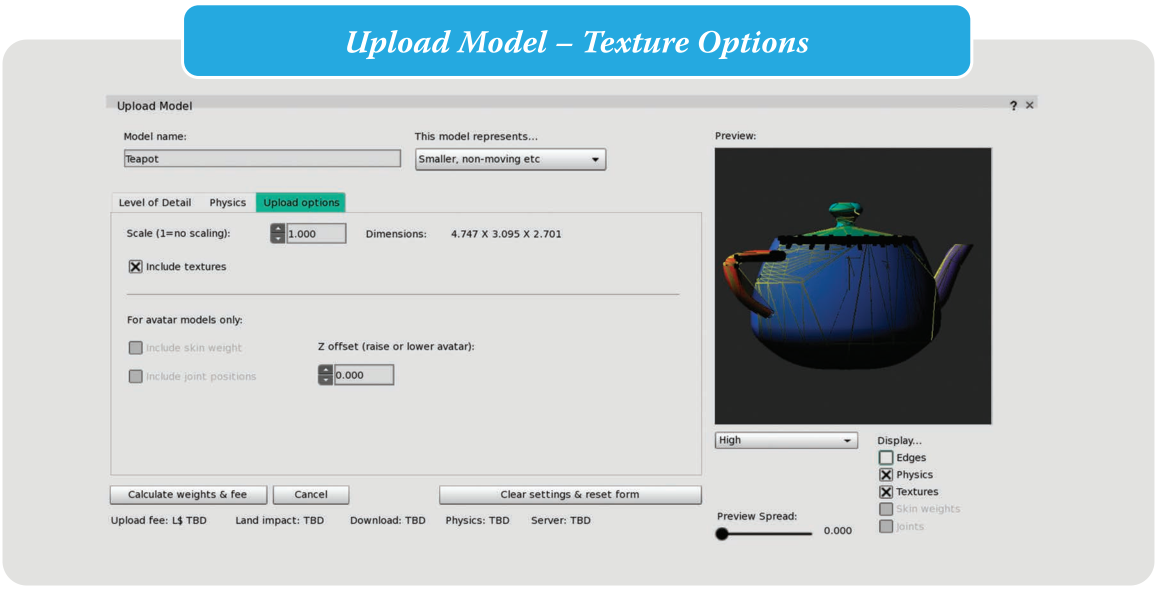
Preview Spread: 0.000

FIGURE 7.7 Mesh model upload menu, second tab for physics shapes.

lowest possible LOD in step 1, either from the model file if the mesh is a simple shape, a simplified version of that model if it is more complex, or from one of the generated level of details created in the first tab, the geometry upload menu page. Tick the Physics box under the preview window to see how complex the physics shape will be. If you do not have a physics shape of great complexity, you can probably skip steps 2 and 3. However, if you do see lots of crossing planes and even some red lines when you try to upload it, then try the Analyze button (step 2) and the Simplify button (step 3) to reduce the complexity of the physics shape. There is much more detail about how to manipulate these settings at http://wiki.secondlife.com/wiki/Mesh/Upload_Model_UI_reference.

7.5.3 Uploading 3D Models with Scaling and Graphic/Texture Parameters

On the third tab of the upload menu are options for scaling your model, including graphics/textures and adjustments for an avatar upload. Let us focus on the scaling and texture parameters. Notice the highlighted settings on the menu in Figure 7.8. Regarding the texture settings, if you are making your own LOD models for the uploader, make sure that each level of the model has the same graphic/texture as the highest resolution version. You may upload a model with up to eight materials, so if your highest resolution model has eight materials, then your lowest resolution model should also. The scaling spinner is relatively self-explanatory; just watch the numbers on the right to see what size the final uploaded model will be. There is more granular detail on all of these settings in this wiki: http://wiki.secondlife.com/wiki/Mesh_and_LOD.



Model name: Teapot

This model represents... Smaller, non-moving etc

Scale (1=no scaling): 1.000 Dimensions: 4.747 X 3.095 X 2.701

[✓ Include textures]

For avatar models only:

[ ] Include skin weight

[ ] Include joint positions

Z offset (raise or lower avatar): 0.000

High

[ ] Edges

[✓ Physics]

[✓ Textures]

[ ] Skin weights

[ ] Joints

Upload fee: L$ TBD Land impact: TBD Download: TBD Physics: TBD Server: TBD

Preview Spread: 0.000

FIGURE 7.8 Mesh model upload menu, third tab for texture options.

7.6 PROJECT: CREATING MESH CONTENT FROM PHOTOGRAMMETRY; WORKING WITH LOD

In this project you will learn a workflow method for making 3D meshes from photographs, a process called photogrammetry. There are many ways to do this process, and the marketplace is filled with many types of photogrammetry software that can help you, as noted in Chapter 10, Section 10.2.5. In this project, we will create a mesh with a relatively low-cost method, utilizing the following equipment and Internet resources, most of which you may already have access to

- A good digital camera with a fixed lens
- A tripod or monopod for it
- Some photo lights or a good location with lots of natural indirect light
- Software that will let you convert from formats like yourmodel.obj, yourmodel.fbx into yourmodel.dae (the COLLADA.dae format is used by virtual worlds for mesh uploads)
- A 2D paint/graphics processing program that will let you tweak the graphics captured by the photogrammetry process
- A good high-speed Internet connection, because you will be uploading lots of pictures to the cloud-based photogrammetry site called Autodesk Memento (http://memento.autodesk.com), which is currently in beta development

Although it may not need to be mentioned, you will want to have a good desktop computer or a high-performance laptop that allows you to process images quickly and easily, and has no problem with the graphics demands of virtual worlds. You will also need an Autodesk account, so you can download and access the cloud for Autodesk Memento project editing and exportation. I highly recommend that you watch the videos provided by Autodesk, especially the one called "Autodesk Memento Webinar #3 – It's All in the Photos." In that video, Craig Barr and Tatjana Dzambazova demonstrate and explain tips and best practices for taking photos for 3D reconstructions.

In this project there will be several opportunities to modify the LOD of the model you are making and these "LOD Decision Points" will be called out in the process.

OK, got your tools and toys together? Let us begin.

7.6.1 Capturing Meshes with Photogrammetry

The first thing is to set up for capturing the 3D form with your camera. You will want to make sure the lighting is bright enough, so that you can capture large digital images in the RAW or large JPG formats. You will want to use a tripod or monopod for the camera so the images stay as crisp and focused as possible. In Figure 7.9 is a SketchUp diagram of the studio setup used for this project (the models used to make this image: the camera, tripod, lights, and stands are from the SketchUp 3D Warehouse, https://3dwarehouse.sketchup.com/). Note that the top of the table is marked out like a 16-hour clock, and the object being photographed is on a turntable, so it can be rotated in even arcs of precise degrees for each sequential photo. Rotate the object 360 degrees, then change the camera angle, and repeat the same rotation. Change the angle two to four times so you are photographing the object from lower and higher angles. You will need to make 60 to 100 photos of the object and upload them to create a 3D mesh. Do not get frustrated if the photo stream

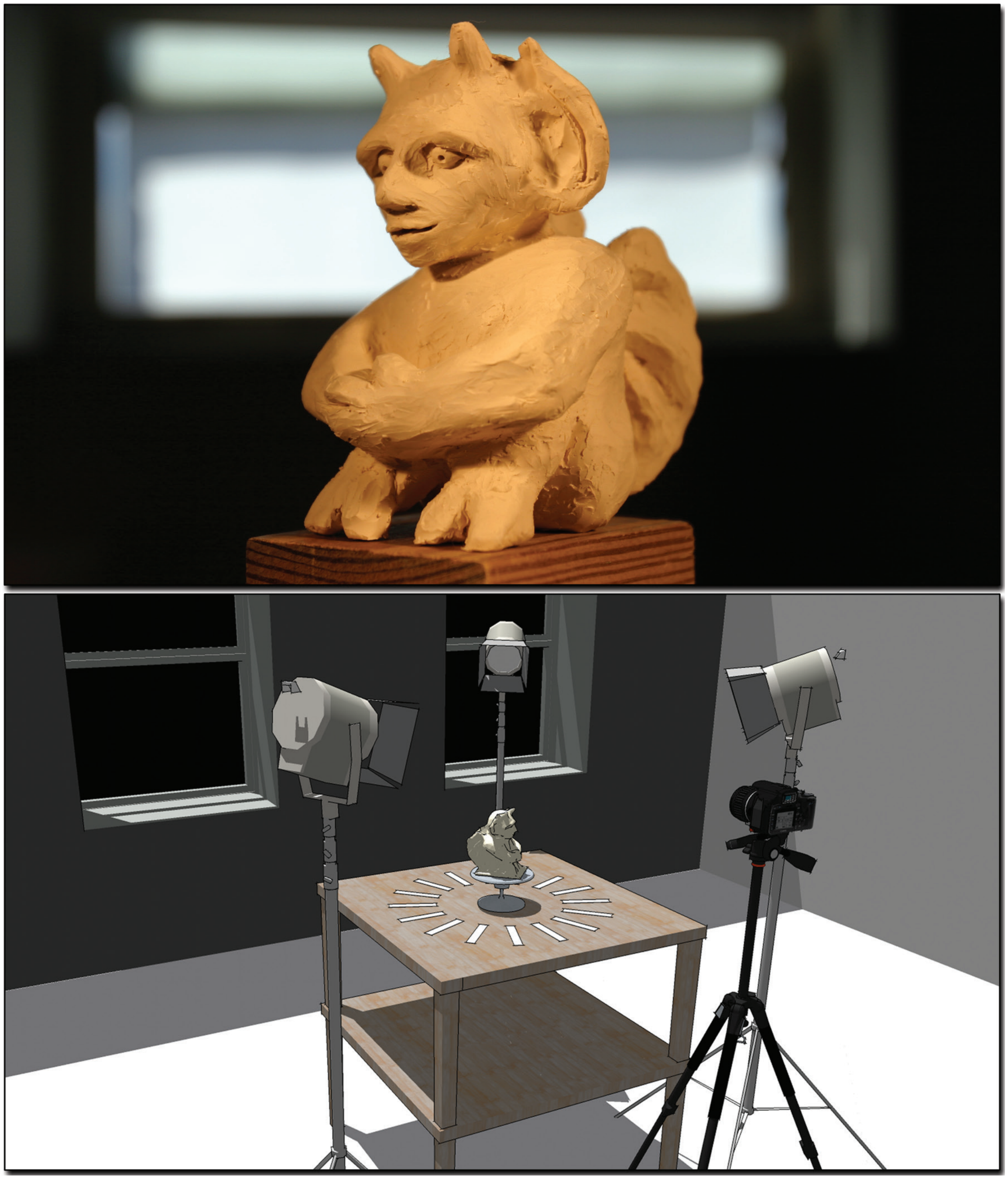
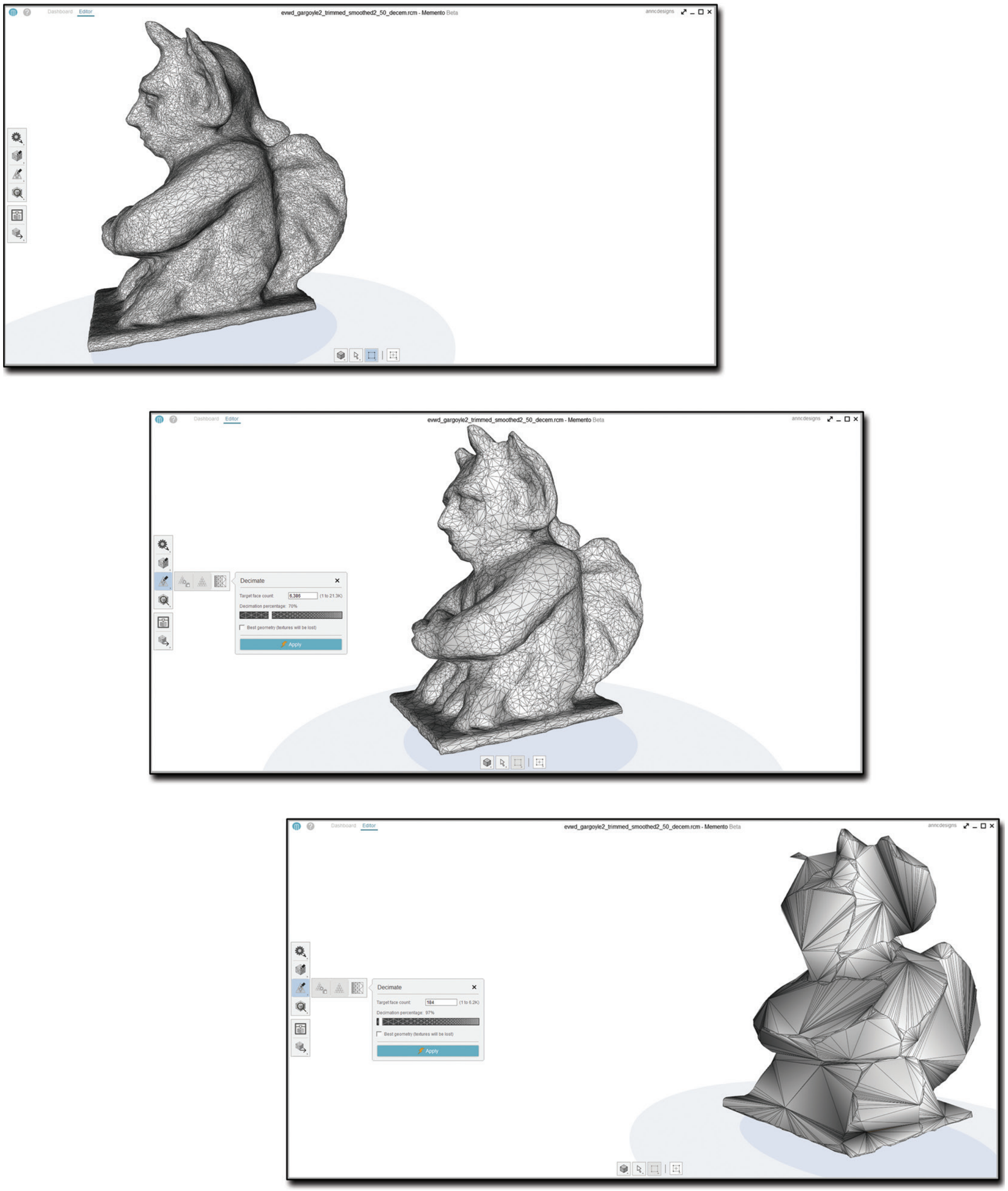


FIGURE 7.9 Studio setup for mesh capture with sample photo of the gargoyle sculpture created during the process. Studio equipment models are from the SketchUp 3D Warehouse.


Dashboard
Editor
evwd_gargoyle2_trimmed_smoothed2_50_decem.rcm - Memento Beta
anncdesigns
Decimate
Apply

Decimate
Target face count: 6,386 (1 to 21.3K)
Decimation percentage: 70%
Best geometry (textures will be lost)

Decimate
Target face count: 184 (1 to 6.2K)
Decimation percentage: 97%
Best geometry (textures will be lost)

FIGURE 7.10 Autodesk Memento interface showing decimation process on the gargoyle.

fails and the model does not start to process. If this happens and you think you did a good methodical job of photographing the model, look back through the images and see if one is out of sequence or out of focus, and delete it. Then try the Autodesk Memento upload process for the model again.

7.6.2 Reducing the LOD in Autodesk Memento

OK, your photos have been accepted by Autodesk Memento, it has processed your 3D model, and you have downloaded the yourmodel.rcm (.rcm stands for Autodesk ReCap Mesh) file to your computer. Hopefully, you got a nice clean model with very little extra "stuff" floating around it or attached to it. With the editing tools that Autodesk Memento provides, you can carve away the extra stuff, fill holes, and generally clean up the model. Just right and left click to access the menus, use the 3 buttons on your mouse to rotate, pan, or zoom on the model as you do this. Once you have gotten the model cleaned up and ready for export from Autodesk Memento, you need to consider the level of detail (LOD) you will want to have on this mesh once it reaches the virtual world. Obviously, an object with fewer faces will load faster and run better, but it may become quite distorted as you lower the LOD with the "decimation" editor.

LOD Decision Point 1: Geometry Level of Detail

In Figure 7.10 is a composite image showing the same gargoyle model undergoing various levels of decimation. As you can see, the final one on the far right has become full of large flat faces. Deciding on what level of detail the model should have when you export it for use in a virtual environment is kind of a rule of thumb process. The best thing to do is to save many versions of the yourmodel.rcm file as you go, so you can always backtrack. Once you have a model that looks good, created with the highest level of decimation possible, then export the model in the file formats you like to work with. For instance, you may like to use the yourmodel.obj format for viewing and cleaning up the Texture (diffuse) graphic in Photoshop, and the yourmodel.fbx format in 3ds Max for creating the yourmodel.dae file from. As you make virtual objects with photogrammetry, you will find the workflow that suits you and your equipment.

7.6.3 Converting the 3D and Graphics/Textures for Use in a Game-Based Sim

Once you have created the export versions of your model from Autodesk Memento, put them through your favorite modeling software to convert them to the COLLADA.dae format.

LOD Decision Point 2: Graphic/Texture Level of Detail

Open the graphic material that is saved with your model file and reduce the entire image to a resolution of 1024 pixels by 1024 pixels (or 512 pixels by 512 pixels). Also feel free to massage the graphic colors and contrast if you want. Save this graphic as a yourtexture.jpg file for the virtual world upload. In Figure 7.11 is a screengrab showing the process of preparing the Texture (diffuse) graphic for upload into a virtual world.

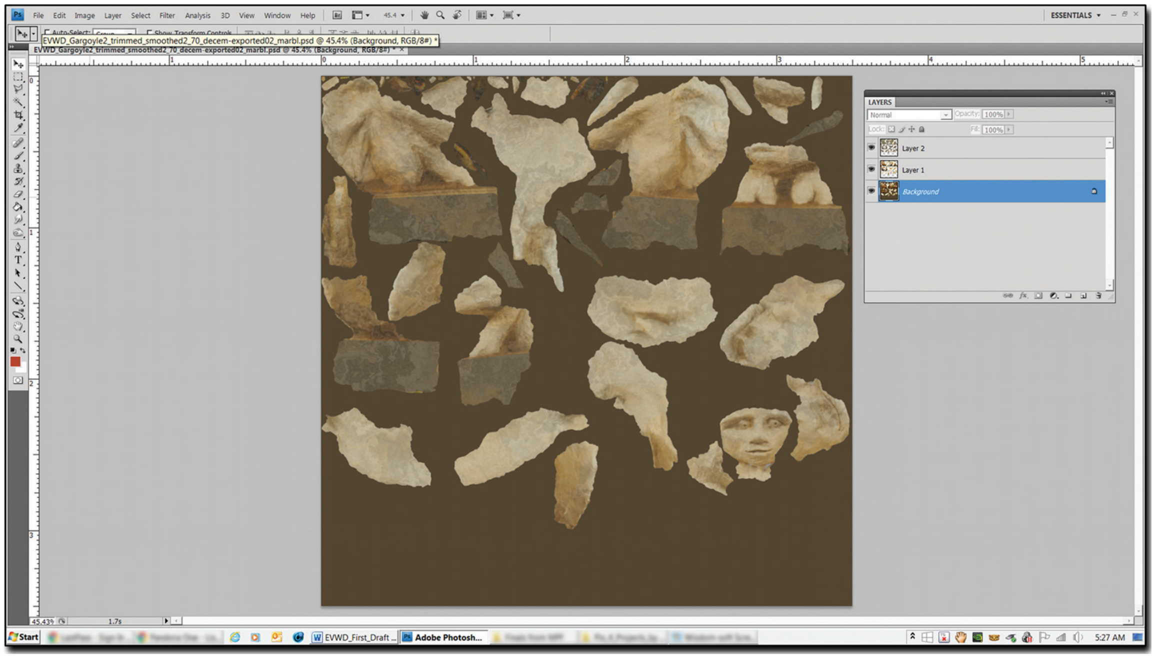

File Edit Image Layer Select Filter Analysis 3D View Window Help
EVWD_Gargoyle2_trimmed_smoothed2_70_decem-exported02_marbl.psd @ 45.4% (Background, RGB/8#) *
LAYERS
Layer 2
Layer 1
Background
ESSENTIALS
Start
Adobe Photosh...
5:27 AM

FIGURE 7.11 Reduction and modification of a graphic/texture map for the gargoyle model.

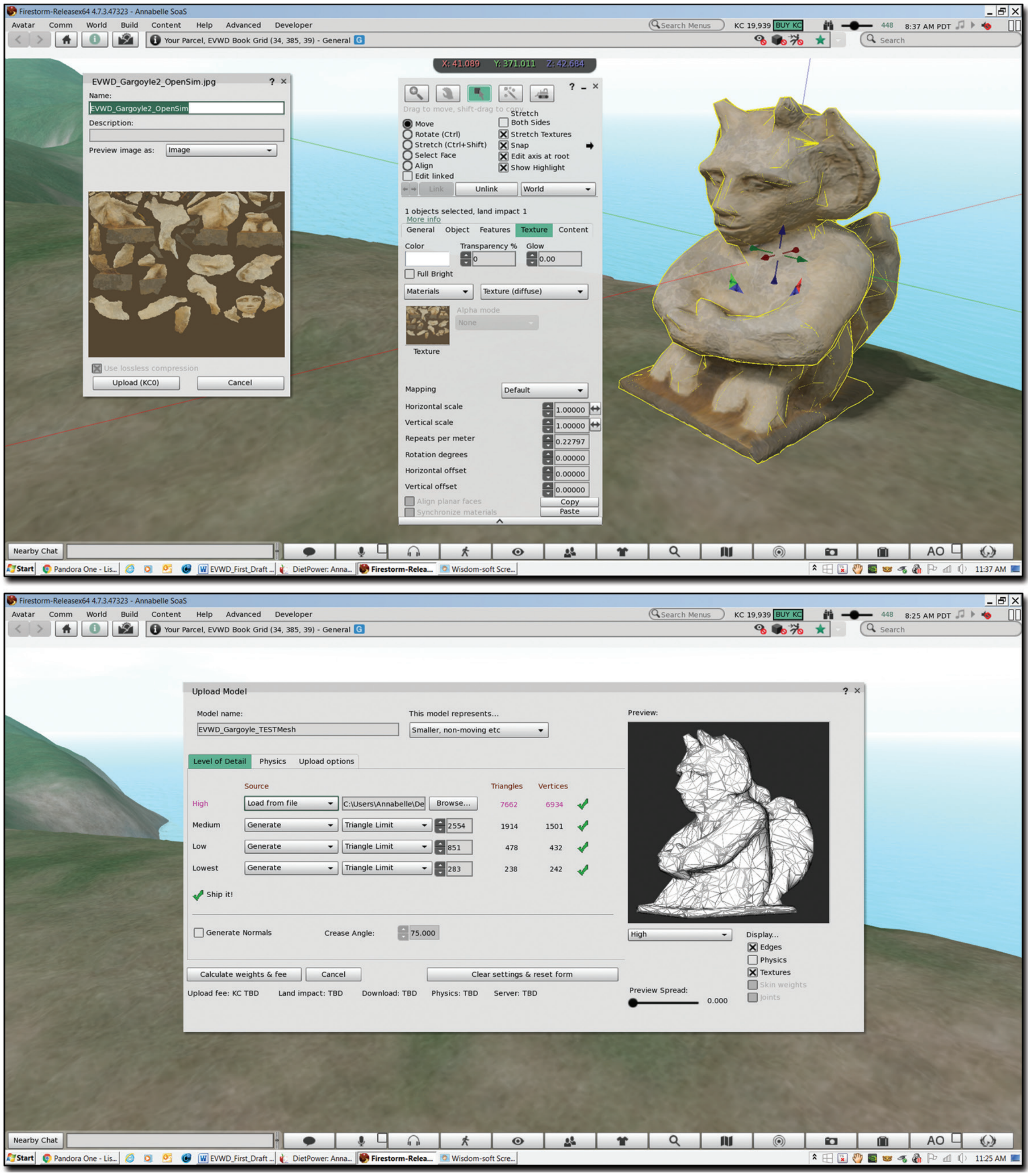
7.6.4 Bringing the Content into the Virtual Environment

The last step is to upload the 3D mesh model into your virtual environment. If you are not familiar with the upload menu, please refer to Section 7.5. As you can see from the screengrab of the upload menu in Figure 7.12, the gargoyle will upload with 8000 triangles in it, a high number, but necessary to bring out his organic form and details.

LOD Decision Point 3: Physics Level of Detail

As you are uploading the model into your virtual environment, consider how the physics of the object will need to function. If this is just a decorative piece and will not need to interact with avatars, then you have some options. You can (1) set the physics upload setting to the lowest possible level or (2) make a special low polygon model such as a box and use that as the physics file for the upload. The second option would also be a good choice if the avatar needs to walk, sit, or move around on your mesh, and will save physics processing time on the server for other things like the vehicles you are using.

Once you have the mesh uploaded, use the Upload/Image menu to bring in the special graphic/texture you created in Section 7.6.3. Apply it with the Build Editor/Texture tool in the Texture (diffuse) slot. The texture should wrap perfectly around the model and create a virtual version of the real object you started with. In Figure 7.13 is a screengrab of the final result.


Firestorm-Releasex64 4.7.3.47323 - Annabelle SoaS
Avatar Comm World Build Content Help Advanced Developer
Your Parcel, EVWD Book Grid (34, 385, 39) - General
Nearby Chat
Upload (KC0)
Cancel
Unlink
Copy
Paste
Calculate weights & fee
Cancel
Clear settings & reset form

EVWD_Gargoyle2_OpenSim.jpg
Name: EVWD_Gargoyle2_OpenSim
Description:
Preview image as: Image
Use lossless compression

X: 41.089 Y: 371.011 Z: 42.684

Move
Rotate (Ctrl)
Stretch (Ctrl+Shift)
Select Face
Align
Edit linked

Stretch
Both Sides
[✓ Stretch Textures]
[✓ Snap]
[✓ Edit axis at root]
[✓ Show Highlight]

1 objects selected, land impact 1

General Object Features Texture Content

Transparency %	0
Glow	0.00
Full Bright	
Materials	Texture (diffuse)
Alpha mode	None
Mapping	Default
Horizontal scale	1.00000
Vertical scale	1.00000
Repeats per meter	0.22797
Rotation degrees	0.00000
Horizontal offset	0.00000
Vertical offset	0.00000

Upload Model

Model name: EVWD_Gargoyle_TESTMesh
This model represents... Smaller, non-moving etc

Level of Detail Physics Upload options

	Source			Triangles	Vertices
High	Load from file	C:\Users\Annabelle\De		7662	6934
Medium	Generate	Triangle Limit	2554	1914	1501
Low	Generate	Triangle Limit	851	478	432
Lowest	Generate	Triangle Limit	283	238	242

Ship it!

Generate Normals Crease Angle: 75.000

Upload fee: KC TBD Land impact: TBD Download: TBD Physics: TBD Server: TBD

Preview: High
Display...
[✓ Edges]
Physics
[✓ Textures]
Skin weights
Joints
Preview Spread: 0.000

FIGURE 7.12 Uploading the graphic/texture (top image) and 3D model (bottom image) into the game-based sim.

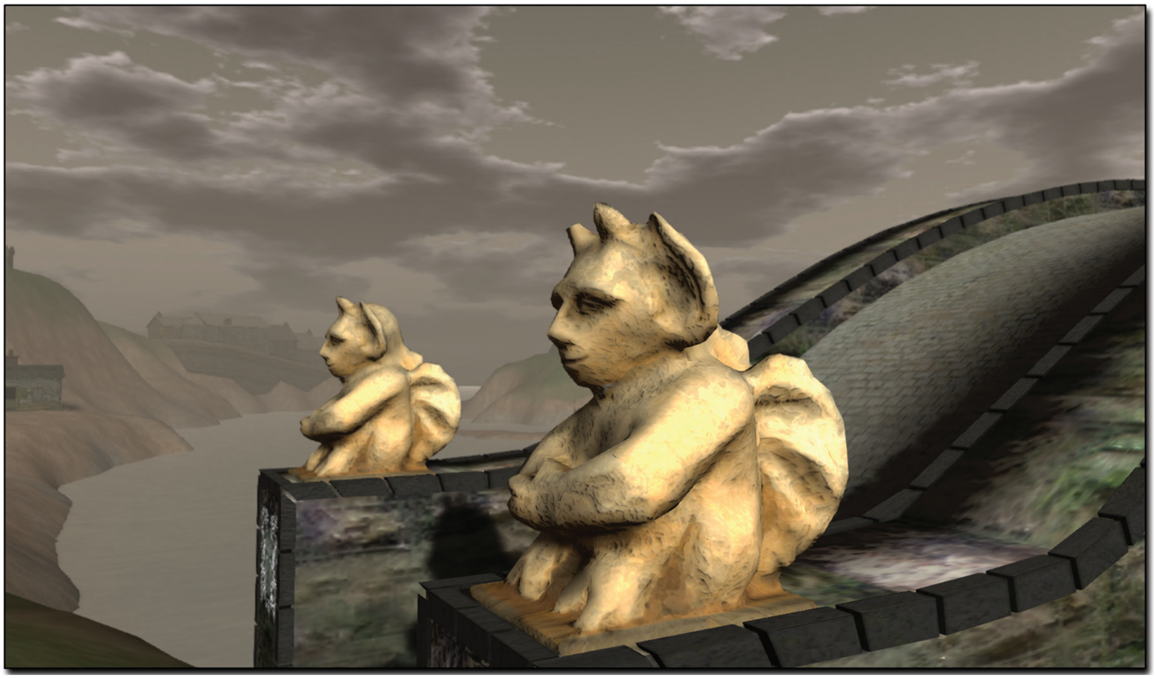


FIGURE 7.13 Gargoyle with graphic applied; final result shown inworld.

7.7 CHAPTER SUMMARY AND FINAL THOUGHTS

Working with 3D models and their LOD is an art and a science. As a designer you should endeavor to plan for how your models may be optimally viewed by the visitor to your virtual environment, and tailor your mesh uploads to serve those ends. Remember that there are individual differences in color and contrast perception, stereovision, and focal distance among your visitors, as well as the environment differences such as background illumination, persistence of visibility, and relative speed of objects that you as a designer may choose to add into the virtual scene. By using your knowledge and experience gained by testing the model's level of detail in various scenarios, you can greatly enhance the visitor's experience in your virtual world as well as provide for smooth performance overall.

REFERENCES

1. James H. Clark, "Hierarchical Geometric Models for Visible Surface Algorithms," *Communications of the ACM* 19, no. 10 (October 1976), online link to digital scan, accessed October 16, 2014, http://design.osu.edu/carlson/history/PDFs/clark-vis-surface.pdf.
2. *Wikipedia*, s.v. "Algorithm," accessed October 18, 2014, http://en.wikipedia.org/w/index.php?title=Algorithm&oldid=640257353.
3. David Luebke, Martin Reddy, Jonathan D. Cohen, Amitabh Varshne, Benjamin Watson, and Robert Huebner, *Level of Detail for 3D Graphics* (The Morgan Kaufmann Series in Computer Graphics), Kindle edition, 276–281.
4. David Luebke, Martin Reddy, Jonathan D. Cohen, Amitabh Varshne, Benjamin Watson, and Robert Huebner, *Level of Detail for 3D Graphics* (The Morgan Kaufmann Series in Computer Graphics), Kindle edition, 283–289.

5. John David N. Dionisio, William G. Burns III, and Richard Gilbert, “3D Virtual Worlds and the Metaverse: Current Status and Future Possibilities,” *ACM Computing Surveys*, 45, no. 3, article 34 (June 2013), accessed October 18, 2014, doi: http://dx.doi.org/10.1145/2480741.2480751.
6. Diemo Schwartz, Roland Cahen, François Brument, Hui Ding, and Christian Jacquemin, “Sound Level of Detail in Interactive Audiographic 3D Scenes,” *Proceedings of the International Computer Music Conference (ICMC) July 31–August 5, 2011, Huddersfield, UK*, accessed October 18, 2014, http://articles.ircam.fr/textes/Schwarz11b/index.pdf.
7. Scott Lawrence (Oz Linden), e-mail message to author, October 30, 2014.
8. *Wikipedia*, s.v. “Mipmap,” accessed on October 31 2014, https://en.wikipedia.org/wiki/Mipmap.
9. Lance Williams, “Pyramidal Parametrics,” *Computer Graphics* 7, no. 3 (July 1983), accessed November 9, 2014, http://faculty.cs.tamu.edu/jchai/CPSC641/p1-williams.pdf.
10. “Draw Distance,” *Wikipedia*, accessed October 31, 2014, http://en.wikipedia.org/w/index.php?title=Draw_distance&oldid=630482644.
11. *Wikipedia*, s.v. “Distance Fog,” accessed October 31, 2014, http://en.wikipedia.org/w/index.php?title=Distance_fog&oldid=613175322.
12. *Wikipedia*, s.v. “Visual System” accessed October 31, 2014, http://en.wikipedia.org/w/index.php?title=Visual_system&oldid=636715377.
13. *Wikipedia*, s.v. “Parallax,” accessed October 31, 2014, http://en.wikipedia.org/w/index.php?title=Parallax&oldid=636325618.
14. *Wikipedia,* s.v. “Depth of Field,” accessed October 31, 2014, http://en.wikipedia.org/w/index.php?title=Depth_of_field&oldid=630350759.
15. *Wikipedia*, s.v. “Contrast Sensitivity,” accessed October 31, 2014, http://en.wikipedia.org/w/index.php?title=Contrast_(vision)&oldid=635408713.
16. David Luebke, Martin Reddy, Jonathan D. Cohen, Amitabh Varshne, Benjamin Watson, and Robert Huebner, *Level of Detail for 3D Graphics* (The Morgan Kaufmann Series in Computer Graphics), Kindle edition, 2856.
17. “Mesh and LOD,” *Second Life Wiki*, accessed November 2, 2014, http://wiki.secondlife.com/wiki/Mesh_and_LOD#Havok_Physics_tips_and_oversimplifications.

This page intentionally left blank

8 Designing with Virtual Physics in Mind

Nothing happens until something moves.

Albert Einstein

8.1 OVERVIEW OF DESIGNING WITH VIRTUAL PHYSICS IN MIND

In a virtual world your avatar can knock down buildings with a finger flick, build giant games full of rolling boulders, and fly vehicles that seem to defy the laws of gravity. However, in order to perform these incredible feats, the virtual world must have a way of translating the real-world laws of physics into the virtual environment. This job is done by a component (or plugin) known as the physics engine. The physics engines used in virtual worlds (BulletSim for OpenSim, Havok4 for Second Life) are components in the virtual world simulator code that apply a “library” of rules (algorithms) to a “physical model” that surrounds an object when its physical state is set to “on” in the Build or Edit menu. This physical model includes things like mass and velocity as well as motive attributes such as acceleration. These attributes allow the object to behave in an almost realistic manner when it is pushed, dropped, shot, kicked, or otherwise imbued with some kinetic energy. Physical qualities of your objects in a virtual environment and the physics engines that define the behavior of these objects are an integral part of a virtual environment. As a designer you should know how to work with them, and how to choose the appropriate physics shape type and physics material property of the object (making them behave like wood, glass, plastic, etc.) for the project. Here are some of the concepts that we will explore in this chapter:

- There are many types of physics engines being used in virtual environments, and their influence on how the objects in your virtual environment can be managed by using scripting in the object, making physical material choices, and optimizing the physics shape settings.
- There are several kinds of simulated physics behaviors including solid body collisions, soft body collisions, and fluid dynamics. Currently, some can be displayed in an online real-time virtual world environment such as OpenSim or Second Life, and some have to be faked with clever scripting.
- Regional terrain design should consider the vehicle type and support the smooth behavior of vehicles if they are going to be used in the project.
- Vehicular design and the physics that animate objects in a vehicle can enrich the narrative and meaning of the environment.
- Universal design should be considered in all vehicular design and include an accessible placement of cameras for the driver and the screen-based heads-up display (HUD) used for the operation of the vehicle.

The project at the end of this chapter will show you the capacities of the BulletSim engine in an OpenSim environment. You will be assembling a fairy boat for the game-based sim environment from the provided 3D content, setting up a prewritten script in it, and testing it for use as part of the game you have designed for the project in this book. (Note: The scripting for this boat will not work well with the Havok engine in Second Life. The source for a default boat vehicle script, from the Vehicle Lab-Learning Center, will be provided for those of you who want to make this vehicle in Second Life.)

8.2 SPOTLIGHT ON PROFESSOR VICKI ROBINSON: TEACHING REAL-WORLD PHYSICS IN A VIRTUAL WORLD

As I was exploring physics engines and how they are being used in virtual worlds, I found Professor Vicki Robinson (Oddprofessor Snoodle in Second Life) and her creation, The Oddprofessor's Museum and Science Center (http://maps.secondlife.com/secondlife/Mujigae/134/218/139). This is the home of the virtual physics lab for the National Technical Institute for the Deaf at Rochester Institute of Technology (RIT) in Rochester, New York. The top image of Figure 8.1 shows the welcome area hub of the OddProfessor's Museum and Science Center, and the bottom image shows three exhibits that teach about volume displacement located on a sky platform above the hub. In this virtual physics lab, which she constantly develops, Robinson teaches about physics, and her students do their lab assignments and homework. She started building the Museum and Science Center in 2009 on the RIT island in Second Life, collaborating with a student who could write LSL scripts, and created interactive displays of real-world physics problems in a virtual world. As an associate professor at RIT, she is curious about how to utilize the 3D virtual world environment to create an even more compelling display of the principles of real-world physics. With an undergraduate degree in physics education from the University of Illinois (Urbana-Champaign), and a master's degree from the University of Illinois (Urbana-Champaign) in deaf education, Robinson is uniquely suited to teaching deaf students about physics. Because deaf students are often at a lower reading level, comprehension of the written-word-based problems in physics can be difficult for them. Once she started her virtual physics lab, Robinson realized that she was onto something, and that 3D would only add to the students' capacity to understand the physics problems that had hitherto been just words on a page. In 2011, when the RIT Island closed, she moved out onto her own land in Second Life. In her real-life classroom, using a combination of sign language and finger spelling, she explains the 3D problem that their avatar is encountering in the virtual physics lab. She has learned to introduce her students carefully into the virtual world and to take things slowly. At the National Technical School for the Deaf there are about 1000 deaf students, and Robinson has learned not to assume that her students have much prior knowledge about virtual worlds. She has found that most of her students tend to take 3D worlds for granted, and often will not explore them without urging. When asked about her plans for the future, Robinson said, "I do intend to keep expanding and refining the equipment I have as my scripting gets better. Of course, in a virtual world like Second Life there are no physics for fluids, and all of that has to be faked with scripting." The website for RIT is http://www.rit.edu/ and for National Technical Institute for the Deaf the website address is http://www.ntid.rit.edu/. You can read more about Robinson's work at http://scholarworks.rit.edu/jsesd/vol17/iss1/5/. In Figure 8.1 are some screengrabs showing the welcome area and one of the demonstration platforms of Robinson's Oddprofessor's Museum and Science Center in Second Life. If you want to visit, the SLURL is http://maps.secondlife.com/secondlife/Mujigae/134/218/139.

8.3 THE BASIC ELEMENTS OF PHYSICS IN A VIRTUAL WORLD

The real world is a physically dynamic place. The cup of coffee by your elbow contains hot water full of actively moving coffee particles, pushing against each other and bouncing off the walls of the container

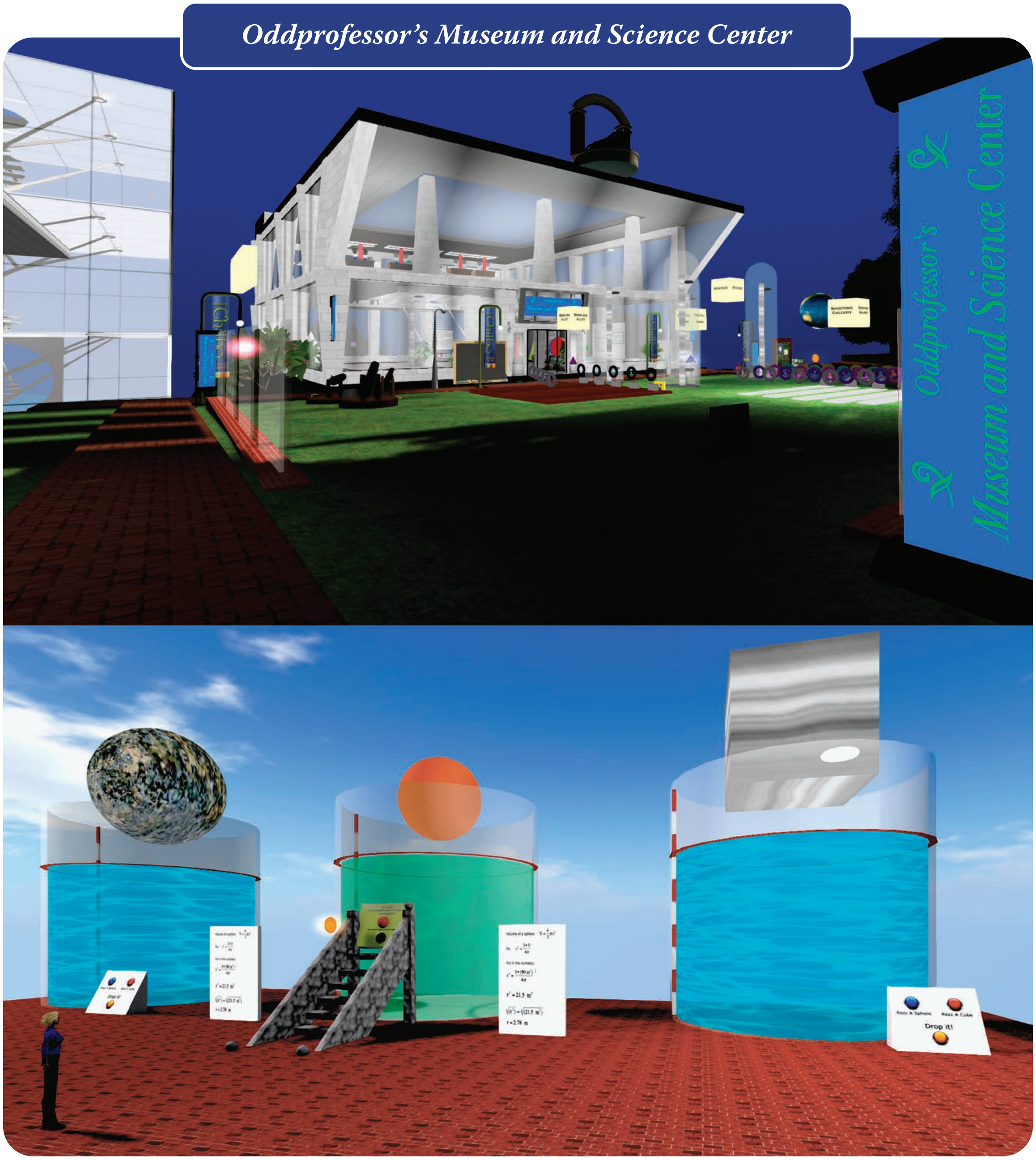

Oddprofessor's Museum and Science Center
Oddprofessor's Museum and Science Center
Drop it!

FIGURE 8.1 Oddprofessor's Museum and Science Center in Second Life.

in constant Brownian motion [1]. The solid floor under you, and the walls and windows around you are undergoing constant size shifts and planar twisting as the building's beams expand and contract in response to environmental temperature changes. The winds circulate our air, our muscles animate our bodies, and motors move our vehicles; these are all examples of the dynamic physics that we experience on a daily basis.

8.3.1 Rigid Body and Soft Body Dynamics

When virtual world developers want to show the motion of a virtual object and the reactions of other objects that come into contact with it in a real-time simulation, they rely on a physics engine, which uses a set of movement-defining rules called a *rigid body dynamics system* [2]. This physics engine establishes a sense of gravity, so things fall toward the ground. The movement of things like vehicles, collapsing building structures, and bowling balls hitting pins are examples of physical behavior that can be simulated with a rigid body dynamics system. Another kind of physics system found in computer simulations is the *soft body dynamics system*. This system is just starting to find a place in real-time virtual worlds, and can be used to display the movement and reactions of soft bodies, or objects that can be deformed. Think about how a tablecloth might bunch up as you grab it and drag it off a table, or how your clothing changes shape as you move around in it, or think about how soft things can layer on our bodies, such as the hat on top of our hair, on top of our skin. Soft body physics are everywhere: in fabric materials, in avatar hair, and in the moving skin and fat sections on a human figure. When your two-year-old throws her cloth dolly down the stairs, the twisting and turning of the doll's body and limbs can be simulated using soft body dynamics, in this case with a set of simulation rules called *ragdoll physics*. This is often used to simulate what happens to our human bodies when the forces of physics (especially those from rigid bodies) are applied to them, such as what happens to our bodies in a car crash or when we tumble down a flight of stairs. There is a lot of calculation necessary for displaying the impacts and resulting change in the position for all the parts in the rigid and soft body objects, often many more than can be handled by a real-time system like an online virtual world. At this stage of their development, online virtual worlds, such as OpenSim or Second Life, promote a sense of the real-world physics in several ways. As you can see from Figure 8.2 the physics engine in a virtual environment is connected to these basic functions:

- It defines empty space versus filled space, so that our avatars can walk on the land, stairs, and across sky platforms without falling through, but not pass through solid things like walls and fences.
- By enclosing objects with physics shapes (conceptualized as wireframe cubes around the dodecahedron shapes in the top right), the physics engine can generate physics-based reactions, such as falling and bouncing, on physical objects within the virtual environment.
- Vehicle behavior is created with special vehicle scripting as well as the basic physics engine responses to the objects the vehicle is made from.
- The virtual world simulator (containing the physics engine) calculates the physical responses, and the physics-enabled client viewer contributes physical material information to the physics engine.
- When the physics engine in the virtual world simulator is connected to a computer using an appropriate client viewer, then we have a means to see how our avatars can interact with objects that have physical properties.

In fact, physics calculations are so intensive that virtual worlds, such as Second Life and OpenSim, that stream data to you over the Internet have to put in limits and slightly change the physical reactions to prevent the whole sim from lagging badly and displaying a kind of jerky stop-motion response from the objects undergoing a physical change on your screen. This kind of *physics approximation* will create physics

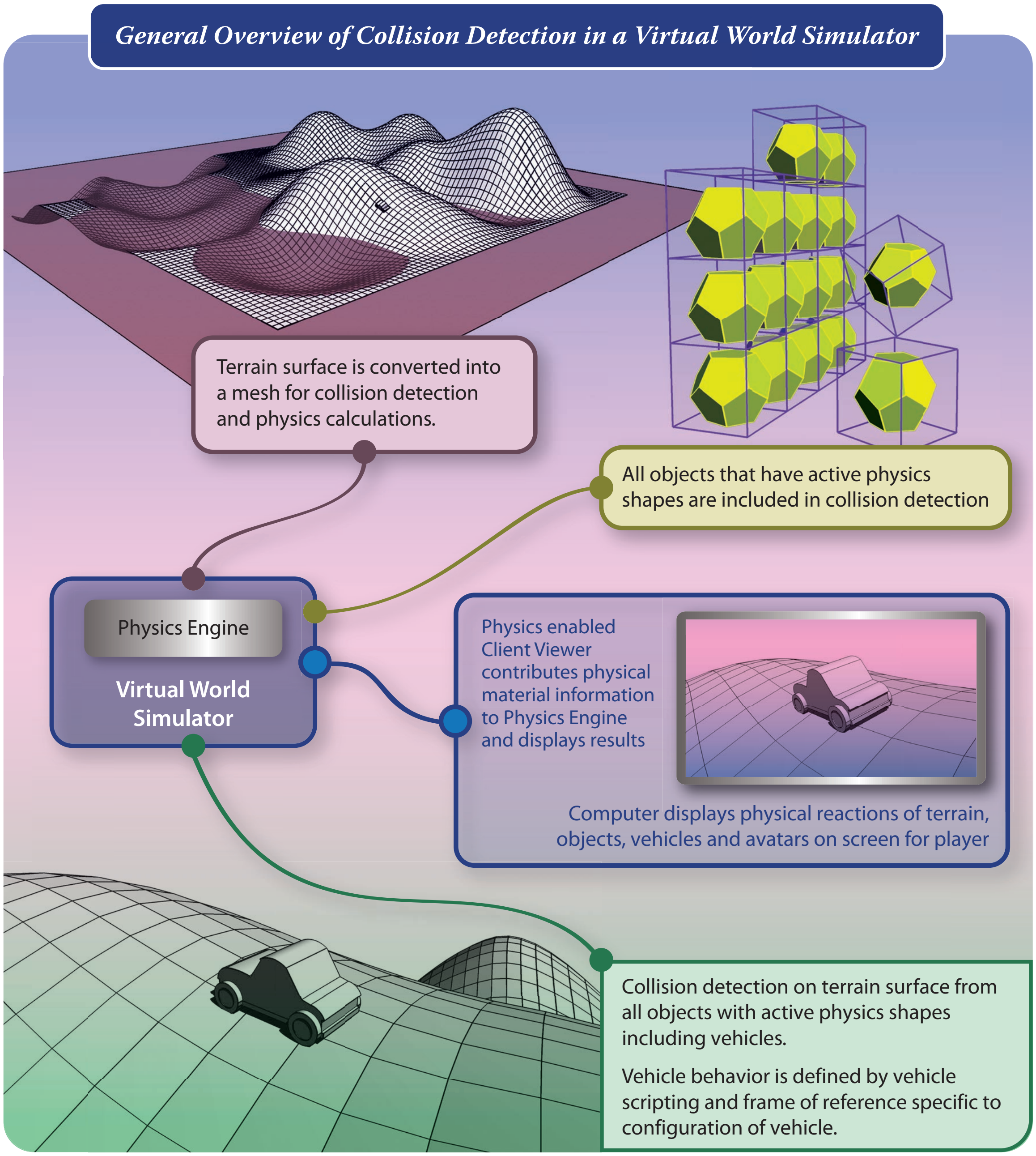

General Overview of Collision Detection in a Virtual World Simulator
Terrain surface is converted into a mesh for collision detection and physics calculations.
All objects that have active physics shapes are included in collision detection
Physics Engine
Virtual World Simulator
Physics enabled Client Viewer contributes physical material information to Physics Engine and displays results
Computer displays physical reactions of terrain, objects, vehicles and avatars on screen for player
Collision detection on terrain surface from all objects with active physics shapes including vehicles.
Vehicle behavior is defined by vehicle scripting and frame of reference specific to configuration of vehicle.

FIGURE 8.2 Overview of physics in a virtual world simulator.

reactions that we visually perceive as correct or "realistic" looking, even if they do not actually represent all the forces, actions, and reactions that would be occurring if these objects were in real life.

Other optimizations that have been added into virtual world simulators or used by designers and builders are

- Physics calculations go to "sleep" when an object has not moved for a specified amount of time
- Sculpted prims are enclosed in simple cube physics shapes to reduce the amount of physics calculations, while a linkset of primitives will use the actual shape of the primitives to create a more complex physics shape that has a higher level of detail (LOD).
- Physics shape LOD and the resulting performance can be optimized through the use of simple shapes developed by the designer for the meshes they are importing into the virtual world.
- Avatar physics that you can activate by creating and wearing a physics shape, such as clothing layer, will create movement of the breasts (female only), belly, and butt.
- Avatar clothing can be created as a "rigged mesh" and connected to the avatar's virtual skeleton, which moves the mesh clothing in a similar way to real clothing (almost like soft body dynamics) as the avatar moves.

8.3.2 Collision Detection and Design

A major function for a physics engine in a virtual world is *collision detection*. This is the capacity to detect when two solid objects are impacting on each other's surfaces. Although it is not mentioned much, the main use of this in a virtual world is to keep your avatar standing on the terrain, not falling through it. The "collision" of your avatar's feet is detected, and the terrain "pushes" back to keep your feet on the land, or on the surface of your floating skybox floor. Objects, such as work platforms, hanging in the air above your region, are in a special physical state. They are not physical themselves, or they would fall to the ground, and yet they can detect and function like a solid floor for the avatar because of their inherent physical shape. The collision detection system keeps houses from falling through the ground. It lets avatars pass through flexi-prim foliage (which the simulator considers as phantom objects), but forces our avatars to walk around the solid tree trunks. All of these physical qualities are being processed by the simulator to support the illusion of a virtual environment with realistic gravity-based physical reactions. This collision detection system is important to designers, and it is utilized by the simulation to create some interesting effects. For instance, physical objects can be maintained at a high altitude over the terrain, so there can be vehicles that work like aircraft. If you have designed a gun or cannon, you are employing the collision detection system to sense a hit or kill on your target. There are some differences in the way this kind of collision detection is done in virtual worlds. In the case of a bullet or cannon ball, Second Life uses a *discrete collision system* to capture the moves of a physical object across a series of frames or physical calculation snapshots, whereas BulletSim physics used in OpenSim utilizes *continuous collision detection*. To overcome the problem of small bullets missing their targets because the physics system cannot capture their positions in every frame with the discrete collision system, Second Life adds an invisible "tail" to each bullet, thereby giving the physics engine a longer time and bigger bullet object from which to calculate trajectory results [3].

8.3.3 Designing with the Client Viewer Physics Material Settings

Obviously we want our virtual guns to shoot bullets that cause virtual damage, so that they will knock down targets or alien intruders. We want our car crashes on the virtual racetrack to provide us with satisfying aerial trajectories as the cars flip over. All of this is the result of a scripted physical object working with the rigid body dynamics system in the physics engine of the simulator, and each gun script, bullet script, or vehicle script has similar variables that you can adjust to get the results you want to see. However, before we get into all that, let us look at the basic physics material settings features that are included in our virtual world and are accessible from the Build/Edit Menu/Features tab of the client viewer. These features settings are an often-overlooked aspect to building that you can utilize to add more realism and more entertainment value into your creations. For example, you can make a window that flies apart with explosive force when an avatar crashes into it, a room full of endlessly bouncing balls that play musical notes, a “zero gravity” lab that has equipment floating about, or perhaps the full-on devastation as a tornado twists across your terrain. These physical qualities can be applied to any primitive or any mesh you create, providing you with control over their gravity, friction, density, and restitution (the bounciness of an object). You also may control the material that these objects will represent physically, so, for example, a glass cube has a lower amount of friction and a rubber sphere has a higher degree of restitution. Think about how using lower friction settings on the runners of the sled you are building will benefit its performance in a downhill race and you will begin to see how these physics material settings can make your builds better.

8.3.3.1 Viewer Physics Material Settings Tests

Head over to a sandbox or somewhere that allows you to create objects, and try these simple experiments to gain an understanding of the materials physics features settings. These settings can be activated in the Firestorm client viewer in the Build Menu/Features tab (look under the Physics Shape Type settings on the right side of the menu).

As you can see from Figure 8.3, we have set up a physics test bed consisting of a large hollow box to act as a low wall around us to prevent our objects from escaping when they are physical. We also have added a box prim as a floor surface and surrounding wall with grid markings on it, so we can more easily judge the height or distance one of the physical objects moves when we interact with it. Using the drag hand (activated with Ctrl+2) will allow you to push these objects around, and the Edit menu tool will let you drag them straight up on the z-axis so you can drop them for tests. You can also tilt the test bed floor to test the friction settings; the low wall will catch the sliding prims before they roll off into oblivion.

8.3.3.1.1 Gravity Test

Gravity in a virtual world is how much an object is attracted to the ground. As you can see from these tests, a weightless condition or zero gravity can be created. Observe the physics material gravity settings by doing the following steps.

1. Create three cubes of equal size (1.5 meters to the side is good) on the ground and space them about a meter apart. Go to the Features tab of the Build menu and make sure that the default settings are Physics Shape Type = Prim and Material = Wood for all of them.
2. Now change the gravity settings for each prim. Set the right side cube at 1.0, the middle cube at 0.0, and the left side cube at –1.0.
3. Elevate them off the floor of your test bed by 5 meters, select them all, and turn them to physical prims in the Build/Edit/Objects menu.
4. Observe what happens to each cube, and notice how one falls down, one just floats, and one flies up into the sky.

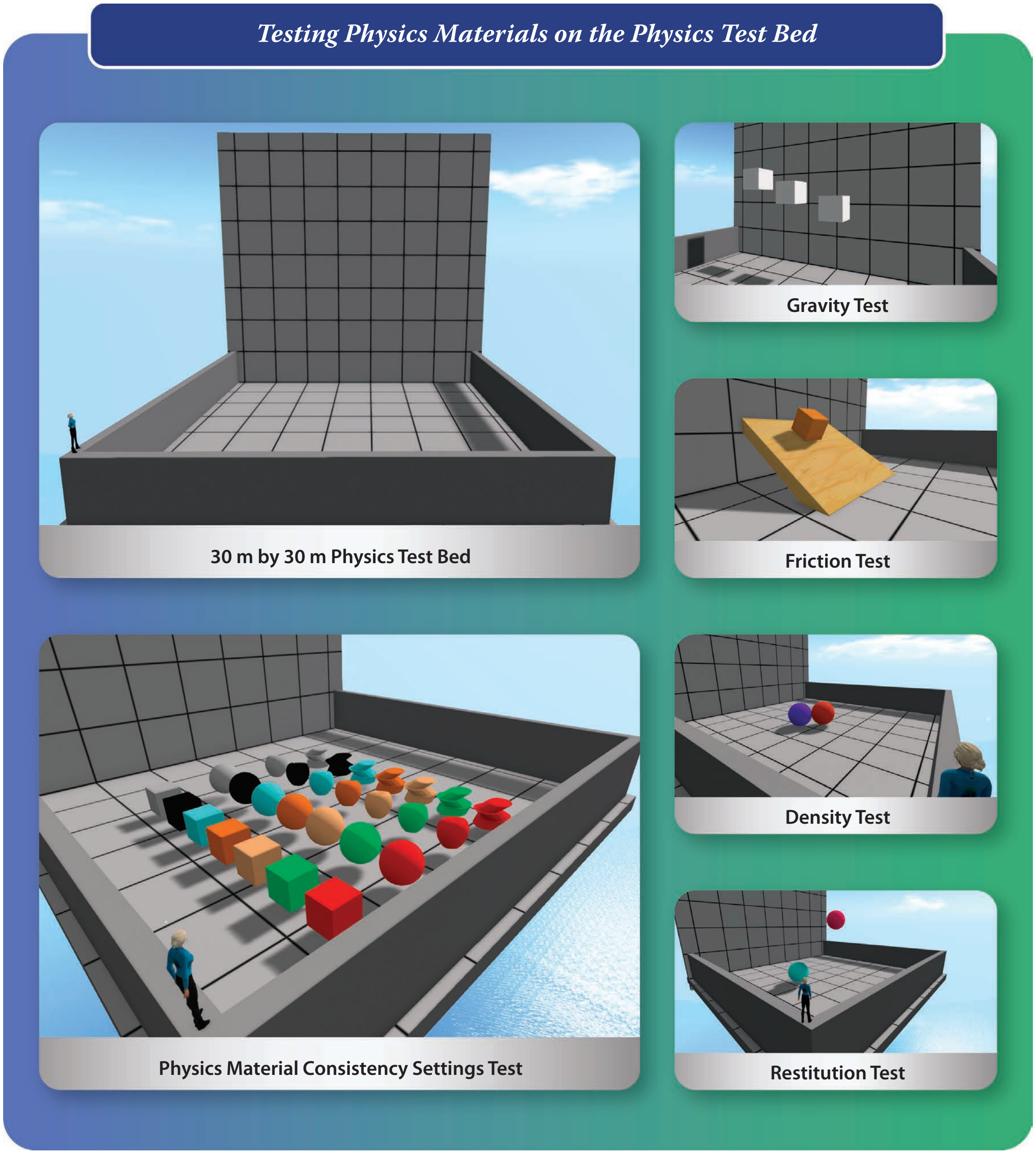

Testing Physics Materials on the Physics Test Bed
30 m by 30 m Physics Test Bed
Gravity Test
Friction Test
Density Test
Physics Material Consistency Settings Test
Restitution Test

FIGURE 8.3 Physics test bed used to study the characteristics of physics materials.

8.3.3.1.2 Friction Test

Friction in a virtual world demonstrates how slippery or sticky the surface of an object can be. It is best tested with flat-sided objects such as a cube sliding down the side of a box or ramp. Test the effects of the friction settings with the following steps.

1. Create a “sliding floor” or surface with a 10 meters by 10 meters box primitive that is tilted 30 degrees, and make it a wood material with a friction setting of 0.
2. Rezz a cube, tilt it 30 degrees so it sits flat on the floor surface (make it also a wood material with a friction setting of 0), and let the “gravity” try to pull it down the tilted surface, noting if the cube slides, how far it slides, or if it sticks in place.
3. Change the sliding floor’s physical material friction setting to 50, 128, and then 255 (highest friction), and make note of any changes in the sliding behaviors.

8.3.3.1.3 Density Test

Density is the specific weight or the mass of some material given a specific volume. In the real world, a granite rock sinks to the bottom when thrown into a pond and a wood plank floats on the top of the pond, because the granite has a higher density than wood or water, and wood is less dense than the water it floats on. In a virtual world, the water does not have real-world properties yet, but you can change the densities of various objects to influence how they collide with each other. Check these settings out with the following steps; in this test you will discover how density affects a collision between two objects.

1. Create two large spheres (3 meters in diameter) and set the density of one to 1000 and the other to 22587 (maximum setting).
2. Use the drag hand to push them into one another and notice how the lighter density sphere will bounce off the heavier density sphere, whereas the higher density (“heavier”) sphere will hardly move from its place.

8.3.3.1.4 Restitution Test

Although we refer to restitution as the bounciness of an object, there is more going on physically than that. The speed at which an object travels after a series of collisions also changes. You can see this when a rubber ball bounces across the surface of the floor, and you will also notice that the height of each succeeding bounce is diminished as the ball loses its kinetic energy. Make your observations by following these steps.

1. Make two spheres at 2 meters in diameter and set their restitution for 0 and 1.0, respectively.
2. Select them both and raise them up 10 meters from the flat floor surface.
3. Change both to physical and deselect them so that they can both fall at the same time.
4. Make note of how the bounce behavior is affected by the settings in restitution.

Now that you have seen the basic range that each setting can provide for a wood cube or sphere, it is time to take the physical materials settings out for a test drive.

8.3.4 Physics Materials Tests on a Cube, Sphere, Sculpted, and Irregular Mesh

Now that you have been introduced to the four physical qualities available under the Features tab, let us have some fun and build your knowledge base of these physics materials properties.

1. Make one cube (1.5 meters to the side), one sphere (2 meters in diameter), one sculpted prim (using the default "apple" shape, and set to 2 meters) and import a simple irregular mesh (as close to 2 meters as possible) to use as your test objects. (Note: In the content provided for this chapter at http://www.anncudworthprojects.com/, there will be a test mesh called Physics Materials Test Mesh.dae for you to utilize if you do not have the means to create one yourself.)
2. Set the four objects to the blank (or plain white texture color) to remove the default wood pattern.
3. Copy each of them seven times, using the Click+Shift+Drag method and the built-in snap grid to lay them out in an organized set of rows, creating seven lines containing the group of four objects.
4. Change all the members of each group (cube, sphere, sculpted, and mesh) into one of the seven materials available on the Build/Edit menu under the Features tab in the physics settings menu.
5. Organize them with a color code, such as Stone (gray), Metal (black), Glass (blue), Wood (yellow), Flesh (tan), Plastic (green), and Rubber (red) so you can keep track of how each kind of material responds to your physics tests.
6. You should have a set of seven cubes, seven spheres, seven sculpts, and seven meshes, each with its corresponding material color and settings.
7. The Physics Materials Consistency Settings Test provides us with the default settings for friction and restitution in the seven basic materials [4]. Use these settings as a baseline to make your observations from.

Material	Friction	Restitution
Stone	0.8	0.4
Metal	0.3	0.4
Glass	0.2	0.7
Wood	0.6	0.5
Flesh	0.9	0.3
Plastic	0.4	0.7
Rubber	0.9	0.9

8. Go ahead and change the settings on all 28 of the test objects, so that the members of each group are all the same, for example, all plastic objects will have a green color, with a friction setting of 0.4 and restitution setting of 0.7.
9. Select them all, drag+Shift a copy and take the copy of the whole group into your inventory. Name the coalesced group in your inventory "Physics Materials Testers."
10. Select and drag the original group up about 3 to 4 meters over your physics test bed.
    Now for the fun part!
11. First you will just turn each one of the objects into a physical object in the Build/Edit menu under the object tab.
12. Start the gravity/restitution test by letting them drop one by one, by shape family, and observe what happens, noting how fast they fall, how many times, and how high they bounce.
13. Also note how the sculpted objects movement may be different from what you expect, because of its boxy physics shape.
14. You can utilize the format shown in Table 8.1 to make your own observations about how each of these objects (cube, sphere, sculpted, and irregular mesh) responds physically when it has a different material assigned to it.

TABLE 8.1
Observations of Physics Features by Material Chosen in Client Viewer

Type of Object = ______	**Notes on Behavior Observed by Type of Material**						
Physics Setting	**Stone**	**Metal**	**Glass**	**Wood**	**Flesh**	**Plastic**	**Rubber**
Gravity							
High = >1.0							
Low = .5							
None = –1.0							
Friction							
(Stick or Slide)							
Low = 50 High = 250							
Density in 100 kg/m^3							
Low = 1000							
High = 22587							
Restitution (bounciness)							
Low = .25							
High = 1.0							
Sound Emitted							
Yes or No							
Like or Dislike							

15. Delete the first set of test objects and drag onto the physics test bed a new copy of the Physics Material Testers from your inventory.
16. Separate the spheres and delete them, and then elevate the rest of the objects up 10 meters to get them out of the way.
17. Raise and tilt the test bed floor, so you have a sliding surface to test the friction of each material.
18. Lower and rotate your test objects so that they can slide on a face down the floor surface into the retaining wall.
19. Again, for all of these tests, you can utilize the format shown in Table 8.1 to make your own observations about how each of these objects (cube, sphere, sculpted, and irregular mesh) responds physically when it has a different material assigned to it.

This kind of basic "research" will supply you with a broad base of experiential understanding relevant to how the physics materials work, and will make you a stronger, more insightful designer overall.

8.3.5 Collision Sounds, Physical Materials, and Design

The physics engine will fill in a default sound related to the chosen material of your object whenever you change its state to physical and have not specified that it play a special sound when it has a collision event. On every collision, there is a "thunk" sound when the cube is made from wood and a "clink" sound when the cube is glass, and so on, one sound specific for each of the seven materials. Obviously this is where you can start to have some fun with physical objects, and many designers have created unusually tuneful environments and even musical instruments by changing the sounds that a physical object emits when it detects a collision [5].

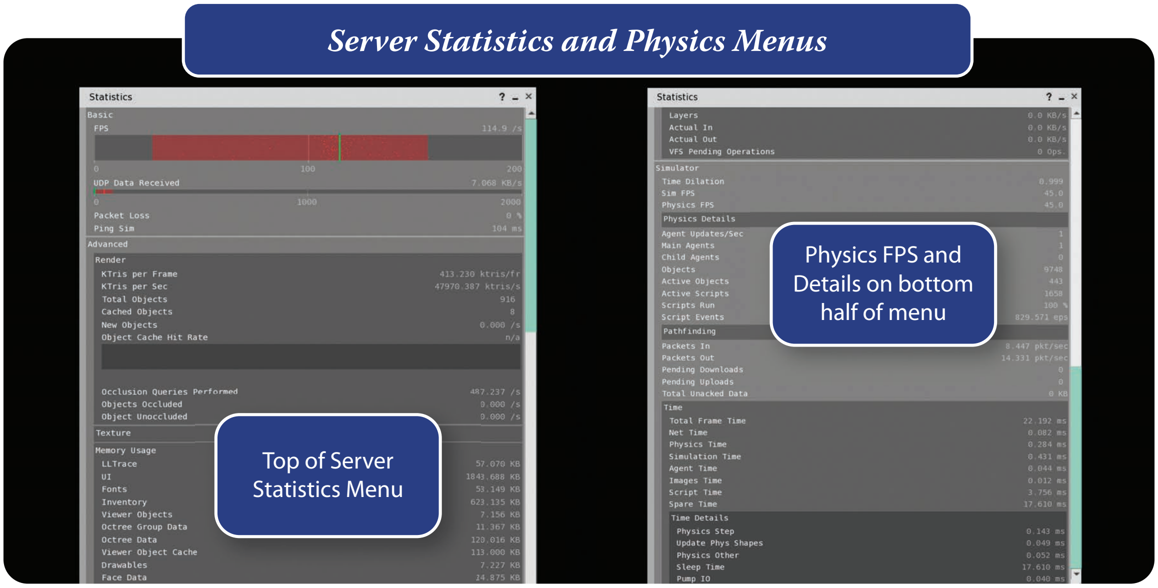
8.3.5.1 Using the Statistics Menu as a Physics Performance Gauge

By using the key command Ctrl+Shift+1 (in the client viewers Firestorm, Singularity, and Second Life) you can bring up the Statistics menu and discover how much of a demand the objects and avatars in the scene are putting on the physics engine. Scroll down until you can see the Physics FPS reading. Anything less than 40 FPS is a sign that the server is falling behind as it tries to calculate the number of collisions on the sim. As the designer, you can track the server's physics calculations to make sure that your physical objects and the resultant collision calculations are not lagging the sim. In Figure 8.4 is a screengrab showing the server Statistics and Physics Display menu for the Firestorm client viewer.

8.3.6 Physics Shapes and Best Practices for Building for a Low Physics Demand

Since any physical object you make or import into the virtual environment will have to calculate collision detection with an avatar and any other physical object, you should consider this usage carefully. Here are some best practice guidelines to help lower the load on the server's physics engine. As you will note in Figure 8.5, in order to see the physics shape each object in your virtual scene has attached to it, you will need to check on the Physics Shapes under Render Metadata in the Developer's menu of the Firestorm client viewer. (Note: You can show the Developer's menu by using the key command Ctrl+Alt+Q in Firestorm and Second Life.) The first basic step is to, whenever possible, choose "none" in your physics shapes options list, under the Features tab of the Build menu.

Try toggling some of the physics shapes of the objects you have built to see how it changes the display for physics in the Render Metadata/Physics Shapes. Also try toggling the object to phantom and physical in the Build Menu. As you can see, *making an object phantom does not remove it from the physics engine calculations, whereas setting the object to "none" for its physics shape does remove it.* There are a couple of options for partial physical shapes, when you do need to have some "solidity" in your compound objects but not in all of the structure. For instance, if you have a mesh tree trunk attached to lots of tree branches, there are two simple methods to reduce the amount of physics calculations.



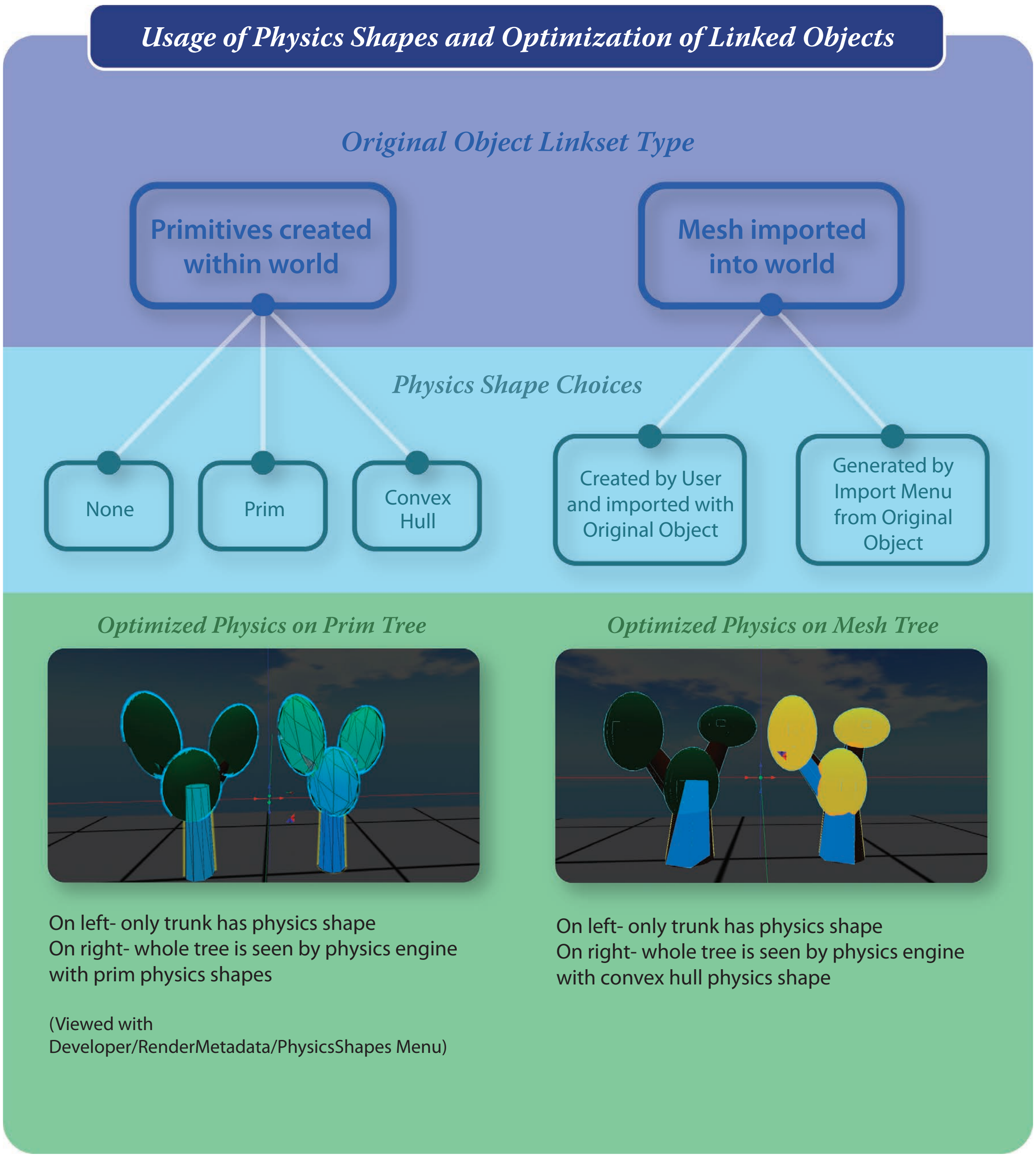
Server Statistics and Physics Menus
Top of Server Statistics Menu
Physics FPS and Details on bottom half of menu
Statistics
? – ×

```
Statistics
Basic
  FPS                                          114.9 /s
  0                    100                     200
  UDP Data Received                            7.068 KB/s
  0                    1000                    2000
  Packet Loss                                  0 %
  Ping Sim                                     104 ms
Advanced
 Render
   KTris per Frame                             413.230 ktris/fr
   KTris per Sec                               47970.387 ktris/s
   Total Objects                               916
   Cached Objects                              8
   New Objects                                 0.000 /s
   Object Cache Hit Rate                       n/a

   Occlusion Queries Performed                 487.237 /s
   Objects Occluded                            0.000 /s
   Object Unoccluded                           0.000 /s
 Texture
 Memory Usage
   LLTrace                                     57.070 KB
   UI                                          1843.688 KB
   Fonts                                       53.149 KB
   Inventory                                   623.135 KB
   Viewer Objects                              7.156 KB
   Octree Group Data                           11.367 KB
   Octree Data                                 120.016 KB
   Viewer Object Cache                         113.000 KB
   Drawables                                   7.227 KB
   Face Data                                   14.875 KB
```

```
Statistics
   Layers                                      0.0 KB/s
   Actual In                                   0.0 KB/s
   Actual Out                                  0.0 KB/s
   VFS Pending Operations                      0 Ops.
Simulator
  Time Dilation                                0.999
  Sim FPS                                      45.0
  Physics FPS                                  45.0
  Physics Details
  Agent Updates/Sec                            1
  Main Agents                                  1
  Child Agents                                 0
  Objects                                      9748
  Active Objects                               443
  Active Scripts                               1658
  Scripts Run                                  100 %
  Script Events                                829.571 eps
  Pathfinding
  Packets In                                   8.447 pkt/sec
  Packets Out                                  14.331 pkt/sec
  Pending Downloads                            0
  Pending Uploads                              0
  Total Unacked Data                           0 KB
  Time
   Total Frame Time                            22.192 ms
   Net Time                                    0.082 ms
   Physics Time                                0.284 ms
   Simulation Time                             0.431 ms
   Agent Time                                  0.044 ms
   Images Time                                 0.012 ms
   Script Time                                 3.756 ms
   Spare Time                                  17.610 ms
   Time Details
    Physics Step                               0.143 ms
    Update Phys Shapes                         0.049 ms
    Physics Other                              0.052 ms
    Sleep Time                                 17.610 ms
    Pump IO                                    0.040 ms
```

FIGURE 8.4 Server statistics and physics display menus in the client viewer (Firestorm).


Usage of Physics Shapes and Optimization of Linked Objects
Original Object Linkset Type
Primitives created within world
Mesh imported into world
Physics Shape Choices
None
Prim
Convex Hull
Created by User and imported with Original Object
Generated by Import Menu from Original Object
Optimized Physics on Prim Tree
Optimized Physics on Mesh Tree

On left- only trunk has physics shape
On right- whole tree is seen by physics engine with prim physics shapes

(Viewed with Developer/RenderMetadata/PhysicsShapes Menu)

On left- only trunk has physics shape
On right- whole tree is seen by physics engine with convex hull physics shape

FIGURE 8.5 Physics shapes seen by using the developer's menu.

Method 1

- Link the branches to the trunk sculpt (or mesh prim), and then in the Build/Features menu, make the branches into nonphysical objects (set their physics shape type to “none”).

Method 2

- Surround the mesh trunk with a regular box prim (physics shape type set to “Prim”) that has a transparent alpha texture on it.
- Then link the branches to the trunk and the trunk to the invisible box prim.
- Activate Edit/Linked, and set the mesh trunk and boxes in the Build/Features menu to have a physics shape type of “none”.
- The box around the trunk will act as a simple, efficient collider for the physics engine, while the rest of the tree is ignored by it.

By consistently watching your physics readout, and planning for simplified physics shapes or colliders around your objects, you can best optimize your builds on the sim. For more examples of how to do physics optimization in your virtual environment see these links:

- For general advice about building with meshes and their physics shapes, visit http://wiki.secondlife.com/wiki/Making_Mesh_Physics.
- For general advice about physics optimization in your virtual environment, visit http://wiki.secondlife.com/wiki/Physics_Optimization.
- For a deeper understanding of the prim physics types and how they relate to LSL scripting, see http://wiki.secondlife.com/wiki/PRIM_PHYSICS_SHAPE_TYPE.

8.3.7 Kinematics, the Geometry of Motion: Yaw, Roll, and Pitch

If you are new to making vehicles, there is some basic information and terminology that is useful to know. One of the primary things you will need is an understanding of how a vehicle can move in space, and how that is defined with a common terminology that you and the scripter can both understand and work with. In the next two sections, we will concern ourselves with the physics principles surrounding movement and how it applies to vehicles in a virtual environment. This is a study of the “geometry of motion,” also known as *kinematics*. Kinematics is an interesting word. It is the English version of *cinématique*, which the French physicist André-Marie Ampère (a scientist we have immortalized with the electric current term *amps*) created from the Greek word κινεϊν or *kinein* that means “to move” [6]. The study of kinematics is the study of how objects move, and the representation of that movement through the use of formulas that define the geometrical position and trajectories of these objects. There are three important terms that you should understand if you endeavor to design and build a vehicle in the virtual environment. Take a look at the upper half of Figure 8.6, and imagine that you are sitting in the aisle seat of a transatlantic flight heading west from Europe to North America; pictured here is how the rotations on the x, y, and z axes would be described:

1. *Pitch* is the amount of upward and downward rotation the nose of the plane is making, as it seeks to adjust its altitude to land or take off, and this happens on the x-axis of the plane.
2. *Roll* is the amount of banking the aircraft is currently doing, how high the left wing is compared to the right, and this is a rotation on the y-axis, as the pilots correct their heading and fly toward the East Coast of North America.
3. *Yaw* is the heading the aircraft is on, in other words, how far left or right it has rotated on its z or upright axis as the plane heads toward a runway in Nova Scotia or Washington, D.C.

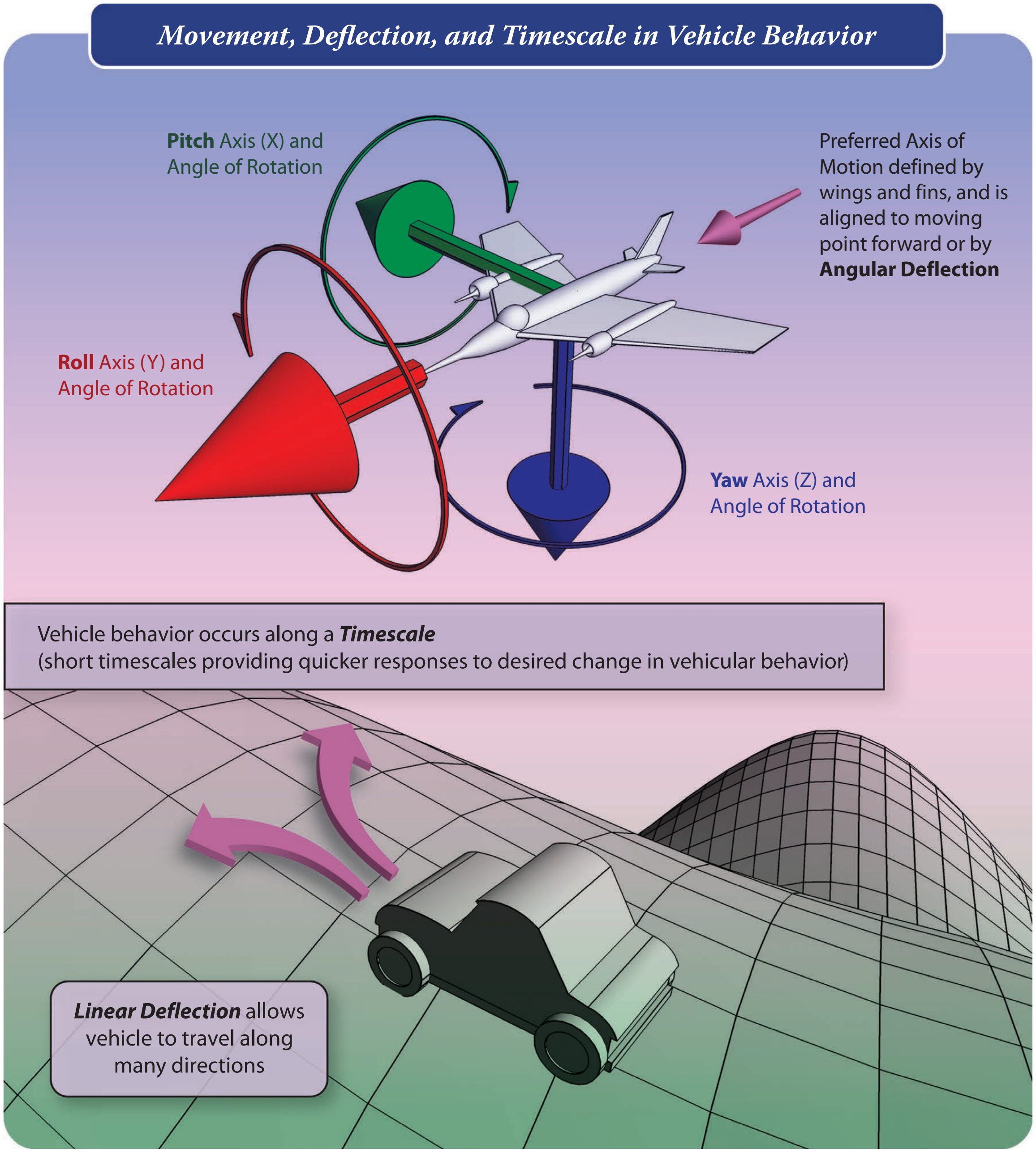

Movement, Deflection, and Timescale in Vehicle Behavior
Pitch Axis (X) and Angle of Rotation
Preferred Axis of Motion defined by wings and fins, and is aligned to moving point forward or by **Angular Deflection**
Roll Axis (Y) and Angle of Rotation
Yaw Axis (Z) and Angle of Rotation
Vehicle behavior occurs along a ***Timescale*** (short timescales providing quicker responses to desired change in vehicular behavior)
Linear Deflection allows vehicle to travel along many directions

FIGURE 8.6 Movement, deflection, and timescale in vehicle behavior.

These rotations are related to the x, y, and z axes because the physics engine needs to calculate how the vehicle will actually move. Every vehicle will move forward along the y-axis of the key prim in the vehicle's linked parts. The aircraft is also moving under the influence of *angular deflection*, as the wings and tail rudder help the vehicle remain pointed forward and moving along that line.

In the lower half of Figure 8.6, *timescale* (the response time of a vehicle to a change in direction) and *linear deflection* (the ability to move in many linear directions) are illustrated in regard to a wheeled vehicle moving on terrain. As a designer, you need to understand these basic movement terms and how they relate to the physics engine, so that you can design and build vehicles that will respond to a vehicle script in a reliable and consistent way.

8.4 BASIC VEHICLES AND THEIR FUNCTIONALITY

Now that OpenSim has incorporated the Bullet Physics engine as "BulletSim" into its configuration, vehicle development and testing has increased across the OpenSim-based grids. If you are developing vehicles and their scripts for usage on Second Life and OpenSim, then you need to know what LSL vehicle scripting functions are supported by each platform. A good place to find that information is at http://opensimulator.org/wiki/BulletSim/Functionality.

8.4.1 Basic Types of Vehicles, Deflection, Timescale, and Where to Get Test Scripts

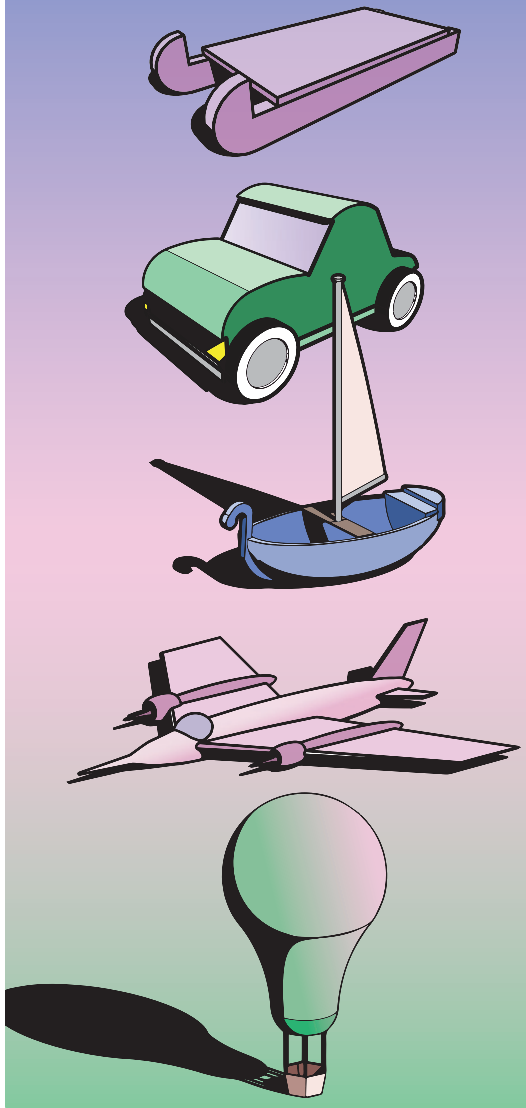
As you recall from Figure 8.6, there are two basic types of deflection common to all vehicles as they move along in their environment. You saw that angular deflection is defined by the wings and rudder of an airplane, which allow for the vehicle to move front point forward, and that a wheeled vehicle like a car can roll along many directions under the influence of linear deflection. Both of these kinds of deflection can be set up in the vehicle scripting to increase or decrease from the major axes along an overall timescale. For instance you could set up the vehicle to be pushed from one side for 60 seconds, so that it rolls around its z-axis and drives in a circle for that time period. For the purposes of learning how to script these vehicles, both Linden Lab and the Vehicle Lab-Learning Center (http://maps.secondlife.com/secondlife/Slate/205/27/267) have created basic vehicle scripts for you to examine and try out in your own builds. You will find the Linden Lab basic vehicle scripts at http://wiki.secondlife.com/wiki/Linden_Vehicle_Tutorial. Generally, the default vehicles are the following types: sled, car, boat, airplane, and balloon, and the basic parameters for their normal behaviors have been set up in the scripts.

8.4.2 Moving the Vehicle: Linear Movement versus Angular Movement

Let us suppose you have made a simple vehicle and would like to describe to your scripter how you see it moving along the terrain. As you know, there are two basic types of movement available for you to refer to. *Linear movement* is the movement along the vehicle's preferred axis, determined by the vehicle type, its wheels, and so on. *Angular movement* is the way in which the vehicle is steered or turned around one or more of its axes, such as a dart or airplane. Both of these movements have two timescale options available for the scripter: (1) the *motor timescale* which defines the timescale within which the motor reaches the limits of acceleration, or turning angle; and (2) the *motor decay timescale*, which sets up a gradual fade in the acceleration or the turning action, just as inertia would catch up to a real automobile when you take your foot off the gas.

As you can see in Figure 8.7, the basic types of vehicles can be described by their common movements. A sled will move in the direction of its runners, a balloon is likely to float up above the terrain. All of these vehicles exhibit movement behavior that relates directly to their angular deflection, linear deflection, and movement. They also have some additional movement qualities, which we will explore briefly next.

Types of Vehicles and Their Movement Behaviors



Sled: Preferred movement is along the local X-axis. Some friction and reaction to bumpiness of ground surface.

Car: Movement is influenced by linear deflection as well as impulse from motor and steering by driver.

Boat: Movement is influenced by angular deflection of rudder and sail as well as friction of water on hull. Hovers on the surface of water to float.

Airplane: Movement is along preferred axis with rudders, flaps and wings, uses linear deflection to create lift and banking angles on turn. No hover.

Balloon: Vehicle hovers and experiences wind friction. There is no deflection from wind unless scripted.

FIGURE 8.7 Types of vehicles and their movements.

8.4.3 Vertical Attraction and Banking

We have all described the flight of an airplane with our hands, and as we turn the imaginary plane in the air, our hand tilts right or left, as we *bank* into the turn. We also see this on racing tracks as the motorcycles racing around in the circle lean over to make the tight turn as fast as they can. Velodrome cycling tracks are tilted, so the cyclists do not have to bank on the turns; the track is banking up to meet them and keep them upright in the speed race. The virtual vehicle has something called a *vertical attractor* that helps keep the topside pointing up; something you would want in a boat or a balloon. This vehicle parameter function (`llSetVehicleFloatParam`) acts like an imaginary screen door spring that is attached from the global z-axis, to the top of your moving vehicle's local z-axis. It is always trying to pull your vehicle upright, and it can be connected to a timescale or dampened to have less or more of an effect (called the *efficiency*). Now to make the vehicle perform a banking maneuver, you will have to have the vertical attractor turned on, because the banking function is really a constrained angular velocity. As your airplane starts to roll on the y-axis in a clockwise manner, the vertical attractor starts to pull the vehicle into a right-hand turn, banking the craft by dropping its right wing. The banking can also be affected by a timescale and efficiency factor to make it bank more quickly or slowly, or with a greater or lesser angle. In Figure 8.8 is an overview of all the kinds of vehicle parameters that affect the movement of a vehicle. One of the most versatile and consequently difficult vehicles to write a script for is the helicopter, since it can move like an airplane or a dragonfly.

8.4.4 Buoyancy and Hover

Buoyancy and *hover* are only slightly related to each other in vehicle behavior. Typically the vehicle script will activate the built-in world buoyancy function if the vehicle buoyancy constant (`Constant: integer VEHICLE _ BUOYANCY`) is included. Since this is a number range, you can think of 0.0 as heavy unmovable gravity and 1.0 as full anti-gravity. If you have a vehicle that needs to hover, you can activate and influence these vehicle parameters just as you did for other behaviors on vehicles described in this section. Hover will be impacted by timescale and efficiency, as well as hover height. Various "flags" or special settings are available to get your vehicle hovering at the appropriate height, for example, a boat at the top surface of the water or a submarine under the water surface.

8.4.5 Reference Frame

The last basic vehicle feature for a designer to understand something about is the reference frame. Just as you shift between the World and Local coordinate axes in the Build menu, the physics engine can also access this capacity to shift the *Reference Frame* from the Local coordinate axes (the default for the Key or Root prim of your vehicle's linkset) to another orientation. This is useful when you want vehicles that move their engines to lift off like a rocket and fly forward like a jet when they are at altitude, changing their pitch to accommodate that maneuver.

8.5 IMPORTANT CONSIDERATIONS OF A VEHICLE DESIGN

Still breathing? Great! This is a lot to absorb and you should take your time with it. Work with your scripter, take out some of the sample vehicles, and mess around with the scripts. There is more detailed information about this in the Linden Vehicle Tutorial you will find at http://wiki.secondlife.com/wiki/Linden_Vehicle_Tutorial. In the design of a vehicle, the two most important spatial–structural considerations are (1) location of the key prim (or the seat that the avatar connects to) and (2) how the avatar's camera is positioned. The first is important

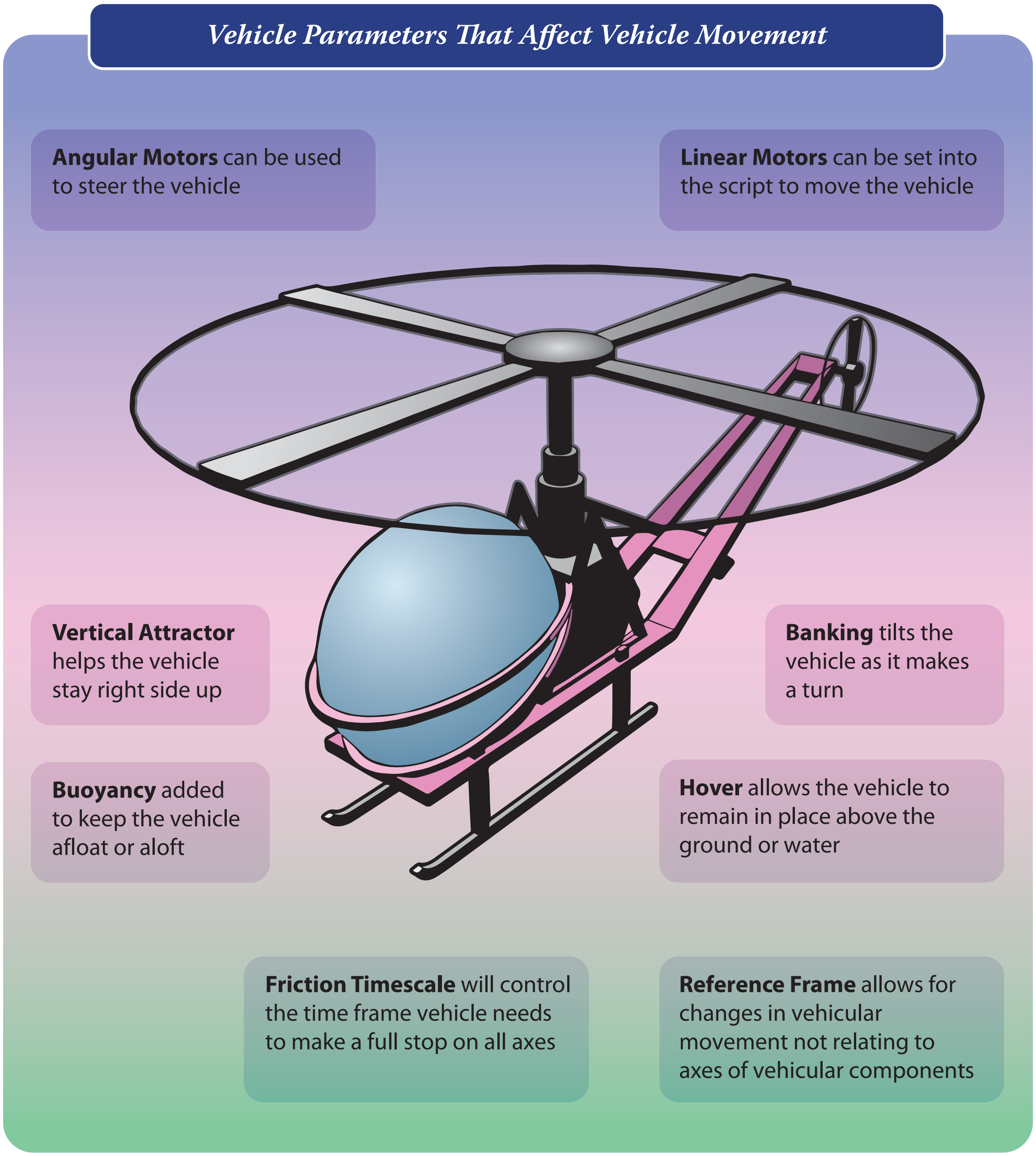

Vehicle Parameters That Affect Vehicle Movement
Angular Motors can be used to steer the vehicle
Linear Motors can be set into the script to move the vehicle
Vertical Attractor helps the vehicle stay right side up
Banking tilts the vehicle as it makes a turn
Buoyancy added to keep the vehicle afloat or aloft
Hover allows the vehicle to remain in place above the ground or water
Friction Timescale will control the time frame vehicle needs to make a full stop on all axes
Reference Frame allows for changes in vehicular movement not relating to axes of vehicular components

FIGURE 8.8 Basic vehicle parameters used by LSL scripting.

because where the avatar connects to the vehicle to "ride" or "drive" it will affect the overall center of mass or center of gravity. The avatar does have mass in the virtual environment, so you will need to account for that extra mass in the balance of your vehicle. This is especially true in vehicles such as airplanes, and you may find it necessary to move the seat prim around or add some invisible prims to counterbalance the vehicle when the avatar is seated, so the nose will stay up or the tail will level off. The second spatial consideration to give serious thought to is where you want the avatar's camera to be when it is seated in the vehicle. Typically the camera defaults to a point of view that looks down on the avatar's head and slightly forward. Obviously that point of view is not going to add to the feeling of being in a small fighter jet or a low-slung Formula 1 racing car. Various camera vehicle flags can be evoked by the vehicle script `llSetVehicleFlags` function when the avatar sits to drive or fly the vehicle, and you should look into them for vehicles like these. Three of the most useful vehicle camera parameters are Mouselook Steer, Mouselook Bank, and Camera Decoupled. You will find a full list of LSL-function-based vehicle parameters here: http://wiki.secondlife.com/wiki/Category:LSL_Vehicle.

8.6 UNIVERSAL DESIGN FOR VEHICLES

In the real world, the International Association for Universal Design (http://www.iaud.net/global/) works very hard to help designers and architects improve the accessibility of their vehicles and buildings and develop objects with their intended usage to be clearly perceptible to their user by design (also known as affordance). As a designer of virtual vehicles, you can improve the usability of your designs and the level of enjoyment for a first-time user by following suit. Here is a short list of qualities you should consider with your vehicular design:

- The vehicle's overall design should let the user know what the preferred direction of travel is and which way the avatar should be facing to operate it.
- Any instructions on the outside surface should be in a clear and readable font with high-contrast colors and translated into the major languages used by the owners.
- The vehicle should be playtested and iteratively refined structurally and programmatically so that ultimately its movement becomes responsive and intuitive when using the mouse, keyboard commands, or other kinds of input devices.
- Any HUD created to provide the driver with additional information as they operate the vehicle should be easy to read at a glance and should not obscure the driver's view.
- Sound cues should be backed up with visual cues in all operational methods, for instance, if you have the sound of engines starting up when an avatar sits down to fly the plane, then the local text chat should say "Engines Starting" or something like that.

8.7 DESIGN PLANNING FOR VEHICLE USAGE

The vehicle, no matter its type (boat, car, or airplane), is influenced by the same set of physics-based scripting parameters. In fact, when asked about vehicle types, Jopsy Pendragon had this to say:

> As far as I'm concerned there are only two fundamental vehicle types: Scuttlers and Floaters. Floaters may or may not give control over up/down movement (boats vs. submarines/balloons for example). Scuttlers, as the name implies, don't float, they rest upon something and move over it, whether it's prims or terrain (but not water), and they have to deal with friction. The biggest challenge I deal with is trying to help people master the "controls()" part of the LSL language, and for non-scripters, the bitwise and Boolean logic is kind of scary. I've tried two approaches, (1) ignore the man behind the curtain, edit these parameters, leave the script logic alone and (2) open the hood and dive in. The first tends to be far more efficient/less laggy, the latter, gets to be quite the opposite in the wrong hands.

If you want to "open the hood and dive in" take a "field trip." You can see all sorts of vehicles and test out basic vehicle scripts at Jopsy's latest educational facility, The Vehicle Lab-Learning Center in Second Life (http://maps.secondlife.com/secondlife/Slate/205/27/267). It is a sister lab to the Particle Lab (http://maps.secondlife.com/secondlife/Teal/180/74/22), where you can pick up basic particle scripts and information about using them in your vehicles.

8.7.1 Avatar Character and Vehicle Design Considerations

Any vehicle, in a real or virtual world, makes a statement about the personality of the driver. Geoffrey Miller, notable evolutionary psychologist, said, "Many products are signals first and material objects second. Our vast social-primate brains evolved to pursue one central social goal: to look good in the eyes of others" [7]. Think about minivans for the soccer mom, sports utility vehicles (SUVs) for the adventurer, and low-slung sports cars for the wealthy bachelor. All of these vehicles were selected to help the owner make a statement about their interests and the creative ways they choose to display themselves. There are six major dimensions of variation in personality—general intelligence, openness, conscientiousness, agreeableness, stability, and extraversion (also called GOCASE)—and these qualities influence consumer shopping and their choices [8]. Now what about the virtual world? In a virtual economy, just about anyone can afford a sports car, and find a road or track to drive it on. In fact, we can go for a drive under the sea and across the surface of different planets; the design possibilities are almost endless. As a designer, you are probably starting to think about these. Here are some interesting questions to consider when designing vehicles for your virtual environment that will appeal to a specific personality type:

1. What kinds of avatars will be using the vehicle? Human? Nonhuman?
2. What tasks will they be using the vehicle for?
3. Will the vehicle play a part in the narrative of the virtual environment?
4. Will the vehicles be shared by the public or for private use only?

Take out a pad and start to jot down your vehicular specifications. Then using Table 8.2, match up the vehicular specs with the avatar character that you see as most likely to be attracted to them. Given an understanding of the GOCASE qualities of your avatar character, what kind of vehicle would they be most likely to use?

As you can see from Table 8.2, the GOCASE personality qualities may respond differently to the kind of vehicle available for use. We need not even specify that the vehicle is a car, or plane, or boat; what matters is the kind of socializing opportunities it presents and the status it conveys upon its owner/driver.

8.7.2 Designing Vehicles with Region Crossings in Mind

One of the most frustrating aspects of driving, flying, or sailing across virtual lands is the loss of continuity in movement, and sometimes even the whole vehicle, when you cross over simulator regional boundaries. This unfortunate "hiccup" in server performance is caused by some of the following factors: (1) the number of prims that have been linked together (known as a linkset) in your vehicle, (2) the physics resource cost (PRC) of your mesh vehicle, (3) the scale or size of your vehicle, (4) the available calculating capacity of the physics engine on the server at the time you are crossing into the sim, and (5) the available object capacity in the region you are entering. As you know from studying the physics overview diagram in Figure 8.2, there is a lot of data being generated and exchanged between the physics engine and its coworkers (virtual world simulator, server, and client viewer). It does not take much of an anomaly to distort or cause lag in the results of the physical objects you are observing in the real-time environment. If you would like to get

TABLE 8.2
Avatar/Character Personality Traits and the Types of Vehicles They May Use

G.O.C.A.S.E. Traits, High to Low Scale (10–1)	General Intelligence High to Low	Openness High to Low	Conscientiousness High to Low	Agreeableness High to Low	Stability High to Low	Extraversion High to Low
Self-powered vehicle for personal transport	G = High (Y) G = Low (Y)	O = High (Y) O = Low (M)	C = High (Y) C = Low (M)	A = High (Y) A = Low (M)	S = High (Y) S = Low (M)	E = High (M) E = Low (M)
Green vehicle (electric or biofuel)	G = High (Y) G = Low (Y)	O = High (Y) O = Low (M)	C = High (Y) C = Low (N)	A = High (Y) A = Low (M)	S = High (Y) S = Low (M)	E = High (M) E = Low (M)
Family transport and/or group transport	G = High (Y) G = Low (Y)	O = High (Y) O = Low (M)	C = High (Y) C = Low (N)	A = High (Y) A = Low (N)	S = High (Y) S = Low (M)	E = High (Y) E = Low (N)
Two-person transport	G = High (Y) G = Low (Y)	O = High (Y) O = Low (M)	C = High (Y) C = Low (M)	A = High (Y) A = Low (N)	S = High (Y) S = Low (M)	E = High (Y) E = Low (N)
High-performance vehicle	G = High (M) G = Low (Y)	O = High (Y) O = Low (Y)	C = High (M) C = Low (Y)	A = High (M) A = Low (Y)	S = High (M) S = Low (M)	E = High (Y) E = Low (M)
High-performance and luxury vehicle	G = High (M) G = Low (M)	O = High (Y) O = Low (M)	C = High (N) C = Low (Y)	A = High (M) A = Low (Y)	S = High (M) S = Low (Y)	E = High (Y) E = Low (N)

Note: Potential inclination to use vehicle described: Y = yes, N = no, M = maybe.

a more detailed picture of what kinds of demands mesh models will put on a physics engine, look at this section of the Second Life wiki: http://wiki.secondlife.com/wiki/Mesh/Mesh_physics#Physics_Resource_Cost. Developers for both Second Life and OpenSim have initiated changes to cope with issues related to region crossings with avatars and vehicles. In 2012, Second Life introduced threaded region crossing (part of Project Shining), and in 2013, OpenSim introduced "Varregions" (named for a "variable region" size option in the code) [9,10].

Some general vehicle build practices to keep in mind for both Second Life and OpenSim are:

1. Decide early on how many objects you want in your complete vehicle linkset, how many will really need to be physical objects, and how many you can set to physics shape "none."
2. If you are designing for Second Life, do your initial uploads onto Second Life Beta, the test grid, so you can pare away the upload physical equivalent number in your mesh model as much as possible without incurring any upload costs.
3. When you are uploading a mesh vehicle to Second Life, keep the prim resource cost under the recommended limits (typically 32).
4. OpenSim physics can be set up on either the ODE or the BulletSim engine, but only BulletSim will support varregions (variable regions).

For a listing of client viewer compatibilities in OpenSim check here: http://opensimulator.org/wiki/Compatible_Viewers.

8.8 PROJECT: CREATING A FAE (FAIRY) BOAT VEHICLE FOR YOUR GAME-BASED SIM

With the help of Michael Thome (Vex Streeter in the Metaverse) who has created some intricate and creative vehicle scripts for us, we will assemble a unique and magical fae boat for your game-based sim. These scripts will work both in OpenSim regions using the BulletSim (a module of the Bullet Physics Engine) as well as Second Life. UUID codes are utilized in the scripts to call in the appropriate sparkle and wake particle graphics, and are specific to the virtual world you are in. We have included a sparkle and wake particle graphic with this project. However, you can also change out those graphics for your own graphic/textures, so you can customize the fae boat. The 3D model of the boat was created by Tim Widger, based on a design by yours truly. Essentially, we shall assemble all the components first, load in the scripts, connect the meshes with the scripted primitives and activate the vehicle. Here it is, step by step.

8.8.1 Getting the Content for Creating the Fae Boat

Everything you will need for making your first fae boat can be found on http://www.anncudworthprojects.com/ under the Extending VWD Downloads tab. Here is a list of what you will need for making your boat vehicle.

Scripts that run in the fae boat

- BoatEVWD.lsl—This is the script that makes the mesh into a vehicle. It is the engine of the boat and works with the physics engine in the sim. Michael has tuned it for the Bullet Sim engine, but it will work in Second Life too. It will not work with the Open Dynamics engine (ODE) physics. The boat is scripted to elevate as it moves along, so eventually it flies above the water magically.
- sparkles.lsl—This is a particle system that activates when you board the boat and dies away when you disembark. These particles can be customized by putting in the UUID code from your own particles.
- wake.lsl—This is a particle system that creates a wake for the boat. Like the sparkles, it activates when you board the boat and stops when you disembark. Also like the sparkles, it can be customized with your own particle's UUID code.

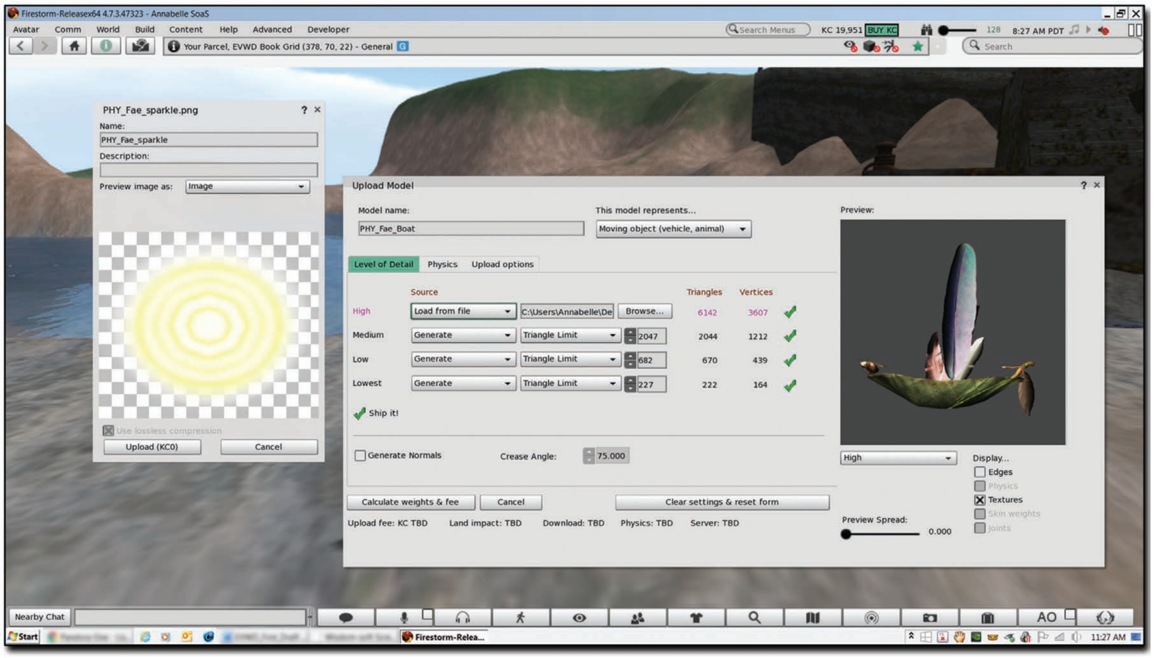
Other components used for making the boat

- PHY_Fae_Boat.dae—This is the boat model and should be uploaded with its graphics. Use this file for the physics file too when you do the upload.
- PHY_Fae_sparkle.png—This is the graphic to be used in the sparkle script. Put the UUID between the quotes in line 3 of this script.
- PHY_Fae_wake.png—This is the graphic to be used in the wake script. Put the UUID of this asset between the quotes in line 6 of this script.

Using your favorite client viewer, upload the mesh and graphic elements into a new folder called Fae Boat_All. In Figure 8.9 you can see a screenshot of the uploading process for the mesh boat and the sparkle particle graphic.

8.8.2 Organizing and Customizing the Content

Now that you have uploaded all the parts, you will need to bring in the scripts. Within the Fae Boat_All folder, create a Vehicle Scripts folder and make three new scripts in there. You can see them near the bottom of the Inventory list on the right side of the top image in Figure 8.10. Then copy and paste the downloaded scripting code from your computer into them to create your own copies of the aforementioned LSL scripts.


Firestorm-Releasex64 4.7.3.47323 - Annabelle SoaS
Avatar Comm World Build Content Help Advanced Developer
Your Parcel, EVWD Book Grid (378, 70, 22) - General
Nearby Chat
Firestorm-Relea...

PHY_Fae_sparkle.png

Name: PHY_Fae_sparkle

Description:

Preview image as: Image

Use lossless compression

Upload (KC0) | Cancel

Upload Model

Model name: PHY_Fae_Boat

This model represents... Moving object (vehicle, animal)

Level of Detail | Physics | Upload options

	Source			Triangles	Vertices
High	Load from file	C:\Users\Annabelle\De	6142	3607	✓
Medium	Generate	Triangle Limit 2047	2044	1212	✓
Low	Generate	Triangle Limit 682	670	439	✓
Lowest	Generate	Triangle Limit 227	222	164	✓

✓ Ship it!

Generate Normals   Crease Angle: 75.000

Calculate weights & fee | Cancel | Clear settings & reset form

Upload fee: KC TBD   Land impact: TBD   Download: TBD   Physics: TBD   Server: TBD

Preview: High

Display... Edges, Physics, [✓ Textures], Skin weights, Joints

Preview Spread: 0.000

FIGURE 8.9 Fairy (fae) boat upload process.

Make sure to use the "select all" option, so you get every bit of Michael's code. Once you have gotten the scripts into your inventory inworld, you will need to make sure that they will call for the correct particle graphic/textures by the specific UUID given to the content when it is uploaded into your virtual environment. To do this, find the PHY_Fae_sparkle.png particle texture in your inventory. Right-click on the graphic and select from the dropdown menu "copy asset UUID." This puts the texture's UUID to your clipboard. Open the sparkle.lsl script, and on line 3, delete the string number between the quotes, and paste in the UUID from the texture in your own inventory. Do the same for the wake.lsl script on line 6 using the PHY_Fae_wake .png graphic from your avatar's inventory. If you want to customize these scripts, simply replace these UUIDs with your own, from textures you have created. OK, now you have everything you need to make a fae boat, and it is all inworld with you, so let's make a vehicle! In Figure 8.10 is a screengrab of this process.

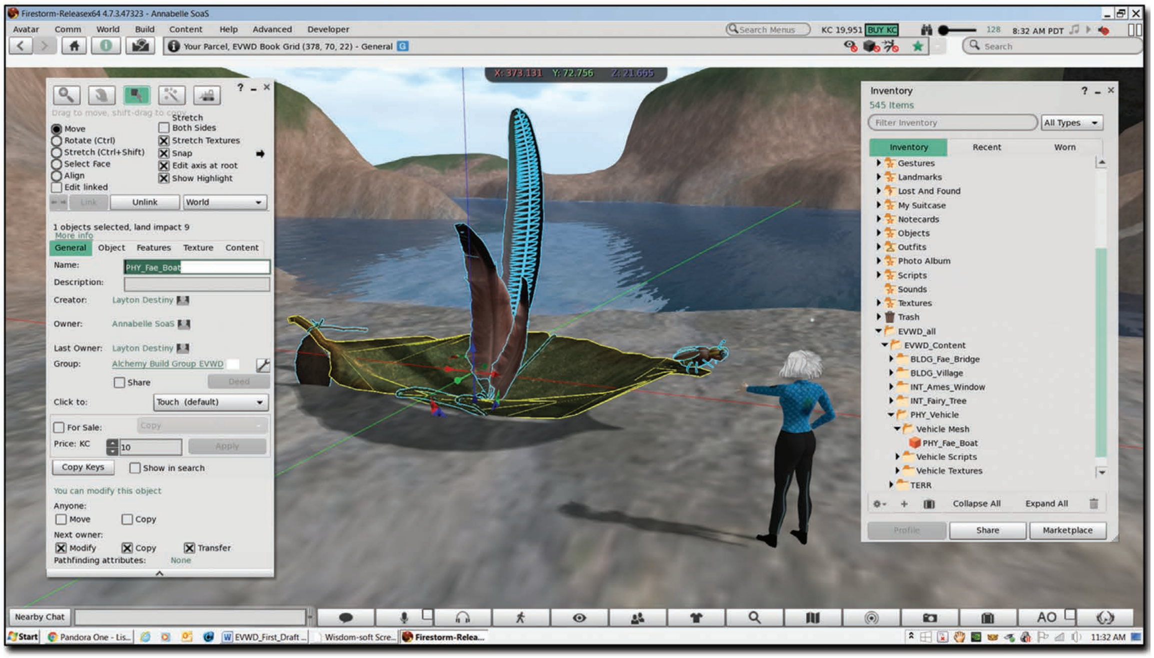
8.8.3 Making the Engine and Particle Generators

The next section is about making the primitives (or objects if you are building this in Second Life) that will contain the scripts that activate and control the vehicle and its particle systems.

Step 1: Make a box primitive, name it Engine, and put the BoatEVWD.lsl script into its contents.

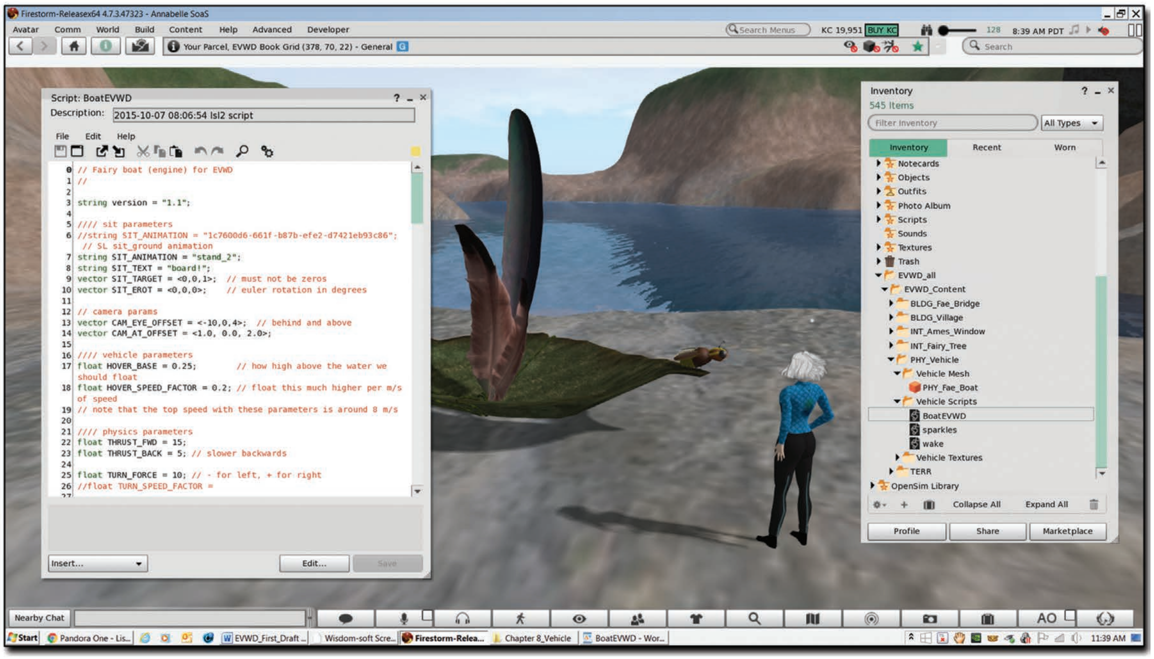
Step 2: About 3 meters away in the x-positive direction (keep it aligned on the y-axis) make a sphere primitive, name it Sparkle, and put the sparkle.lsl script into its contents.

Step 3: About 3 meters away in the x-negative direction (keep it aligned on the y-axis) make a half cone primitive, name it Wake, and put the wake.lsl script into its contents. Set the rotation on this



1 objects selected, land impact 9

Name: PHY_Fae_Boat
Description:
Creator: Layton Destiny
Owner: Annabelle SoaS
Last Owner: Layton Destiny
Group: Alchemy Build Group EVWD
Click to: Touch (default)
Price: KC 10
You can modify this object
Anyone: Move Copy
Next owner: [✓ Modify] [✓ Copy] [✓ Transfer]
Pathfinding attributes: None



Script: BoatEVWD
Description: 2015-10-07 08:06:54 lsl2 script

```
// Fairy boat (engine) for EVWD
//

string version = "1.1";

//// sit parameters
//string SIT_ANIMATION = "1c7600d6-661f-b87b-efe2-d7421eb93c86";
 // SL sit_ground animation
string SIT_ANIMATION = "stand_2";
string SIT_TEXT = "board!";
vector SIT_TARGET = <0,0,1>;  // must not be zeros
vector SIT_EROT = <0,0,0>;    // euler rotation in degrees

// camera params
vector CAM_EYE_OFFSET = <-10,0,4>;  // behind and above
vector CAM_AT_OFFSET = <1.0, 0.0, 2.0>;

//// vehicle parameters
float HOVER_BASE = 0.25;        // how high above the water we
should float
float HOVER_SPEED_FACTOR = 0.2; // float this much higher per m/s
of speed
// note that the top speed with these parameters is around 8 m/s

//// physics parameters
float THRUST_FWD = 15;
float THRUST_BACK = 5; // slower backwards

float TURN_FORCE = 10; // - for left, + for right
//float TURN_SPEED_FACTOR =
```

FIGURE 8.10 Fairy (fae) boat inventory organization and loading scripts inworld.

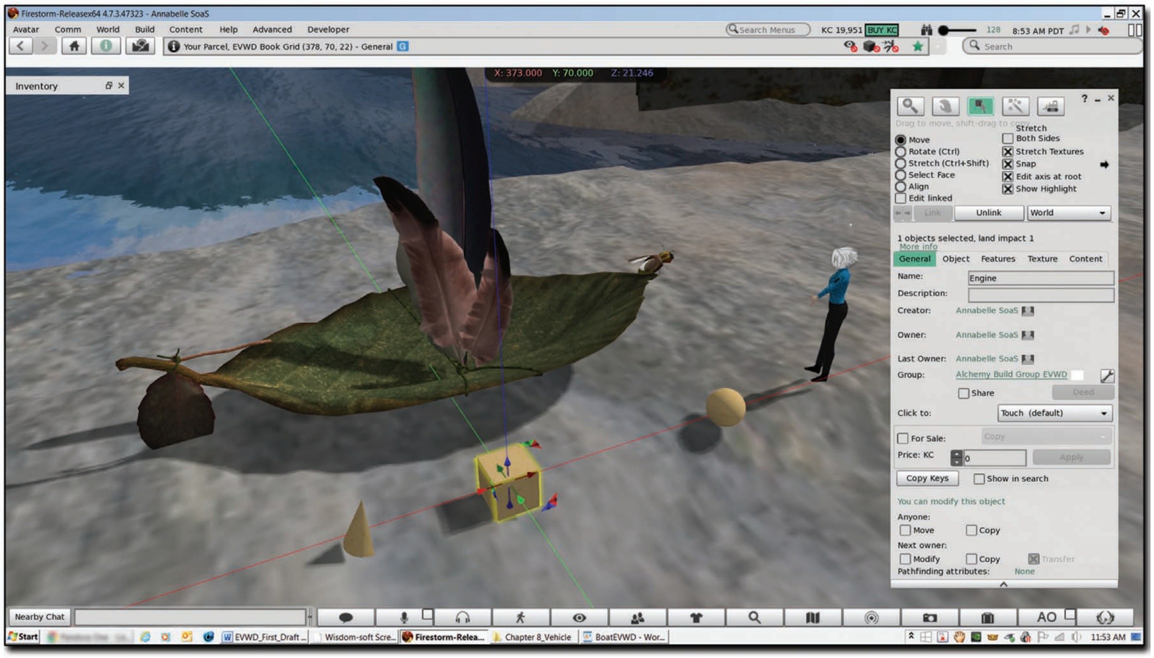

Firestorm-Releasex64 4.7.3.47323 - Annabelle SoaS
Avatar Comm World Build Content Help Advanced Developer
Your Parcel, EVWD Book Grid (378, 70, 22) - General
Inventory
X: 373.000 Y: 70.000 Z: 21.246
Move
Rotate (Ctrl)
Stretch (Ctrl+Shift)
Select Face
Align
Edit linked
Stretch Both Sides
Stretch Textures
Snap
Edit axis at root
Show Highlight
Link
Unlink
World
1 objects selected, land impact 1
General Object Features Texture Content
Name: Engine
Creator: Annabelle SoaS
Owner: Annabelle SoaS
Last Owner: Annabelle SoaS
Group: Alchemy Build Group EVWD
Share
Deed
Click to: Touch (default)
For Sale:
Price: KC 0
Apply
Copy Keys
Show in search
You can modify this object
Anyone:
Move
Copy
Next owner:
Modify
Copy
Transfer
Pathfinding attributes: None
Nearby Chat

FIGURE 8.11 Linking scripted primitives for the fairy (fae) boat.

primitive to X = 0, Y = 270, and Z = 0, so the spray of wake comes from the pointed end and out toward the back of the boat. In Figure 8.11 you can see this arrangement of primitives that will comprise the scripted elements of the fae boat vehicle.

Note: You can use boxes for all three of these. I used different forms so it would be easier to keep track of them as they move around.

Step 4: Link these objects together; make the box or Engine primitive the key prim in the linkset.

Step 5: Right-click on the box (key prim) of the linkset and select “Board!” from the pie menu.

At this step you should see the following things happen: (1) Your avatar will stand on top of the box prim facing in the x-positive direction, (2) the sphere will start emitting sparkle particles, and (3) the half cone will start emitting clouds of wake particles. In Figure 8.12 you can see how this step should look.

Step 6: Fiddle with the settings in the particle scripts until you get the right amount of glow, scale, and so forth.

Step 7: Once you have all the particles looking like you want them to, move your linkset over onto the PHY_Fae_Boat mesh, and position it so that your avatar is standing in a good place in the boat (midway between the feather sails and the tiller), and both the boat and the engine linkset are going in the x-positive direction. Step off the boat to stop the scripts while you make adjustments. (Note: The Engine primitive is where the stand position and the center of the turning point of the boat will be. The standing position [and rotation!] can be changed by altering the values in the llSitTarget call [lines 9 and 10] of the engine script.) You may also want to go into the Edit menu/Edit Linked and push the wake primitive back and the sparkle primitive up. In Figure 8.13 is a screengrab to illustrate this step.

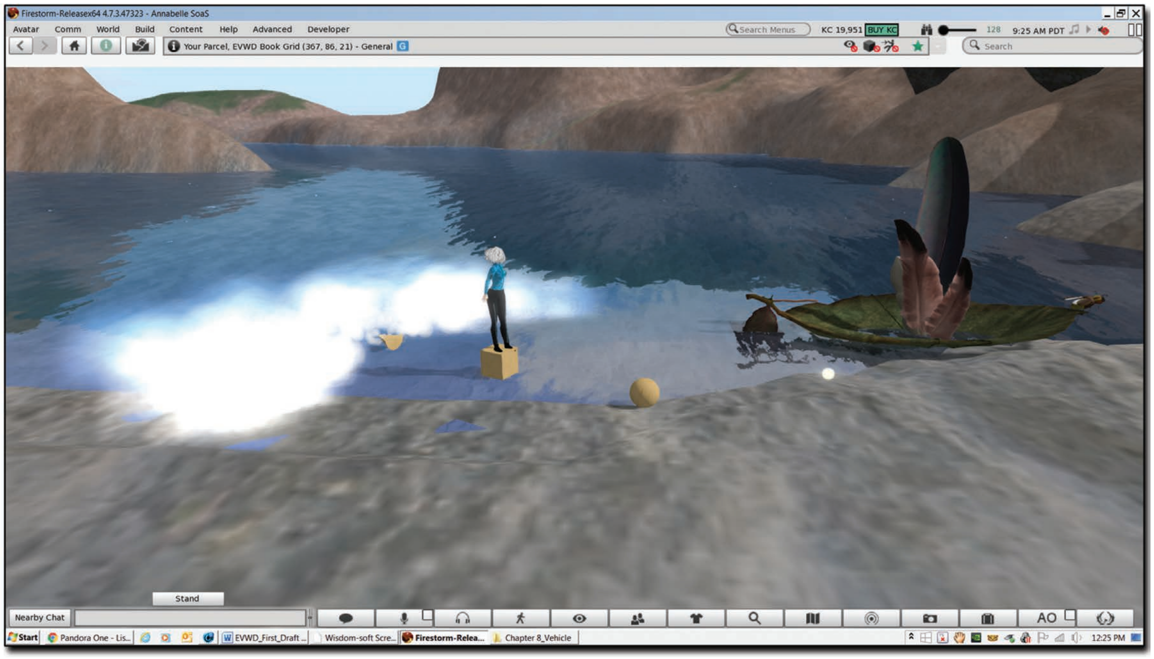
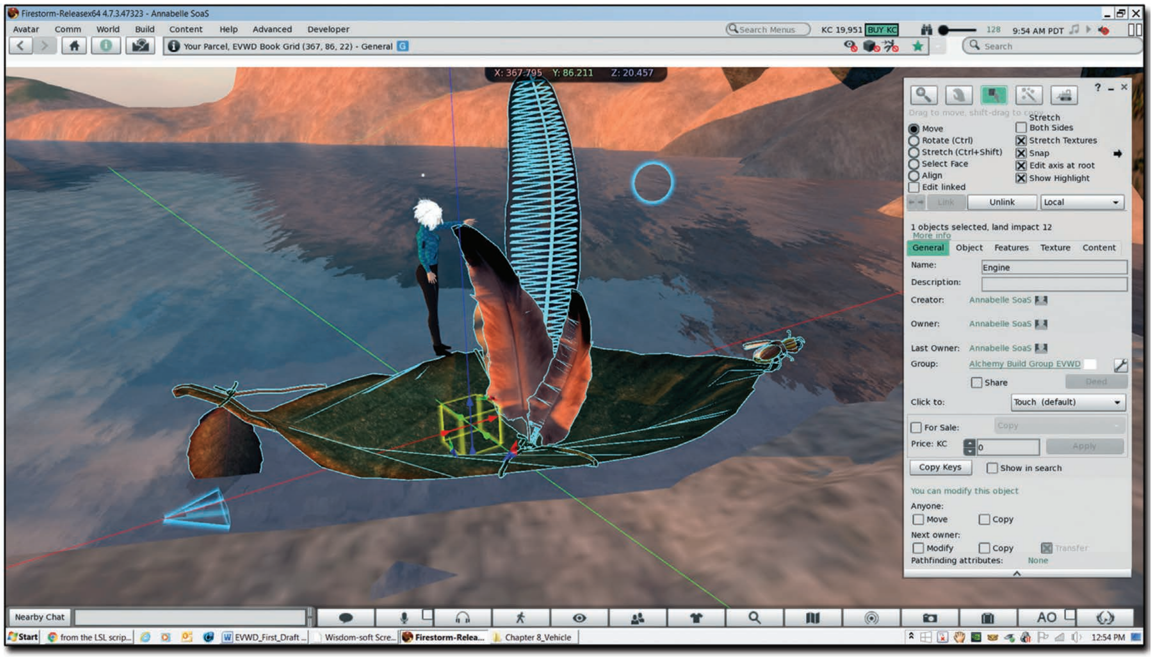


FIGURE 8.12 Fairy (fae) boat linkset with scripts running.


Firestorm-Releasex64 4.7.3.47323 - Annabelle SoaS
Avatar Comm World Build Content Help Advanced Developer
Search Menus
KC 19,951 BUY KC
128 9:54 AM PDT
Your Parcel, EVWD Book Grid (367, 86, 22) - General
Search
X: 367.795 Y: 86.211 Z: 20.457
Nearby Chat
AO
Start
from the LSL scrip...
EVWD_First_Draft...
Wisdom-soft Scre...
Firestorm-Relea...
Chapter 8_Vehicle
12:54 PM

Drag to move, shift-drag to copy

[● Move] Rotate (Ctrl) Stretch (Ctrl+Shift) Select Face Align
☐ Edit linked

Stretch
☐ Both Sides
☒ Stretch Textures
☒ Snap
☒ Edit axis at root
☒ Show Highlight

Link Unlink Local

1 objects selected, land impact 12
More info

General Object Features Texture Content

Name:	Engine
Description:	
Creator:	Annabelle SoaS
Owner:	Annabelle SoaS
Last Owner:	Annabelle SoaS
Group:	Alchemy Build Group EVWD
☐ Share	Deed
Click to:	Touch (default)
☐ For Sale:	Copy
Price: KC	0

Apply

Copy Keys ☐ Show in search

You can modify this object

Anyone:
☐ Move ☐ Copy

Next owner:
☐ Modify ☐ Copy ☒ Transfer

Pathfinding attributes: None

FIGURE 8.13 Avatar positioning on scripted prims of fairy (fae) boat vehicle.

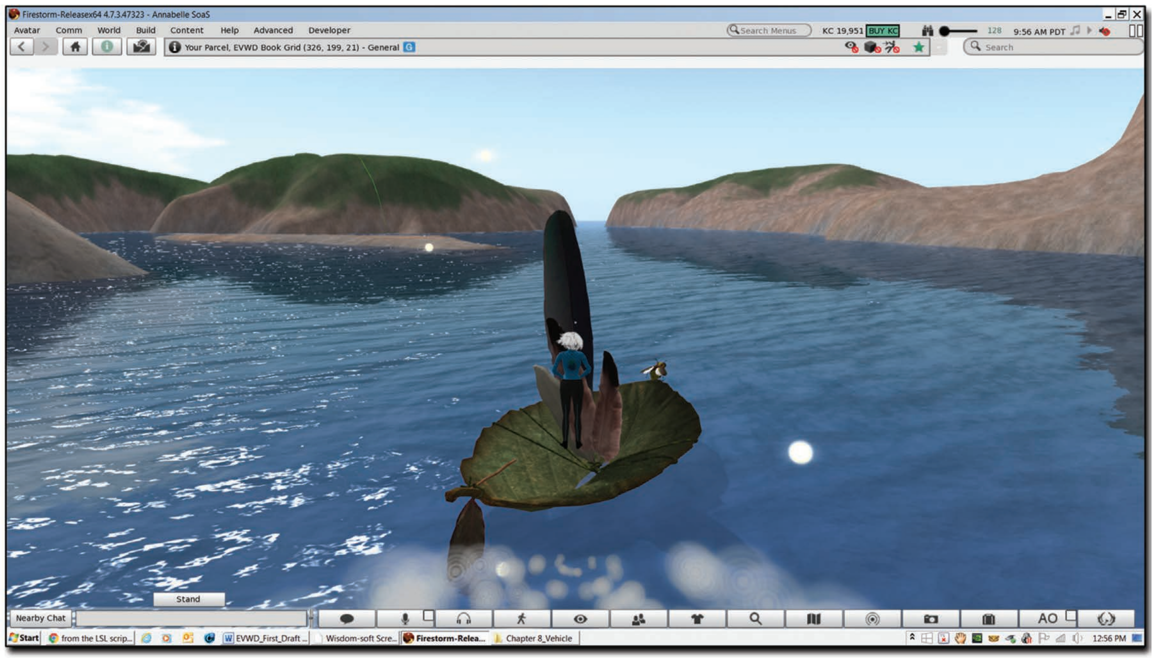

Firestorm-Releasex64 4.7.3.47323 - Annabelle SoaS
Avatar Comm World Build Content Help Advanced Developer
Your Parcel, EVWD Book Grid (326, 199, 21) - General
Stand
Nearby Chat

FIGURE 8.14 Maiden voyage and shakedown cruise of fairy (fae) boat vehicle.

8.8.4 Connecting It All Together and Making a Maiden Voyage

Once you have gotten the prims into the right position, make the Engine/sparkle/wake linkset completely transparent. Select the mesh boat, and link it to the Engine linkset. (Note: Make sure that the engine linkset and the boat linkset are both perfectly aligned on the x-positive axis, so that you move straight in the direction the boat is pointing.) Now you can take it for a test voyage. In Figure 8.14 is a screengrab of the completed boat taking its first voyage across the lake on the game-based sim.

8.8.5 Additional Steps to Consider

You may also want to put in a boat “rezzer” so visitors to your game-based sim can rezz a boat during game play. In that case, the fae boat should be set as a temporary object, so you don’t end up with lots of content to clear off your sim.

8.9 CHAPTER SUMMARY AND FINAL THOUGHTS

This chapter has been a brief overview of virtual physics, physics materials, and how their influence on the real-time virtual environment can impact your design thinking. Taking the time to get to know how your mesh models and designs affect the physics engine supporting your virtual environments will pay off in the end with a smooth running sim. Understanding the basic terminology about kinematics and physical movement will help you work with the scripter to create visually exciting and physically responsive vehicles. As virtual worlds mature and develop their capacity to generate physical behavior from fluids and soft bodies, a new pallet of materials will become available allowing you to expand and enhance your capacities as a designer. Imagine being able to sculpt water or build spaceships from clouds, and viewing them in stereoscopic display. Knowing your virtual physics fundamental principles will make that possible.

REFERENCES

1. *Wikipedia*, s.v. "Brownian Motion," accessed November 22, 2014, http://en.wikipedia.org/w/index.php?title=Brownian_motion&oldid=638884457.
2. *Wikipedia*, s.v. "Physics Engine," accessed November 21, 2014, http://en.wikipedia.org/w/index.php?title=Physics_engine&oldid=632810018.
3. "Physics Engine," *Second Life Wiki*, accessed June 6, 2015, http://wiki.secondlife.com/w/index.php?title=Physics_engine&oldid=91196.
4. "Physics Material Settings Test," *Second Life Wiki*, accessed December 4, 2014, http://wiki.secondlife.com/wiki/Physics_Material_Settings_test.
5. "llCollision Sound," *Second Life Wiki*, accessed December 6, 2014, http://wiki.secondlife.com/wiki/LlCollision Sound.
6. Wikipedia, s.v. "Kinematics," accessed November 23, 2014, http://en.wikipedia.org/w/index.php?title=Kinematics&oldid=636896122.
7. Geoffrey Miller, *Spent: Sex, Evolution, and Consumer Behavior* (Penguin Group, 2009), Kindle edition, 56–57.
8. Geoffrey Miller, *Spent: Sex, Evolution, and Consumer Behavior* (Penguin Group, 2009), Kindle edition, 2213–2216.
9. Linden Lab, "A Look Back to Improvements in Second Life in 2012 and Forward to 2013," *Second Life Blogs*, December 20, 2012, accessed December 1, 2014, http://community.secondlife.com/t5/Featured-News/A-Look-Back-at-Improvements-to-Second-Life-in-2012-and-Forward/ba-p/1775925.
10. Maria Korolov, "Variable Regions Come to OpenSim," *Hypergrid Business*, March 2, 2014, accessed December 1, 2014, http://www.hypergridbusiness.com/2014/03/variable-regions-come-to-opensim/.

This page intentionally left blank

9 Landscape and Terrain Design in Virtual Environments

Dreams are our only geography—our native land.

Dejan Stojanovic

9.1 OVERVIEW OF LANDSCAPE AND TERRAIN DESIGN IN VIRTUAL ENVIRONMENTS

To be a designer of virtual worlds and the landscapes they contain, you must understand the relationship between the terrain and your subjective point of view. Fundamentally, our virtual landscape experience is a duality. It comes from our actual position on the terrain (including the height of our eye level and the draw distance settings in the client viewer), as well as from our emotional/subjective reaction (point of view) to what we see. To develop your design vocabulary and make terrain that will create an emotional impact on the visitors to your virtual environments, you need to experience as many landscapes in the real world as possible and make your own personal notes about how they affect your emotions as your brain processes the visual, auditory, and tactile sensory input from them. At the risk of sounding like a travel brochure, see the massive scale of El Capitan in Yosemite National Park, visit the sand dunes in Colorado or in the Sahara, or take a boat across the delta of the Mississippi or the Ganges River. Various high-level concepts regarding terrain design will be explored in the following sections of this chapter, including these key ideas:

- Inside-out or outside-in; understanding two approaches for designing
- Individual and collaborative storytelling supported with landscape design
- Usage of heightmaps, polygons, and voxels for landscape creation

At the end of this chapter, there is a project in which you will work with a 512 meter × 512 meter terrain heightmap. This landscape will serve as the foundation for the game-based sim environment developed throughout this book.

9.2 LANDSCAPE PERSONALITY AND SOCIAL RELATIONSHIPS TO THE VIRTUAL TERRAIN

On our planet Earth, we have many kinds of terrain that are included in biomes or climates with specific plants and animals. Each of our planet's biomes has a personality; just think of the harsh, brittle beauty of the Arctic or the welcoming softness of a tropical beach. Each type of virtual biome and the terrain it contains can be developed to express a personality that interweaves the visual geographical structures with the plants and animals that you have designed and created. The scale of a landscaping project for a game-based environment can quickly become immense, as you will have to manage the lists of place names, graphic files, heightmaps, and sociomythical connections to the terrain. Efficient use of your time while developing the landscape involves good management of this database. In Table 9.1 you will see a sample

TABLE 9.1
Worksheet for Planning Landscape and Sociomythical Relationships

Project Name: Game-Based Landscape
Designer Name: ________________ Date: __________

	Geographical Structural Type and Relationship to Other Areas in World (Top, Middle, Bottom)	**Type of Interactivity with Avatar (Direct–Physical and/or Indirect–Emotional)**	**Personality of Landscape (Negative, Positive, Neutral)**	**Additional Notes**
Biomes and type of terrain	–	–	–	–
Mountain	Top: Fairy hill	Direct–physical (powerful source of magic in world)	Negative to humans, positive to fairy folk	Contains a stone circle and other magical sites
Mountain range	–	–	–	–
Cliff	Middle: Adjacent to fairy hill and woods	Indirect–emotional (near source of magic)	Neutral to all visitors	Contains dangerous footing
Plain	–	–	–	–
Taiga (snow forest)	–	–	–	–
Glacier	–	–	–	–
Polar regions	–	–	–	–
Forest	Top: Adjacent to fairy hill and castle	Direct–physical (powerful source of magic)	Negative to humans, positive to fairy folk	Contains enchanted beings and landscaping
Rainforest	–	–	–	–
Jungle	–	–	–	–
Woodland	Middle: Adjacent to castle and village	Indirect–emotional (near source of magic)	Neutral to all visitors	Contains enchanted safeguards to protect humans
Shrub land	–	–	–	–
Wetland	Middle: Adjacent to village and shoreline	Indirect–emotional (near source of magic)	Positive to humans, neutral to fairy folk	Contains enchanted safeguards to protect humans
Tundra	–	–	–	–
Desert	–	–	–	–
Coast	Middle: Adjacent to village and lake	Indirect–emotional (near source of magic)	Positive to humans, negative to fairy folk	Contains enchanted safeguards to protect humans
Littoral zone (tidal zone)	High: Adjacent to lake	Direct–physical (powerful source of magic in world)	Positive to humans, negative to fairy folk	Contains enchanted safeguards to protect humans

worksheet that organizes these factors. You can utilize a worksheet like this to help you decide what kinds of physical qualities your virtual terrain will have and how that terrain will relate to the social groups in the world of your game-based sim. In the example in Table 9.1, the landscape of the game is divided between areas friendly to the fairy population (mostly the mountainous and wooded parts) and those areas friendly to the human population. This geographical structure is the physical manifestation (albeit a virtual version) of the emotional relationship that these two main groups will have in the narrative story of the game. It delineates how those physical areas will interconnect with each other and influence the players of the game. If you are world building, you may want to utilize many kinds of biomes and terrains, and if you are region building, you may want to settle with one biome and vary the terrain. Remember, when you are designing a game-based sim, landscape terrain design is not just about elevations and ground covers; it is also about relationships and influence.

9.3 WHAT WORLD ARE YOU FROM?

In my 20 years of professional game design, I have approached world building in two ways. Sometimes, I start with a local setting and then expand it until the world is fleshed out. This is building from the inside out. Other times, I start with a broad framework that is large but shallow, and then focus in on certain areas to give them greater detail. This is building from the outside in.

Chris Pramas [1]

As Chris Pramas, one of the contributing authors of *The Kobold Guide to Worldbuilding*, states in the preceding quote, there are two basic ways to go about creating virtual worlds and their terrain. This is not to say that you always have to use one or another; in fact many projects start with one method and then as they evolve they switch to the other. Perhaps the most useful way to utilize this concept is to decide what your virtual world needs to be on the most basic level first: What kind of planet is it on? Is it earthlike and within the “Goldilocks” zone [2] of humanly habitable environments, or does your world have intense hazardous conditions that require super technology for human survival?

9.3.1 Understanding Believable Terrain by Designing an Alien World

Perhaps the greatest challenge to a virtual environment designer is the creation of an alien world. Until we make contact with other sentient alien life-forms, the audience for what you create will be human and have human-based observational biases. Therefore, what you design needs to be at once familiar to the human observer and yet still seem alien in design. One place to start looking for inspiration for your alien terrain is in the extreme environments of our own Earth. Biologists and astrobiologists study the environments that are inhabited by “extremeophiles” or creatures that can tolerate the extreme ends of our environment’s climate. Earth’s biological life has found viable niches in these extreme climates and conditions, so with some research and a little imaginative extrapolation, you can probably dream up an environment based in the liquid hydrogen seas of Jupiter and design the creatures that live there [3]. In Table 9.2 is a list of some of these extreme climates, their locations on Earth, and main characteristics. The terrains in some of these places have actually been used as scenic backgrounds for science-fiction films and can serve as inspiration for your virtual designs as well [4].

9.3.1.1 Mini-Challenge: Defining and Designing an Alien Environment

Using Table 9.2, pick three of the extreme climates, and try your hand at creating an alien environment for each of them. Make a table and fill in the following requirements:

- What is the gravity like?
- What is the atmosphere like?
- What kind of basic climate and seasons does it have?
- How many animal domains, kingdoms, and phyla are there in these environments?
- Is there sentient life in any of these?

TABLE 9.2
Extreme Climates of Earth: Locations and Main Terrain Characteristics

Type of Environment	Location on Earth	Main Terrain Characteristics
Hot/dry environments	High deserts in the Andes Mountains such as the Atacama Desert	Large barren plain surrounded on all sides by mountain ridges
Hot/wet environments	Hydrothermal vents in the oceans	Ocean floor that has large cracks surrounded by tall sea vents releasing superheated water from crust
Cold/dry environments	Dry desert valleys on Antarctica	Large barren valleys surrounded by snow-covered ridges and frozen ice blocks
Cold/wet environments	Antarctic Ocean seafloor	Sandy, bumpy rolling terrain surrounded by large ice walls
Super salty environments	Dead Sea	Salt-encrusted chunks of asphalt and crystal salt deposits coat all edges and extend into the sea, which lies at the bottom of a long, wide valley
Super alkaline environments	Hot springs	Varying kinds of colorful barren terrain that contain pools of water that are often deeply colored blue or red or green
Super acidic environments	Hot springs	Varying kinds of colorful barren terrain that contain pools of water that are often deeply colored blue or red or green
High-radiation environments	High mountaintops and polar circle	Terrain comprised of the frozen ocean and snow-covered islands
High-pressure environments	Deep-sea bottom and the ocean crust	Terrain varies from flat, sandy plains to rising seamounts

If you get stuck, there are many resources online about exoplanets in our galaxy, some of which are listed in the Important Links and Resources section of this book. From this information, you should be able to invent some alien and yet believable scenarios and life forms. For comparison, look at the climates in Table 9.3; these are the standard terrestrial climates grouped by their latitudes. With the creative thinking and ideas generated in this mini-challenge, you can build the framework of your own alien environment based on a more believable and comprehensive foundation, an environment that makes sense to your Earth-based audience.

TABLE 9.3
Standard Earth Environments: Three Basic Climate Groups

Location	Climate	Terrain
Low latitude (near the equator)	Tropical wet or dry including rainforest, savanna, desert	Terrain includes great river basins such as the Amazon basin, plains of Africa, and southwestern parts of United States
Midlatitude	Environments including steppes, chaparral, grasslands, and deciduous forest	Terrain includes the Great Plains of the United States, the coast of the Mediterranean sea, Eurasian interior, and parts of Eastern Europe and Northern China
High latitude (far from the equator)	Environments including taiga, tundra, and alpine	Terrain includes Alaska, Siberia, Hudson Bay, Rocky Mountains, Himalayan Mountains

9.4 USING YOUR WORLDVIEW AND STORYTELLING TO CREATE A LANDSCAPE/TERRAIN DESIGN

Weltanschauung (worldview) is an interesting word, and it contains a "world" of meaning within its letters [5]. If you think of a worldview as a layer cake of conscious observation, unconscious processing by the visual system, cultural customs, social memories, and artistic interpretation, then you can see that there is a huge source of information at hand that can inform and inspire our virtual landscape designs. Think about the instinctual wayfinding that we do, and spatial memories we carry in our heads every waking day. Think about the symbolic 3D landmark shorthand our brains use to navigate through our daily commute or toddle over to the kitchen for a snack. Try reversing the direction on a familiar path you take, and see how much looks new, how much is being noticed by your brain because it has not been built into your memory map. Your memory utilizes this topographical shorthand to function in daily life, picking out memorable landmarks to guide by, and loading new observations into memory to build a more complete spatial map of the environment in your brain [6]. In fact, studies show that exposure to novel environments, virtual ones included, will even enhance the capacity of our brains to utilize its memories and recall functions [7]. This "memory landscape" could be embedded in a worldview when you consciously link symbolic representation of spatial relationships to the important hierarchies and relationships of your social experience. Build and tap into these connections. Allow the creation of a virtual landscape and its terrain to develop out of your personal spatial guidance system, reflecting your individual voice. Tune into the spatial relationships, cultural customs, and social memories that tint your worldview. Ask yourself these questions:

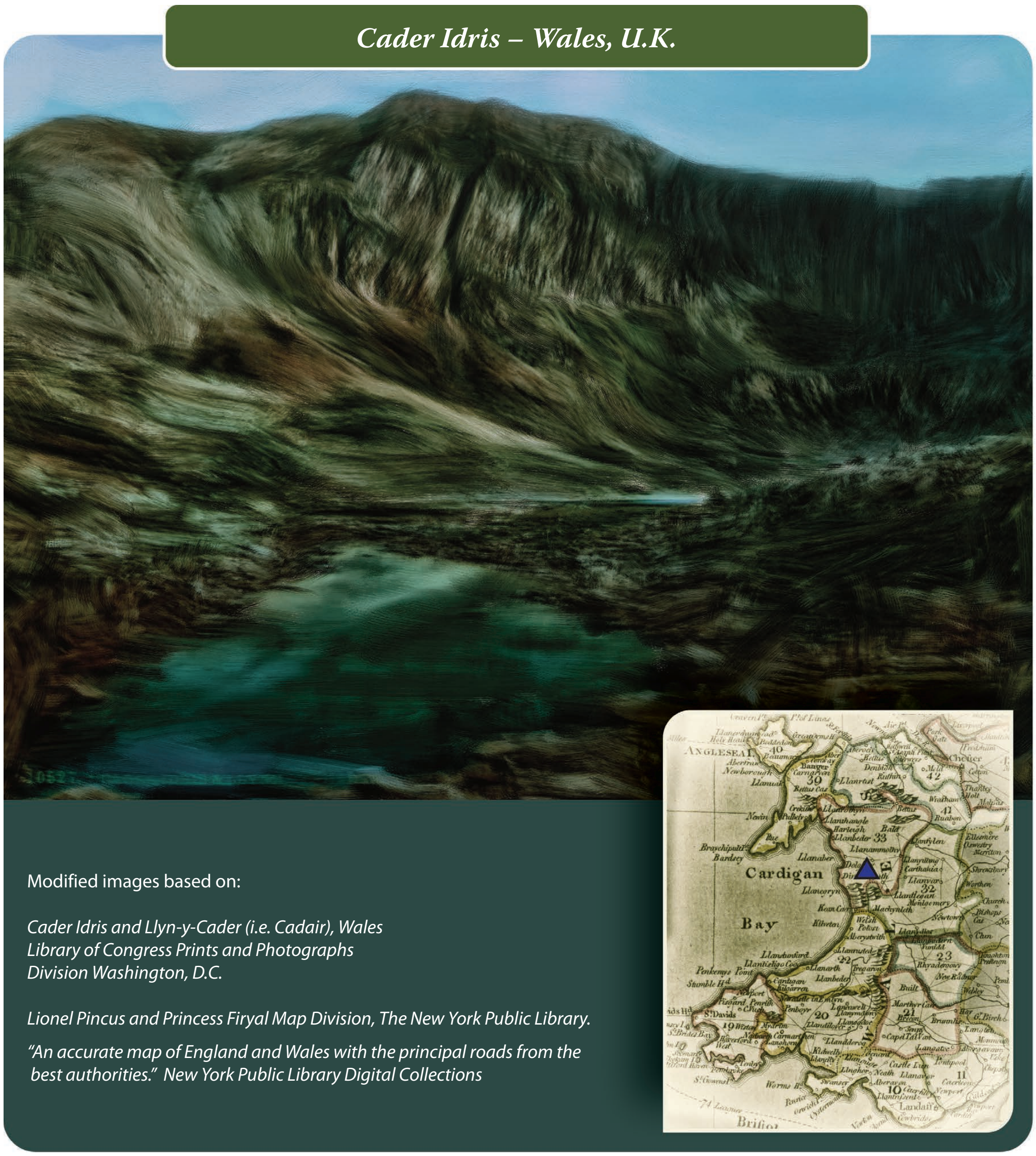
- What are your cultural stories, and how are they intertwined with the spatial relationships in your physical environment?
- What would your virtual terrain look like if you overlay the stories and myths of your virtual world onto the landscape, and what kind of character or personality does it add?
- How would that resonate with the geopolitical needs of the real-world virtual community?
- What are the key aspects of the story you would like to tell through the landscape?

You may decide to create a large valley floor or mountain range that influences avatar movement and perspective through its topography. You may decide that the terrain has to be an interactive surface that loses its solidity or changes its configuration in response to the avatar's presence or vocal command. Your terrain could be a major character, such as the Arabian desert in *Lawrence of Arabia* (the historical epic about E.T. Lawrence, released in 1962 by Horizon Pictures) or the Mississippi River in *The Adventures of Huckleberry Finn* (written by Mark Twain, published 1884). Everything on your virtual landscape should support your story, the narrative, or the meaning of your virtual world. It may be something as humble as the regional flower, or as malevolent as the blackened slopes of the local volcano.

Note: At this point you may want to start working with the creation myth mini-game in Chapter 12, Section 12.2.1. Creation myths tend to reflect not only on a society but also on that society's relationship to the environment.

9.4.1 Landscape/Terrain Design and How It Defines Game Play: Creating the Journey

The people of Wales, UK, have a legend about their mountain called Cader Idris (also spelled Cadair Idris). The Welsh say that someone who sleeps on the slopes of that mountain will awaken the next day as either a



Modified images based on:

Cader Idris and Llyn-y-Cader (i.e. Cadair), Wales
Library of Congress Prints and Photographs
Division Washington, D.C.

Lionel Pincus and Princess Firyal Map Division, The New York Public Library.

"An accurate map of England and Wales with the principal roads from the best authorities." New York Public Library Digital Collections

FIGURE 9.1 Cader Idris and Llyn-y-Cader (i.e. Cadair), Wales, UK. (Based on "Cader Idris and Llyn-y-Cader (i.e. Cadair), Wales," Library of Congress Prints and Photographs Division Washington, D.C. [digital file from original], ppmsc 07384, http://hdl.loc.gov/loc.pnp/ppmsc.07384; "An Accurate Map of England and Wales with the Principal Roads from the Best Authorities," Lionel Pincus and Princess Firyal Map Division, New York Public Library, New York Public Library Digital Collections, http://digitalcollections.nypl.org/items/510d47e1-cbaa-a3d9-e040-e00a18064a99.)

madman or a poet, or never awaken again [8]. In Figure 9.1, take a look at the vintage photo from that mountain area and ask yourself these questions:

- If you were designing a virtual terrain and there was a myth like this about the land, how would it affect your design approach?
- How would you design virtual terrain that looks like it might enrich your speech with poetry or snatch your sanity away in a howling wind?

In Figure 9.1, you can see how the barren slopes of Cader Idris might have inspired this myth. This is an old photo taken between 1890 and 1900. How the wind must howl through those passes during the long dark nights on that lonely landscape tucked away at the southern end of the Snowdonia National Park, near Gwynedd, Wales.

9.4.1.1 Using Visitor Experience Pathways on Your Landscape/Terrain

Let us suppose you decided to take up the Cader Idris story as a basis for your terrain design. A quick check online will show you lots of information about the character of this environment. Several tall mountain forms surround a large lake that was scraped out by a glacier long ago. Few trees occupy the slopes as the biome is in the arctic/alpine category, and the tops of the mountains are covered in scree, or loose stone. The only way to access these peaks is by three hiking trails, which gives this real-world place a timeless quality. One way to approach the myth–landscape terrain idea and how to link that into a virtual landscape design concept is by using the hiking trails to create *visitor experience pathways*. See Figure 9.2 for an example of this idea based on a virtual landscape similar to Cader Idris. On your virtual version of Cader Idris, you can add design features that visibly and audibly reveal the secrets of the local myth to the visitor. For instance, you may block the views off the path with strategically placed boulders, so that the visitor is slowly introduced to the landscape around them, adding to the sense of mystery or enchantment. Or you might build in some twisting turns to those paths that open onto unexpected vistas with dangerously steep embankments to increase the sense of danger or imbalance.

9.4.1.2 Balancing Journey versus Experience Event Frequency: Designing in a Narrative

Let us suppose you decide to add all three trails into your design of a virtual Cader Idris. To get down to the important elements, ask yourself these questions:

- What would you do to enliven the journey along the trails and reinforce the myth–terrain linkage?
- How would you add a sense of foreboding and poetic inspiration?
- How often do you want to remind the visitor that this is a mysterious place?

Since there is little vegetation, the terrain graphic/textures will have to say it all, and they could certainly contain subtle semiology, or mysterious symbols on their surfaces. You can also design in events, such as rockslides and tremors, that will add movement in the rocky scree that covers the hills. You can set up a random plan for these events and the unpredictable *event frequency* will be enjoyed by any observant visitor you have. More obvious objects like Celtic standing stones and strange windswept rock forms will also add to the visual poetry and emotional content. As you arrange these changes in the landscape make sure that they create a different experience on the return trip to the base of the mountain.

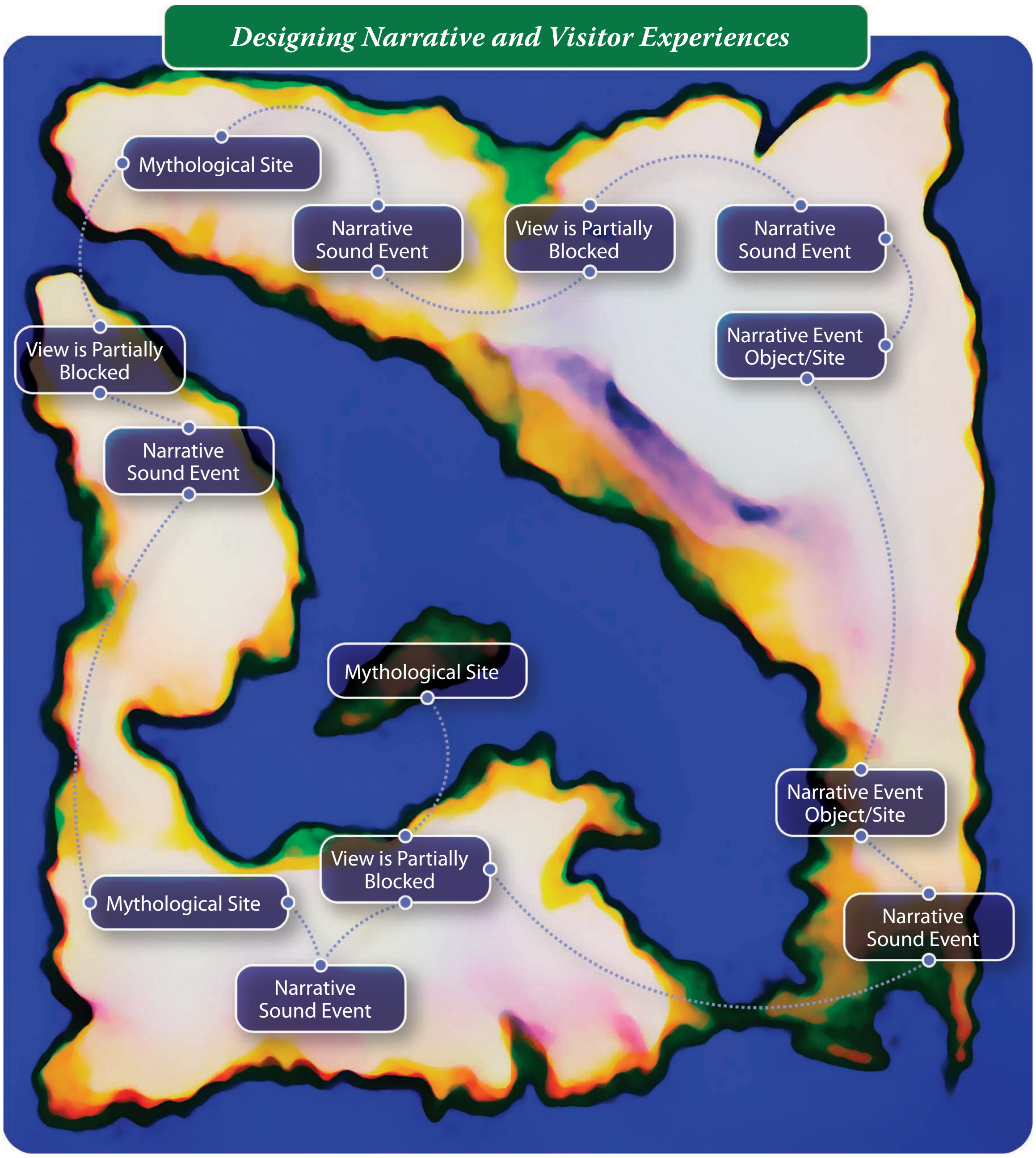

Designing Narrative and Visitor Experiences
Mythological Site
Narrative Sound Event
View is Partially Blocked
Narrative Sound Event
Narrative Event Object/Site
View is Partially Blocked
Narrative Sound Event
Mythological Site
Narrative Event Object/Site
View is Partially Blocked
Mythological Site
Narrative Sound Event
Narrative Sound Event

FIGURE 9.2 Visitor experience pathways in a virtual Cader Idris.

9.4.2 Creating Terrain That Explores Other Planes of Existence

Cader Idris is associated with many Welsh myths, including those containing mention of Gwyn ap Nudd, the king of the fairies (or fae). High mountains with lakes were considered to be his hunting grounds, and the Annwfn (Otherworld), the land under the earth, his home [9]. If you were going to include the Annwfn into your terrain plan, how would you do it? The Otherworld is no mere cave; it is a separate land with royal palaces that provide refuge for the fairy folk. There are several approaches you might consider: You might make the Otherworld a part of your regional terrain that is scripted to only appear at certain times, or perhaps as a separate skybox or different region such that your visitor must use a scripted device (magic!) to teleport to it. The method by which you deliver your visitor to this other plane of existence is not as important as the visual and audible differences you make between the two worlds. For instance, the terrain of the Otherworld should be languorous if your Cader Idris terrain is dynamic. Designing for other planes of existence is very similar to designing alien environments; you need to completely change the visual and auditory framework of the visitor's observation. Important choices you will make as a designer are concerned with the following:

- How fast will you reveal the change in a plane of existence and its related terrain?
- What methods will you use to reveal them?

LSL scripts added to objects that create particle effects, or rezz special mini-environments, or vehicles are some of the methods often used for these kinds of transitions.

9.5 EXPLORING TERRAIN METHODOLOGIES: HEIGHTMAPS, POLYGONS, AND VOXELS

The parameters of virtual terrain have changed considerably in the past few years. In 2002, when Second Life started, the virtual land was divided into square regions (256 meters × 256 meters), and these regions were joined to form "continents," such as the currently existing "Mainland" [10]. In 2014, with the release of OpenSim version 0.8.0, variable sizes were allowed for regions, with scale dimensions always in the power of 2 (i.e., 256, 512, 1024 meters to the side) [11]. Now in OpenSim, you can have a megasized landform that is one continuous region or keep it divided into separate parts. The developers incorporated the large region protocol extensions developed by the team at Aurora Sim (http://aurora-sim.org/), and the Varregion is a standard in OpenSim. This terrain configuration along with the growing use of mesh-based content has opened the door for more creative terrain possibilities. While voxels are not used in OpenSim or Second Life yet, they are being utilized in next-generation MMO (massively multiplayer online) environments, such as *EverQuest* Landmark, as an underlying system that adds dynamic qualities to the terrain. (Note: The creator of *EverQuest* Landmark, Sony Online Entertainment, was bought by Columbus Nova, an investment company in early 2015. Columbus Nova changed the name of SOE to Daybreak Studios, and has shifted most of the development from *EverQuest* Landmark to *EverQuest* Next). This system allows for a "destructible and regenerative world," according to Miguel Cepero of Voxel Farm [12]. Voxel-based terrain makes it possible to create landscape features such as hollow caves and overhanging cliffs, which are not achievable with a heightmap-generated terrain.

9.5.1 Overview of Building Multiregion Terrain with Heightmaps and Meshes

The most common way to create terrain on platforms like OpenSim, Second Life, or Unity is with the terrain editors built into the client viewer or interface. There is lots of information about how to do it that way in Chapter 5 of *Virtual World Design*. For more experienced builders, heightmaps are a good option, especially if you like working in 2D programs such as Photoshop or GIMP, or in 3D terrain programs that will generate heightmaps, such as Terragen (http://www.planetside.co.uk/) and L3DT (http://www.bundysoft.com/L3DT/). Radu Privantu has created a free terrain editor (PC and Linux operating systems only) called Height Map Editor (http://hme.sourceforge.net/), which fills in the sweet spot between 2D and 3D with a simple, easy-to-use interface that can quickly generate heightmaps.

9.5.1.1 Using Mesh Elements in Your Terrain

You may decide that you would like to have even greater control over your terrain, and create some of it using a modeler such as Blender, SketchUp, or 3ds Max. The results of mesh terrain can be stunning and well worth the extra work. In Figure 9.3 is an overview of the fundamentals of making a basic mesh cave element for your virtual terrain. By roughing this shape in SketchUp first, you can get some test models done quickly and try them out inworld to see how they might work. As you can see in the top two images in Figure 9.3, the cave is roughed out on top of a full-scale map of the area it will occupy. You will generate the cave model in real scale, taking up approximately half of a standard region. The cave will be about 60 meters long and 22 meters high inside, so you can start with a box in those dimensions to give you a scale guide. The overall shape needs to be compatible with the height-field generated terrain in that area and the x-ray side view confirms that. When the rough structure is ready, the model can be imported into the OpenSim or Second Life region. Eventually the mesh terrain will be embedded and blended into the surrounding sloping hills. The floor of the cave should be around 20 meters from the ocean bottom, or just slightly lower if you want to see pools of water inside it, as is shown in the final picture on the bottom right. If you do not have a region of your own to work with, you can utilize Diva Distro's OpenSim and download the Sim-on-a-Stick version of the terrain. Downloads for that are available at http://simonastick.com/ or http://metaverseink.com/Downloads.html.

9.5.2 Working with Voxel-Based Terrain

In addition to its great tools for creating a landscape from heightmaps, the Unity game-making middleware has many voxel terrain plugins. A quick search of the Asset Store (https://www.assetstore.unity3d.com/) reveals more than 10 competing products you can try with your Unity terrain project.

9.6 PROJECT: CREATING A FOUR-REGION, GAME-BASED TERRAIN

If you follow the steps given in the next few sections you will discover how to develop a game-based sim environment from concept to grayscale heightmap to actual terrain. The graphics program used in this demo was Photoshop, but any graphics software that will let you paint and draw a grayscale image, slice it into even sections, and save it in the .PNG format is usable. You will also need to get the latest copy of Sim-on-a-Stick (http://simonastick.com/) installed on a high-performance USB stick (the Patriot Rage XT works for me) or have server access to your own OpenSim grid online. The four-region terrain OAR file made for this project was created with Sim-on-a-Stick 0.8 and then loaded into a four-region OpenSim grid running on Kitely .com (https://www.kitely.com/). For more information about how to use OpenSim server commands for terrain manipulation refer to the OpenSim Wiki (http://opensimulator.org/wiki/Server_Commands). OpenSim will utilize appropriately sized grayscale heightmaps with these commands once they have been placed in the /bin folder on the server. You can download copies of all the heightmaps used as well as the final OAR terrain file in the Extending VWD Downloads section of http://www.anncudworthprojects.com/; it is under the Chapter 9 section.

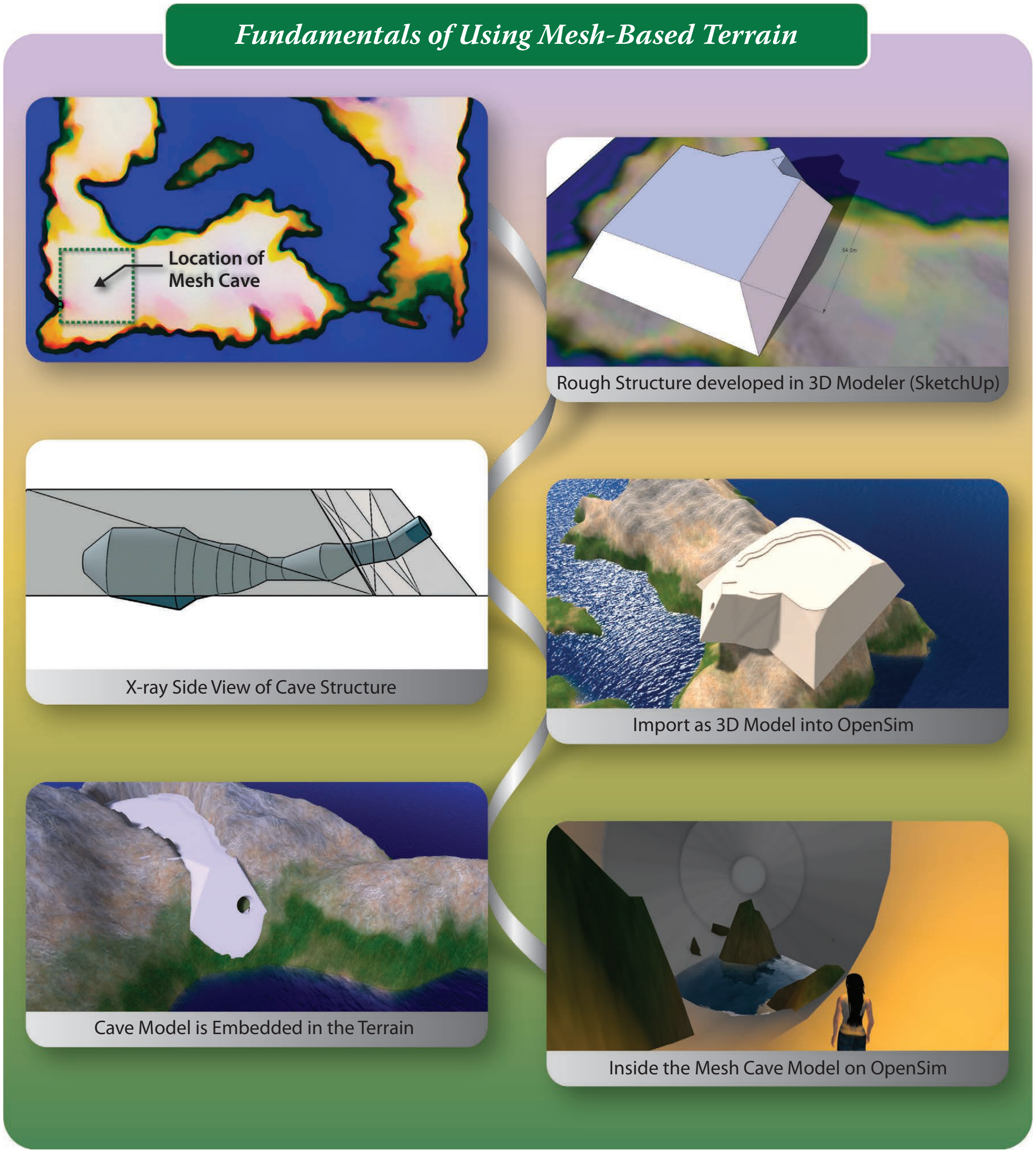

Fundamentals of Using Mesh-Based Terrain
Location of Mesh Cave
Rough Structure developed in 3D Modeler (SketchUp)
X-ray Side View of Cave Structure
Import as 3D Model into OpenSim
Cave Model is Embedded in the Terrain
Inside the Mesh Cave Model on OpenSim

FIGURE 9.3 Fundamentals of using a mesh-based terrain element in OpenSim or Second Life.

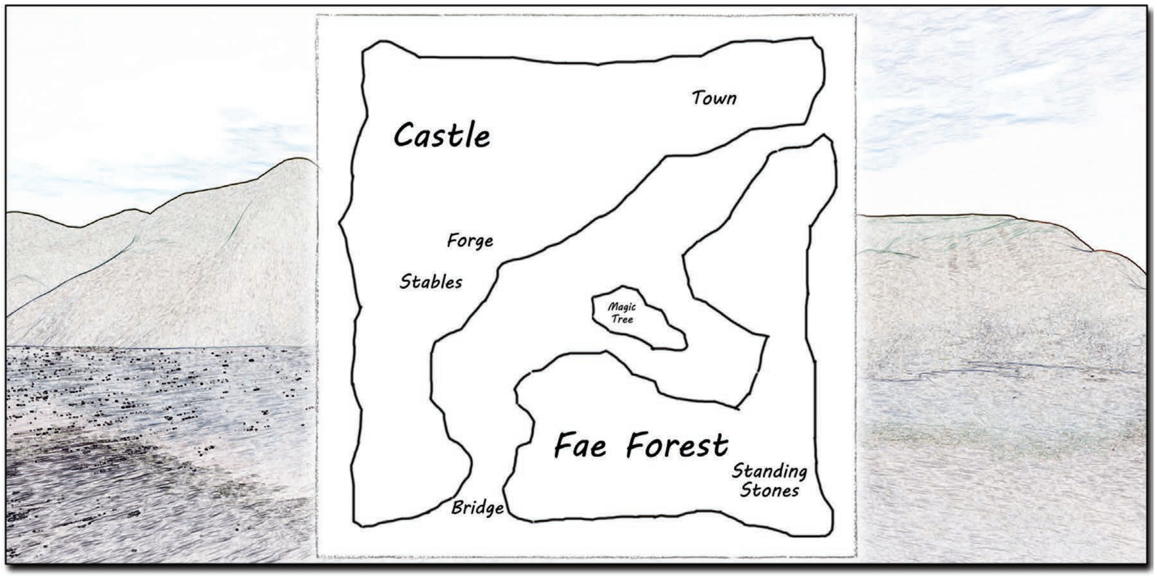


FIGURE 9.4 Hand-drawn map of world.

9.6.1 Step 1: Decide on Overall Concepts and Usage of the Game-Based Terrain

OK, now is your chance for pure creative thought. Do not let that statement intimidate you; dive right in! The first thing you should do is some research. Find images of actual terrain with the right feel and atmosphere for your project. Save all of those images in a file for your constant reference as you proceed through the next steps. For the purposes of our book and this sample terrain project, we referred to the Cader Idris mountain range in Wales, as you can see in Figure 9.1. Continue your terrain development by using Table 9.1 to define and design the interactions that will occur within the game on this terrain. Once you have all of this information organized, make a "working map" for the terrain. This is created by thinking about the games you want to have on your game-based sim and how the terrain arrangements can support them. Pull out a sketchpad and doodle until a rough map of your terrain appears. In Figure 9.4, you can see a hand-drawn diagram that was made for the terrain used in the game-based sim we are building with the projects in this book. This sketch was scanned, opened in a 2D graphics editing program (I used Photoshop), cropped to the proper size (512 by 512 pixels), and became the developmental base layer our four-region heightmap.

9.6.2 Step 2: Creating the Regions in 3D with SketchUp: Alternative Method

If it is easier and better to visualize your terrain in 3D, then try making a rough model in SketchUp (http://www.sketchup.com/). This free 3D program from Trimble Navigation Ltd. is very useful for visualizing all sorts of things for a game-based sim, and the models can even be exported and uploaded into OpenSim. As you proceed through the following steps think about these things:

- Visual cohesion in the terrain
- Game mechanics you are planning to use

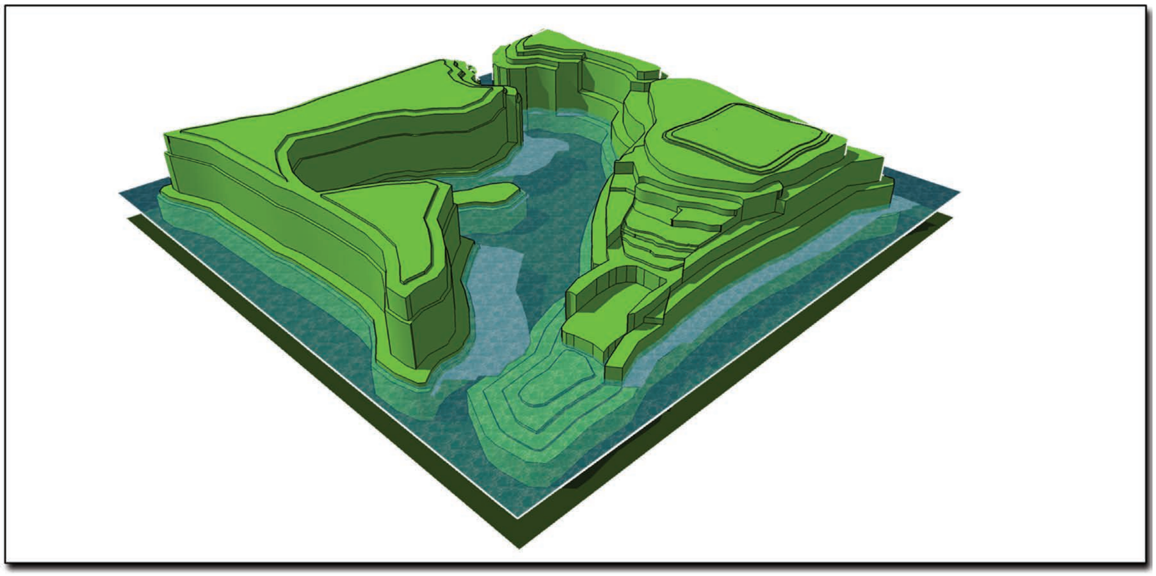


FIGURE 9.5 3D model of terrain in SketchUp.

- Optimal number of regions needed
- Location of region border crossings (these should be in the best spots possible to support the physics engine, i.e., not under the middle of a bridge)

In this case we decided on a four-region grid; the bridge is located on the top corner of one of the regions, and the terrain is full of rolling hills to encourage explorative behavior in the visitors. In Figure 9.5, you can see an overhead screengrab taken from the massing model of the terrain realized in SketchUp. A 2D image of this model was taken into Photoshop and cropped into a 512 pixel by 512 pixel image. This is an alternative way to create the developmental base for a terrain heightmap.

9.6.3 Step 3: Developing Heightmaps for the Terrain in Photoshop

I recommend that you cover the basics of Photoshop with tutorials from Lynda.com or the Adobe site if you are new to that program. If this is the first terrain heightmap you have ever made with Photoshop, please refer to the basic terrain project in my first book, *Virtual World Design*, starting in Section 5.4. Essentially the process is a simple painting in and blending of height values, represented by the black (lowest elevation) to white (highest elevation) scale. I make use of many Gaussian blur filters as well as liberal use of the smudge tool to create smooth, natural-looking landforms. Gradient fills were used to make ramps in the terrain. In Figure 9.6 is a screenshot of the four regions as they look in Photoshop. Remember, this should be made as a grayscale image.

Once you have gotten the terrain heightmap smooth and complete, flatten any and all layers you have made into one grayscale image and save that image as a four-region master file. Then cut it up into four (256 by 256 pixel) regions so you can test them separately in OpenSim or Sim-on-a-Stick. In Figure 9.7 is a screenshot showing what the terrain heightmaps look like as they are loaded into the regions on Sim-on-a-Stick.

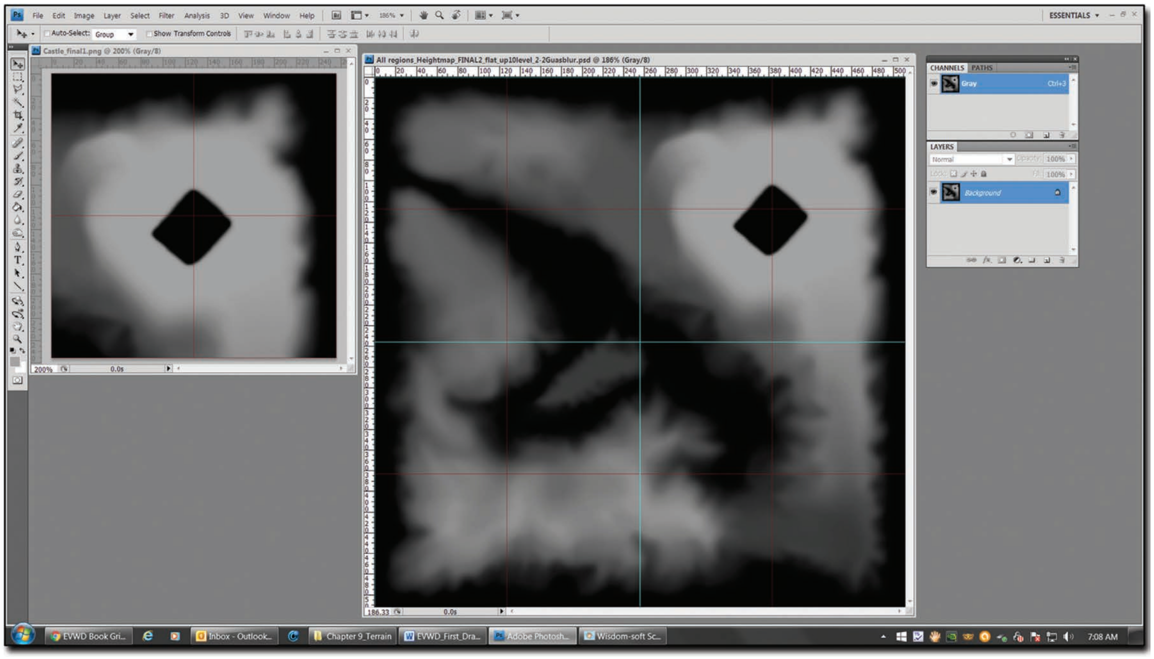

Castle_final1.png @ 200% (Gray/8)
All regions_Heightmap_FINAL2_flat_up10level_2-2Guasblur.psd @ 186% (Gray/8)
CHANNELS
PATHS
Gray
LAYERS
Background

FIGURE 9.6 A four-region heightmap in Photoshop.

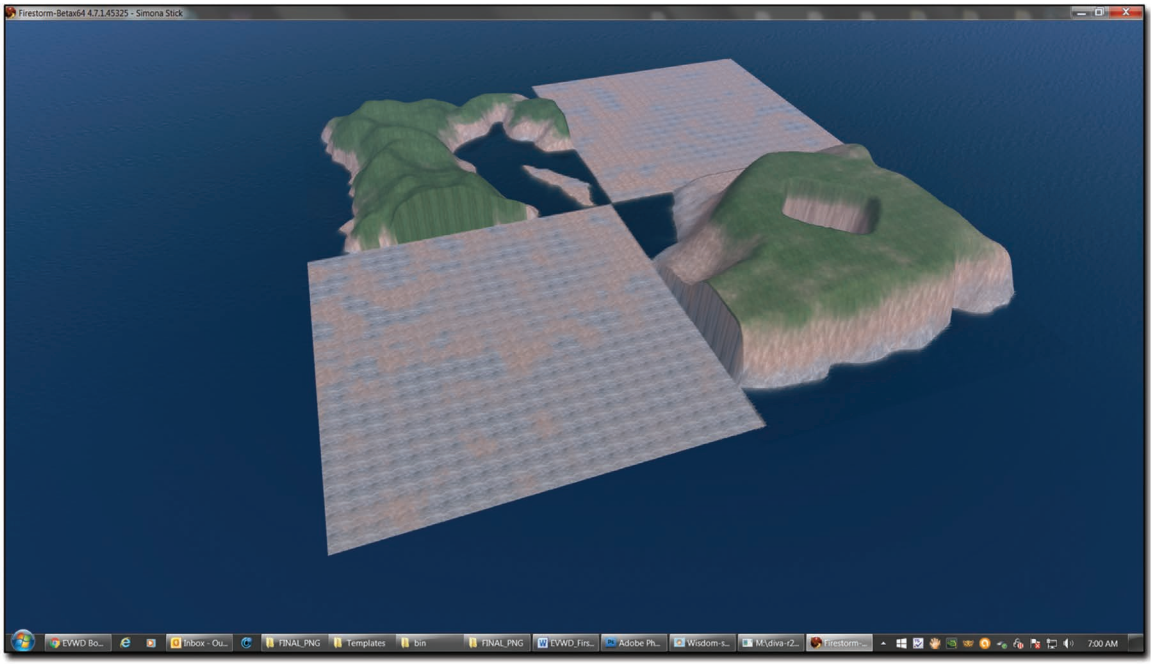


FIGURE 9.7 Heightmaps loading into a Sim-on-a-Stick–based virtual environment.

If the terrain is good but not quite perfect, you can utilize the OpenSim Server commands, mentioned in the overview section of this project, to raise, lower, or reset the entire region. The server commands I find most useful for terrain-making are

- To change from one region to another one, so you can load the heightmap in a specific region: *change region name of the specific region*
- To load in a terrain height map: *terrain load name_of_your_land.png*
- To raise terrain 10 meters: *terrain elevate 10*
- To lower terrain 10 meters: *terrain lower 10*
- To flatten the terrain at the elevation of 25 meters: *terrain fill 25*

If you need to save out an OAR file for loading into another grid, all of the regions can be saved together from the Root region with this command: *save oar - - all name_of_your_land.oar.*

9.6.4 Step 4: Testing the Terrain in OpenSim

In Figure 9.8 is a screenshot of the final terrain loaded into our grid, EVWD on Kitely.com. Because we have a complete four-region terrain OAR file, we can wipe the ground clean at any time and restart with empty land.

In the project for Chapter 10, we will load in and place the mesh building objects and complete the main areas of this game-based environment. During that process, you use the built-in terrain tools to make minor smoothing adjustments in the terrain.

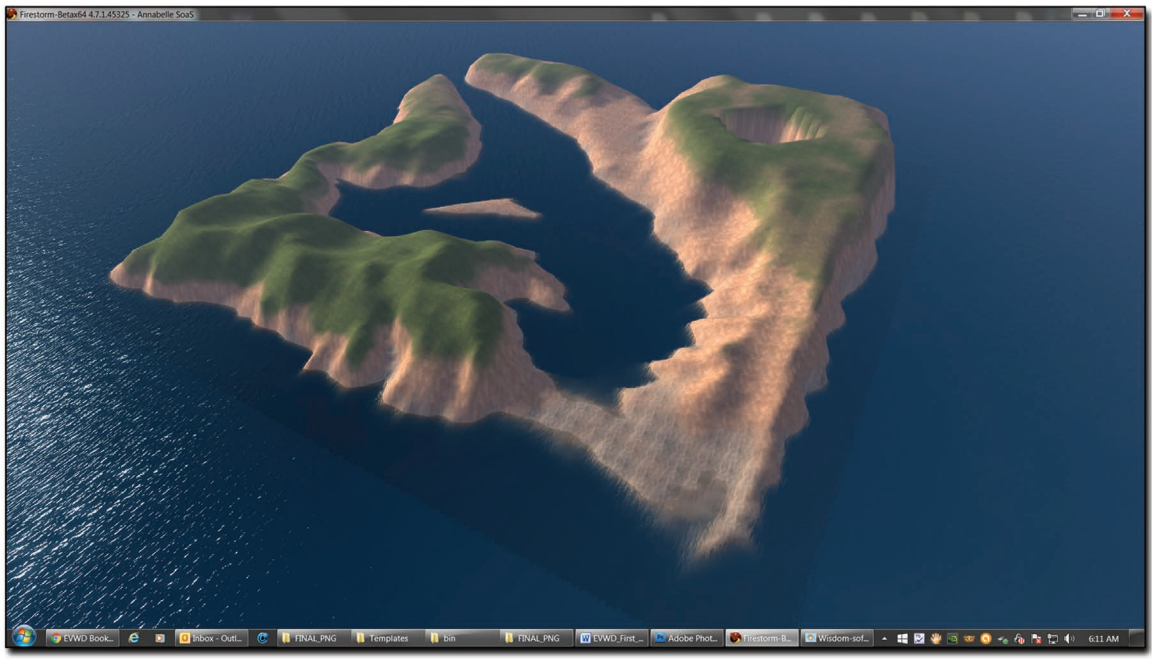


FIGURE 9.8 Final terrain on regions in Kitely grid.

9.7 CHAPTER SUMMARY AND FINAL THOUGHTS

This chapter has been an exploration of how landscape and terrain design can scale with your imagination. On one hand, as a designer, you have to think on a global scale, and on the other hand, you need to honor the details and plan on how all the moving parts will fit together. Terrain is not just a surface you walk on in a virtual environment; it is a crucible into which you pour your observations and sensory interpretations of real-life landscapes. In there, all of this will be amalgamated with the visual and audible stories you want your virtual visitor to experience. In the embrace of a landscape you show and tell about the myths of the peoples who live there, and orchestrate the visitor's attention toward important experiences using techniques like visitor experience pathways and experience event frequency. Our capacities to create terrain with all its complexities including overhanging ledges and caves will continue to expand as more virtual worlds start to utilize meshes. Eventually we may be able to add in real-time erosion and destruction of terrain as voxel-based systems become more widespread. Terrain and the environments it creates stimulates our brains. The relationship between novelty in our visual and physical environment and the increase in our capacity to recall new information has been documented. As a designer you can help reinforce learning and recall by providing a visually exciting and ever-changing landscape in your virtual environment.

REFERENCES

1. Wolfgang Baur, Jeff Grubb, Michael Stackpole, Chris Pramas, Keith Baker, Steven Winter, and Jonathan Roberts, *Kobold Guide to Worldbuilding* (Kobold Guides to Game Design) (Open Design, 2013), Kindle edition, 260–264.
2. "Eight New Planets found in 'Goldilocks' Zone," *Harvard-Smithsonian Center for Astrophysics*, January 6, 2015, accessed January 6, 2015, http://www.cfa.harvard.edu/news/2015-04.
3. Ron Miller, "The Most Dangerous Places in the Solar System," *io9* (blog), February 27, 2012, accessed January 6, 2015, http://io9.com/5888775/the-most-dangerous-places-in-the-solar-system.
4. Tony Reeves, *The Worldwide Guide to Movie Locations* (Titan Books, 2006), http://www.movie-locations.com/book.html.
5. *Wikipedia*, s.v. "World View," accessed January 19, 2015, http://en.wikipedia.org/w/index.php?title=World_view&oldid=642672222.
6. Timothy P. McNamara and Amy L. Shelton, "Cognitive Maps and the Hippocampus," accessed June 7, 2015, http://www.psy.vanderbilt.edu/faculty/mcnamara/lab/CogMap.pdf.
7. Judith Schomaker, Marthe L.V. van Bronkhorst, and Martijn Meeter, "Exploring a Novel Environment Improves Motivation and Promotes Recall of Words," *Frontiers in Psychology* 5 (August 2014): 918, doi: 10.3389/fpsyg.2014.00918.
8. "Myths and Legends," *Snowdonia: Mountains and Coast*, Heritage and Culture of Wales, accessed January 17, 2015, http://www.visitsnowdonia.info/myths_and_legends-89.aspx.
9. Sioned Davies, *The Mabinogion* (Oxford World's Classics) (Oxford University Press, 2007), Kindle edition, 6327–6330.
10. "History of Second Life," *Second Life Wiki*, July 5, 2014, 13:26 UTC, accessed May 17, 2014, http://wiki.secondlife.com/w/index.php?title=History_of_Second_Life&oldid=1191745.
11. Maria Korolov, "Variable Regions Come to OpenSim," *Hypergrid Business*, March 2, 2014, accessed May 17, 2014, http://www.hypergridbusiness.com/2014/03/variable-regions-come-to-opensim/.
12. Fantomex, "An Interview with Miguel Cepero of Voxel Farm," *Nexus Blog*, October 2014, accessed January 19, 2015, http://eqnexus.com/2014/10/interview-miguel-cepero-voxel-farm.

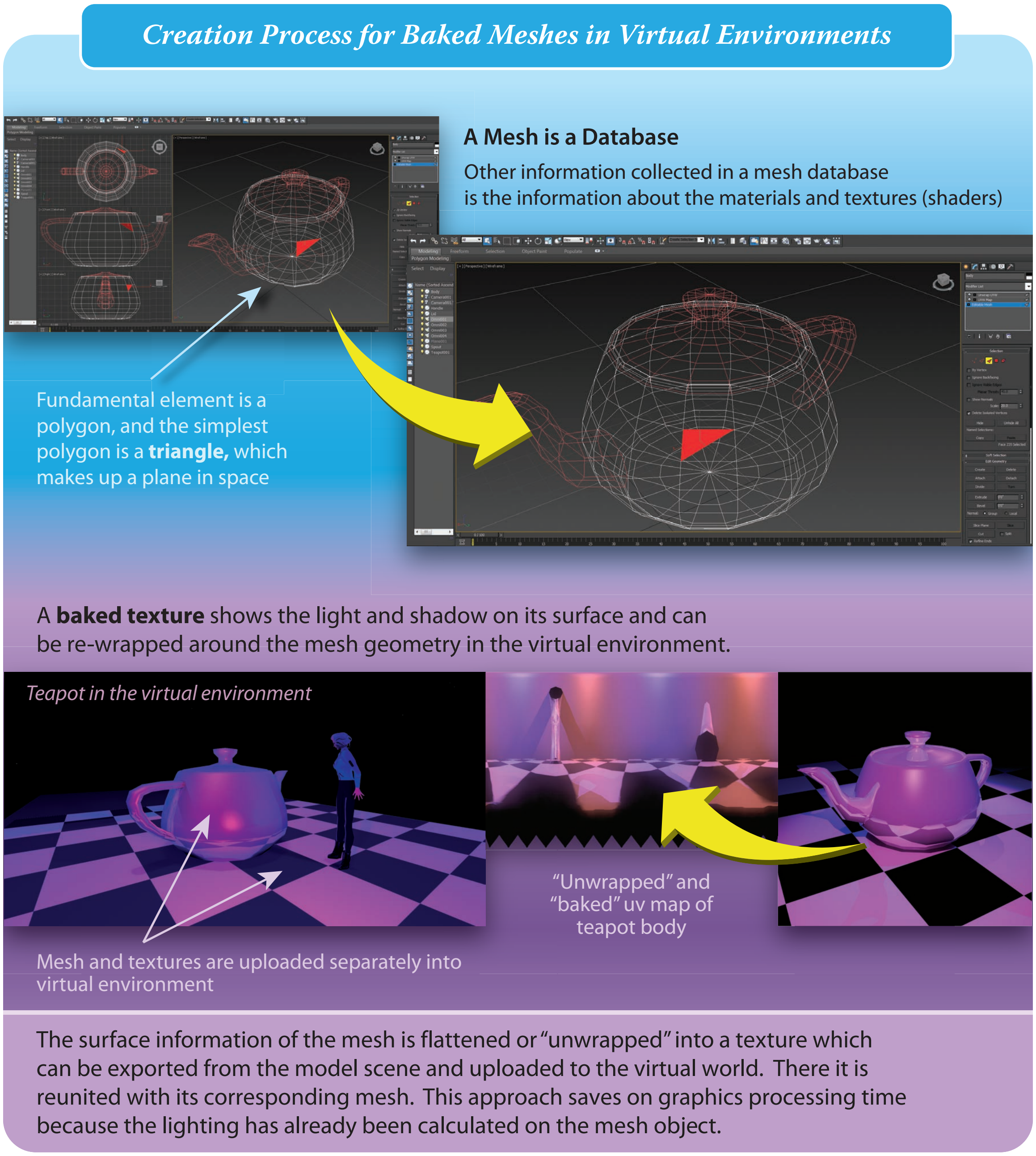
10 Design Considerations and Mesh Usage in Virtual Environments

Good design is making something intelligible and memorable. Great design is making something memorable and meaningful.

Dieter Rams

10.1 OVERVIEW OF DESIGN CONSIDERATIONS AND MESH USAGE IN VIRTUAL ENVIRONMENTS

As a designer of virtual worlds you will probably spend a great deal of time thinking about meshes. Meshes comprise a significant portion of a virtual environment, and their appearance and physical behavior in response to an avatar's actions is crucial to creating an immersive sense of space for the visitor. Now, please understand that a highly detailed, almost realistic version of some real-world scene that you built for your game-based virtual environment may go mostly unnoticed by your visitor. This is not a bad thing; in fact, the ability to create an environment that is easily understandable and supports an instinctual interaction for the visitor is the hallmark of a good designer. A familiar environment clears the way for good communication too, because it sets up a common frame of reference for all the visitors. Of course, you may want to design an alien–fantasy environment that comes directly from your imagination and has elements not included in our typical real-world environment. If you wind the constructs of the alien–fantasy world around your core awareness of real-world design, so that the visitor is at once experiencing the alien and familiar, then your overall design will become deeply immersive. They will be thinking, "So this is how the aliens create a meeting space. I understand its function, but how strangely interesting and different it is!" This is what meshes are; they are the fabric of your imagination. But, what is a mesh actually? It is a data set that contains spatial and visual information about the elements within it. The simplest element of a mesh is a polygon, or many-sided plane. Every polygon in your mesh is subdivided into triangles, which, as you know from basic geometry, contain the smallest number of vertices necessary to determine a plane in space or a face on a polygonal mesh. The information about where these vertices are located in 3D space, data about the kinds of materials and shaders on the faces (triangles) of each part of the mesh, and how each face is lit or shaded by a light source in the scene are all part of the mesh data set that is passed on to the rendering engine in your computer. In Figure 10.1 is a diagram showing the structure of a polygonal mesh (the classic teapot) and how all of its parts work together to create a mesh model in your virtual world. Notice how the triangles form the basic elements of every mesh, no matter how complex. In the lower half of Figure 10.1, note how the Texture (diffuse) graphic of a mesh can be created from a process called *baking*, which calculates the lighting and reflections on the entire surface and renders it out onto a flat graphic, which is also known as a UV map (UV coordinates are the XY equivalents on graphics, maps, and textures in a 3D modeler). This baked map will be wrapped back onto the mesh once it is imported into your virtual environment. This is the best way to create a realistic look with the least amount of real-time processing,



Creation Process for Baked Meshes in Virtual Environments

A Mesh is a Database

Other information collected in a mesh database is the information about the materials and textures (shaders)

Fundamental element is a polygon, and the simplest polygon is a **triangle,** which makes up a plane in space

A **baked texture** shows the light and shadow on its surface and can be re-wrapped around the mesh geometry in the virtual environment.

Teapot in the virtual environment

"Unwrapped" and "baked" uv map of teapot body

Mesh and textures are uploaded separately into virtual environment

The surface information of the mesh is flattened or "unwrapped" into a texture which can be exported from the model scene and uploaded to the virtual world. There it is reunited with its corresponding mesh. This approach saves on graphics processing time because the lighting has already been calculated on the mesh object.

FIGURE 10.1 Fundamental concepts of meshes and UV map/baked graphic/textures.

especially when combined with Bumpiness (normal) and Shininess (specular) graphic/textures, which are discussed in Chapter 11.

Mesh usage in virtual environments has grown rapidly since October 2010 when the Second Life and OpenSim code and interfaces such as Firestorm and Second Life Viewers began to support it [1]. All mesh is brought into these virtual environments via the Collada.dae (digital assets exchange) file format, which is almost universally supported by 3D modeling programs [2]. While it is vastly rewarding to use meshes in your virtual environment, the inclusion of them can be a complex process and should be approached with an organized workflow plan. The basic steps to organize this workflow should include the following:

- Make a list and/or do a series of sketches to define the scope of the project, with a comprehensive naming system that keeps all the parts organized into related groups. For instance, as you can see in Appendix A, Section A.6, the components for the game-based sim will all have special prefixes that identify their group type.
- Decide what mesh elements (the inworld primitives) of your design are going to be made with the built-in tools of the virtual environment and what mesh elements are going to be made by an outside 3D modeling program.
- Make sure everyone on your team is working in compatible file formats so that they can exchange models and graphic/textures back and forth.
- Test your workflow between team members, exporting into and out of the 3D modelers and the virtual environment, to discover any snags or pitfalls.
- Allow specialists to develop. While everyone on the team should be able to work with basic models, some folks may have a knack for lighting or perhaps rigged mesh creation, and you should let them specialize in that if they want to.

In this chapter, you will explore the usage of meshes and the following "meshadologies":

- Meshes can come from many sources and knowledge of the various methods for obtaining them is useful for virtual world building.
- There is a special branch of meshes called "rigged mesh" that is used for moving avatars and the clothing they wear.
- 3D model or mesh creating software should be chosen to support your workflow on every project, and the naming systems should be decided early on to keep everything compatible.

Since this chapter does not concern itself with how to use 3D modelers to make meshes for virtual environments, you will be working with premade meshes that are part of the 3D content provided with this book. At the end of this chapter, you will upload the elements for a castle, its outbuildings, and the local village nearby. For those who are working in OpenSim, this will be via the OAR format, and for those who are working in Second Life, these elements will be available inworld for you to arrange on your own regions. This project is about how to approach a large design challenge, and how to utilize plans and diagrams, so the team can work in a coordinated fashion.

10.2 SOURCES FOR MESH IN A VIRTUAL ENVIRONMENT

Where do we get meshes from? Here is a list of the most prevalent sources:

- Primitives or objects we rezz inworld using the Build/Create menu in the client viewer
- Construct our own with a 3D modeling program

- Generate complex meshes from procedural programs that run as scripts in your 3D modeling program
- Create them from 3D sculpting programs
- Construction of a mesh object from 3D information collected via user-assisted scanning or photogrammetry [3]
- Websites that supply 3D meshes for purchase. (Note that you must comply with the licensing terms of use for these meshes, and some sites do not allow usage of their content in virtual worlds like Second Life.)

In Sections 10.2.1 to 10.2.5, we will look at these mesh acquisition methods in more detail. Knowing how to choose the right source of 3D meshes can save you time and effort when you are creating a large game-based virtual environment.

10.2.1 Some Considerations regarding the Standard 8 Inworld Mesh Objects

As you probably know, the client viewer for virtual worlds already provides you with a nice selection of 3D meshes, which are illustrated in Figure 10.2. They are known as "objects" in Second Life, and "primitives" in Open Sim, and commonly called "prims" in most virtual worlds. Although they have been available since the early days of virtual worlds, these native components are still very useful for building and should not be discounted as out-of-date methodology. There are a few caveats that you should bear in mind:

- If you are working in a virtual world that charges you for import of mesh and you have a small budget, consider using the native mesh objects as much as possible for your virtual environment to save on import fees.
- Alternatively, you can test your meshes on a beta grid for free, such as Aditi in Second Life.
- If you are trying to keep the number of faces and land impact to a minimum, you can build your meshes with fewer triangles in an outside 3D modeling program, such as Blender or 3ds Max. The native objects have more triangles per face to allow for the manipulation of their appearance with the built-in editing modifiers provided by the client viewers, such as Firestorm or Second Life Viewer.

10.2.2 Some Considerations regarding Meshes Created in 3D Modelers

If you want to have custom meshes such as the content distributed with this book, you will need to learn the program interface and building techniques for a 3D modeling program, such as Blender or 3ds Max. There are many options in program sophistication (and related price point) as shown in the Second Life Wiki at http://wiki.secondlife.com/wiki/Mesh/Tools. In Section 10.4 is a guide to help you decide what program you should choose for your main tool. In this section let us review what these kinds of meshes bring to your virtual environment design. In phase 1, the *prebuild phase* of your project when you are defining the scope and look of your game-based sim, usage of meshes can

- Provide the initial 3D visualization of your game-based sim with the use of a 3D modeler like SketchUp Pro
- Generate documents useful for your build team and game programmers from a program such as Layout, which interfaces with your models in SketchUp Pro
- Make fly-through or walk-through previews of the virtual environment that is being designed

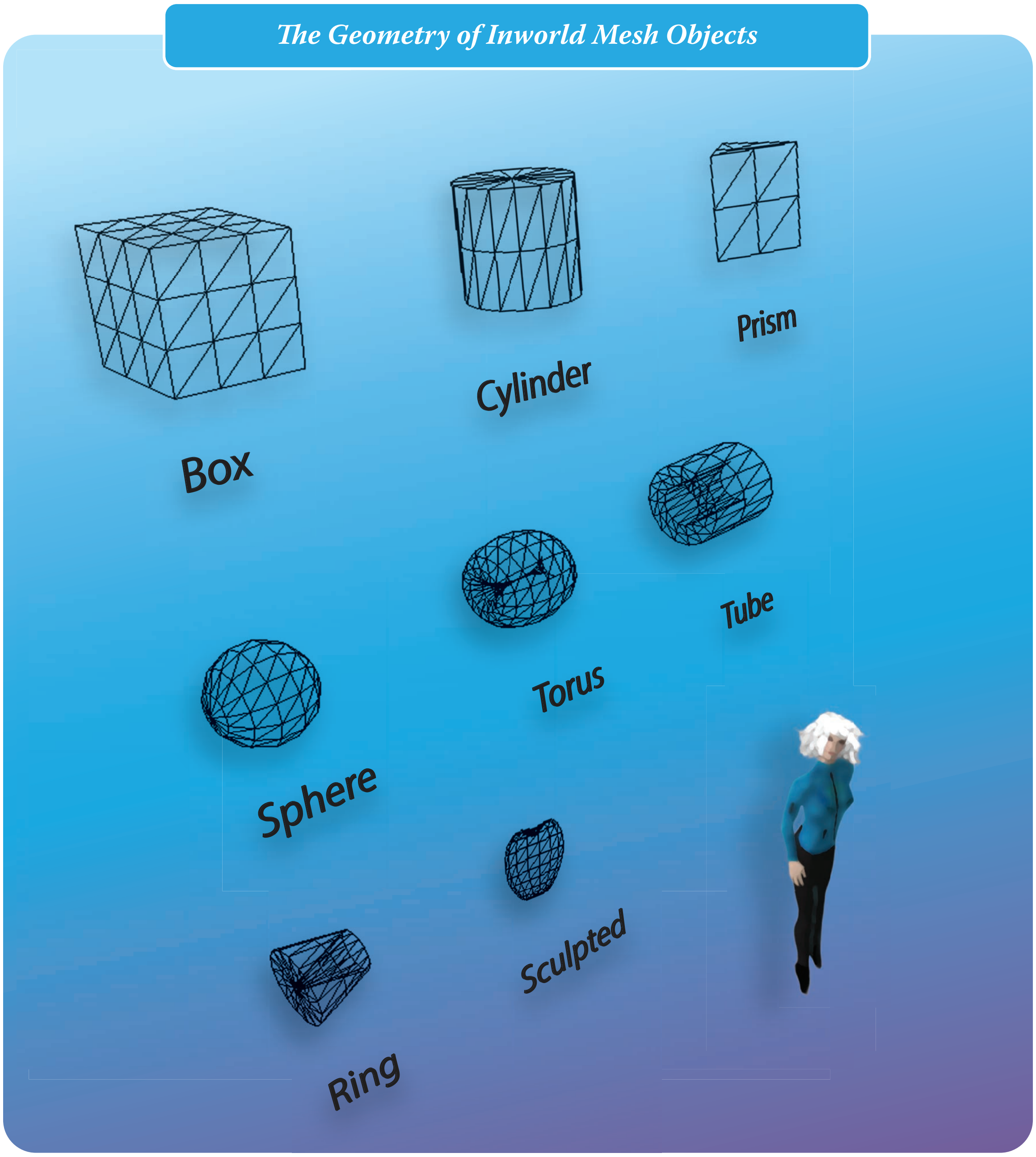


FIGURE 10.2 The eight types of inworld mesh objects provided in the client viewer.

In phase 2, the *building phase* of your virtual environment, these mesh modelers become the workhorses for content creation for all members of your team, and are involved with the creation of

- 3D content that generates the terrain heightmaps as well as 3D landscape elements
- Architecture and space-defining elements in the virtual environment
- Objects that are worn or held by the visitor's avatar, such as clothing, weapons, or props

Finally, in phase 3, the *extension phase* of your build, meshes can help you extend your game-based sim into the real world. These meshes can be utilized as

- 3D models that you have used in the virtual environment can be repurposed to make real printed objects for promotions or a real-world aspect to the game play in your sim
- 3D models repurposed for augmented reality to create a mixed reality experience based on the content in your virtual world

In Table 10.1 is an overview of the general procedures used for importing or uploading meshes from three programs into a virtual world. This is intended as an overall guide to the principles of this process; each step requires a good knowledge of the modeling software used and you may have to experiment with it a bit to get the settings for your particular system and personal workflow just right. As you can see, SketchUp does not provide baked graphic/textures, as it has no native lighting system. However, you can add photographic images to the faces of the SketchUp mesh for a realistic look.

10.2.3 Some Considerations regarding Mesh and Procedural Generation

Procedural generation of mesh is a way of combining two forms of building. When you use this approach, the standard meshes you have created in your 3D modeling program are augmented and embellished with a plugin program developed in the scripting language of your 3D modeler. These kinds of plugins are extremely useful when you have to build a mesh model containing many similar but repetitive parts, such as a cityscape or a weapons arsenal. If you are an experienced user of 3D modeling programs like 3ds Max or Blender, you are probably cognizant of the existence of scripts that work within these programs. Although the number of free scripts for procedural generation is limited, some do exist and are available to you for download and usage. You will find Blender scripts on the Blender Nation site (http://www.blendernation.com) and 3ds Max scripts at the Script Spot site (http://www.scriptspot.com/3ds-max/scripts). Of course, you can also try writing your own procedural program plugins in the native scripting languages of Blender (Python) and 3ds Max (MAXScript). As you can see in Chapter 12, Section 12.4, there are also fractal-based programs that will let you develop mesh within your virtual environment and cloning scripts that you can use to create non-player characters (NPCs) for your game-based sim. In Chapter 12, you will have the opportunity to work with a fractal-based procedural script within the virtual environment by making a procedurally based fairy tree. Here are some key questions and answers to consider about procedural generation of meshes right from the very beginning of your virtual environment design project:

Question: What are the most likely candidates in your virtual environment for procedural generation?
Answer: Terrain, cityscapes, vehicles, and weapons would be most prevalent.
Question: What kind of procedural generation methods and tools do you have access to?
Answer: Scripters who can code for the 3D model program, some purchased procedural programs, and online procedural creation programs.

TABLE 10.1
Overview of Mesh Creation for Uploads: Comparing 3ds Max, Blender, and SketchUp

Procedural	3D Modeling Software Used			
Steps	**3ds Max**	**Blender**	**SketchUp**	**Notes**
1	Make or obtain mesh model.	Make or obtain mesh model.	Make or obtain mesh model.	
2	Make or obtain a material (and related graphics) for the model and then light the model so that it relates to the virtual environment scene.	Mark seams for UV unwrap.	Apply graphics to the faces of the model.	Consider what kind of real-time lighting you will be using in the virtual environment as you light this model.
3	Apply the material (and graphics) using UV mapping if the geometry is a simple form.	Select all vertices and press U and select your unwrap method. (I would usually just use the Unwrap option for most applications, although you may then want to adjust and tweak the result to maximize use of the graphic space and ensure that your material maps correctly.)	Export mesh as COLLADA (*.dae); set the export to include textures. Textures should come along with mesh, no need for separate upload.	Set the SketchUp export to include graphics. 3ds Max and Blender will not include graphics in their COLLADA export.
4	Use an "Unwrap UVW map" modifier on the mesh if you want to apply a graphic to a more complex form and arrange it so that the graphic fits nicely on the geometry.	Add a blank UV graphic to your UV map ("New" in the UV/Image editor).	Upload mesh model into virtual world.	
5	With your unwrapped UV template, add photo graphics or hand-painted graphics to the geometry.	Add your lights.		
6	Render to Texture with the lighting in the scene to capture the whole look.	Select your mesh and press Bake in the Properties/Render menu and save your baked image.		

(*Continued*)

TABLE 10.1 (CONTINUED)
Overview of Mesh Creation for Uploads: Comparing 3ds Max, Blender, and SketchUp

Procedural	3D Modeling Software Used			
Steps	3ds Max	Blender	SketchUp	Notes
7	Upload the final rendered graphic into virtual world.	Select your mesh and save as .dae (File/Export/COLLADA) using the Second Life Static operator preset for static mesh and the Second Life Rigged operator preset for rigged meshes.		Any changes or tweaks to the "baked" UV map should be done in Photoshop before this step.
8	Export the mesh model as an Autodesk COLLADA.dae file and upload it to the virtual environment.	Upload to your virtual world simulator.		
9	Other maps such as Specular and Normal Maps that have been generated with plugins or Photoshop can be uploaded and added to the mesh model through the same menu.			
10	Rezz the mesh inworld and add the rendered graphic/texture in the Build/Edit/Textures menu.			

Question: Will the use of procedural generators for mesh in your virtual environment really save you labor and time?

Answer: Models that require lots of optimization may not be worth the effort; try a test first.

10.2.4 Some Considerations regarding Meshes Derived from Digital Sculpting

Imagine that you can get your hands around some digital clay and make a form appear. This is digital sculpting in a nutshell. It allows you to manipulate the 3D data set just as you manipulate a ball of clay. Methods for the generation and refinement of 3D form in digital sculpting have developed out of two major categories: polygonal mesh sculpting and voxel-based sculpting. You will find polygonal-based sculpting available in most 3D modelers such as Blender, 3ds Max, and MODO. You will find voxel-based sculpting in programs such as 3D-Coat, Mudbox, and Geomagic Freeform. However, these two approaches to making 3D sculptures are not mutually exclusive these days; you will find voxel-based modelers that combine their workflows with polygonal modeling, and polygonal modelers that incorporate aspects of voxel modeling [4]. For example Blender offers "Sculpt Mode," which lets you start sculpting right in the main interface; and 3ds Max allows you to export a polygonal model into Mudbox and refine it with sculptural tools. ZBrush offers a polygonal modeling tool called the ZModeler brush (with ZBrush version 4R7), along with its proprietary "pixol" technology for creating volumetric digital sculpture [5]. 3D-Coat offers both voxel and polygonal sculpting within its sculpting toolset, and a proprietary "skinning" algorithm that creates a polygonal mesh from the voxel-based sculpts [6]. As these digital sculpture tools improve, the line between real sculpture and digital sculpture is beginning to blur. Digital sculpture can be digitally printed to make real sculpture and real sculpture can be scanned to make digital sculpture. In the next section we shall look at some of the ways you can do that with photogrammetry that combines digital photography and 3D digitizing programs. Once again, there are some important questions to ask before you leap into digital sculpting.

Question: Do you need to create very low polygon meshes from very organic high polygon models for your virtual environment?

Answer: If the answer is yes, then you need a digital sculpting program that links to a more traditional polygonal modeling program, such as ZBrush and Mudbox.

Question: Do you need to create relatively low polygon digital sculpture?

Answer: If the answer is yes, then 3D modeling programs that also have sculpting tools in them may be the solution to your digital sculpting needs, such as Blender and 3ds Max.

Question: Does the digital sculpture program integrate seamlessly with other 3D modeling programs you use?

Answer: Always check the import/export file formats carefully, because it is a huge time saver when files can move easily back and forth between your programs.

Question: If you are going to use unrelated programs to create meshes, what is a good file converter?

Answer: Useful converters to know about are MeshLab (http://meshlab.sourceforge.net/) and MilkShape 3D (http://www.milkshape3d.com/).

10.2.5 Some Considerations regarding Scanning and Mesh Creation for Virtual Environments

Until recently, the most accurate scanning and the digitization of real 3D objects and environments was accomplished with systems that use lidar-based scanners (lidar is a portmanteau combining *light* and *radar*) or contact scanners that collect the data set by touching point after point on the surface of the object. In the last few years, more powerful and accurate scanning that leverages photogrammetry, or the use of digital pictures to gather the data set for a 3D environment or object, has become available. With a good digital camera and open source software like Python Photogrammetry Tools (PPT), Mesh Lab, and Blender, archeological researchers are creating accurate 3D models of artifacts and historic sites [7]. For commercial users who want to try a cloud-based digitizing system, there is Autodesk Memento, available at http://memento.autodesk.com. Software such as D Sculptor 2 from D Vision Works (http://www.d-vw.com/index.htm), iModeller (http://www.imodeller.com/en/), and PhotoModeler (http://www.photomodeler.com) are also readily available. Our options for scanning and digitizing a real 3D object with a digital camera and a computer have been proliferating since 2012. These are tools we are familiar with and are readily accessible to most of us. Here are some things to keep in mind as you get ready to make a series of photographs for the creation of a 3D model from inanimate objects:

- Use a good digital camera with a fixed lens and a tripod or monopod.
- Matte surfaces work best, so do not use shiny or transparent objects in the real-life setup.
- Remember to consider the resolution of each image; the object you are photographing should be the majority of the image.
- Consider the lighting, and try to get it as even and shadow-free as possible.
- Think of the capturing process as a circular dance, with measured and equal steps that move only a few degrees around the object with each step.
- If you are making a model from many movable objects or a public site, try to capture your photos in one long session during a quiet period of time, so that you do not have to return to the location or reset the display and risk having two sets of photos that are slightly different.
- If you are just starting out with photogrammetry, then check out the Autodesk Memento resources website http://memento.autodesk.com/resources. It covers the process of taking photos for photogrammetry in great detail.

In Figure 10.3 are some screenshots showing a 3D object (a gargoyle sculpture) that was photographed, analyzed by Autodesk Memento, and viewed as a 3D model in 3ds Max. Next to that is an image of the digital model after it has been uploaded into a virtual environment as a COLLADA.dae file.

10.3 OVERVIEW OF RIGGED MESHES; MOVING AVATARS AND THEIR CLOTHING

As you probably know your avatar and some of the clothing it wears are made from polygonal meshes. If you toggle on the wireframe render mode by using the key command (CTRL+Shift+R), you can see how all the parts fit together. Your avatar is a special kind of mesh model that can move in response to looping animation files like a walk cycle or a sit pose. Like your human body, your avatar mesh is connected to an internal structure called a skeleton, and this skeletal structure is what controls the movement of the polygons in your avatar mesh. Creating these “rigged” meshes or models for virtual environments is challenging and rewarding. This kind of mesh modeling comes with its own concepts and related terminology, which you should have a general knowledge of before you attempt to make your first rigged mesh. As you can see in Figure 10.4, a basic virtual world avatar used in Second Life or OpenSim is made from three major mesh parts: the head, the upper body, and the lower body. The minor mesh parts that complete your avatar are the eyes, inside mouth surface, and your avatar’s basic hair mesh, all inside the skull. Inside of the major mesh body parts is a “skeleton” made from “collision bones,” which you can see with this process: (1) toggling into the wireframe mode, (2) activate the Develop menu tab (CTRL+ALT+Q), and (3) check on the Collision Skeleton under the Render Metadata listing. Develop/Render Metadata/Collision Skeleton is the menu path. As you may note,

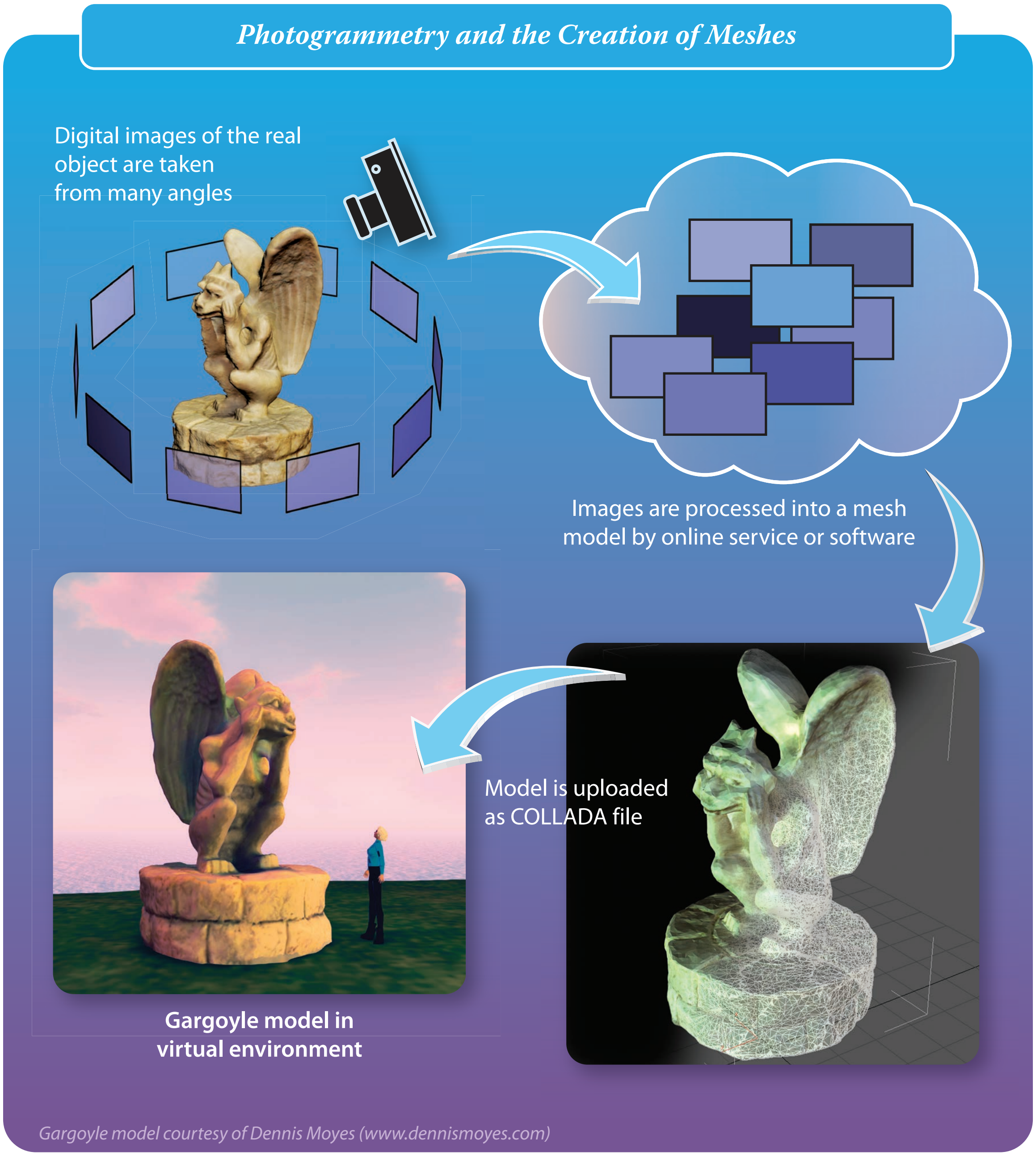

Photogrammetry and the Creation of Meshes
Digital images of the real object are taken from many angles
Images are processed into a mesh model by online service or software
Model is uploaded as COLLADA file
Gargoyle model in virtual environment
Gargoyle model courtesy of Dennis Moyes (www.dennismoyes.com)

FIGURE 10.3 Photogrammetry, 3D models, and virtual environments.

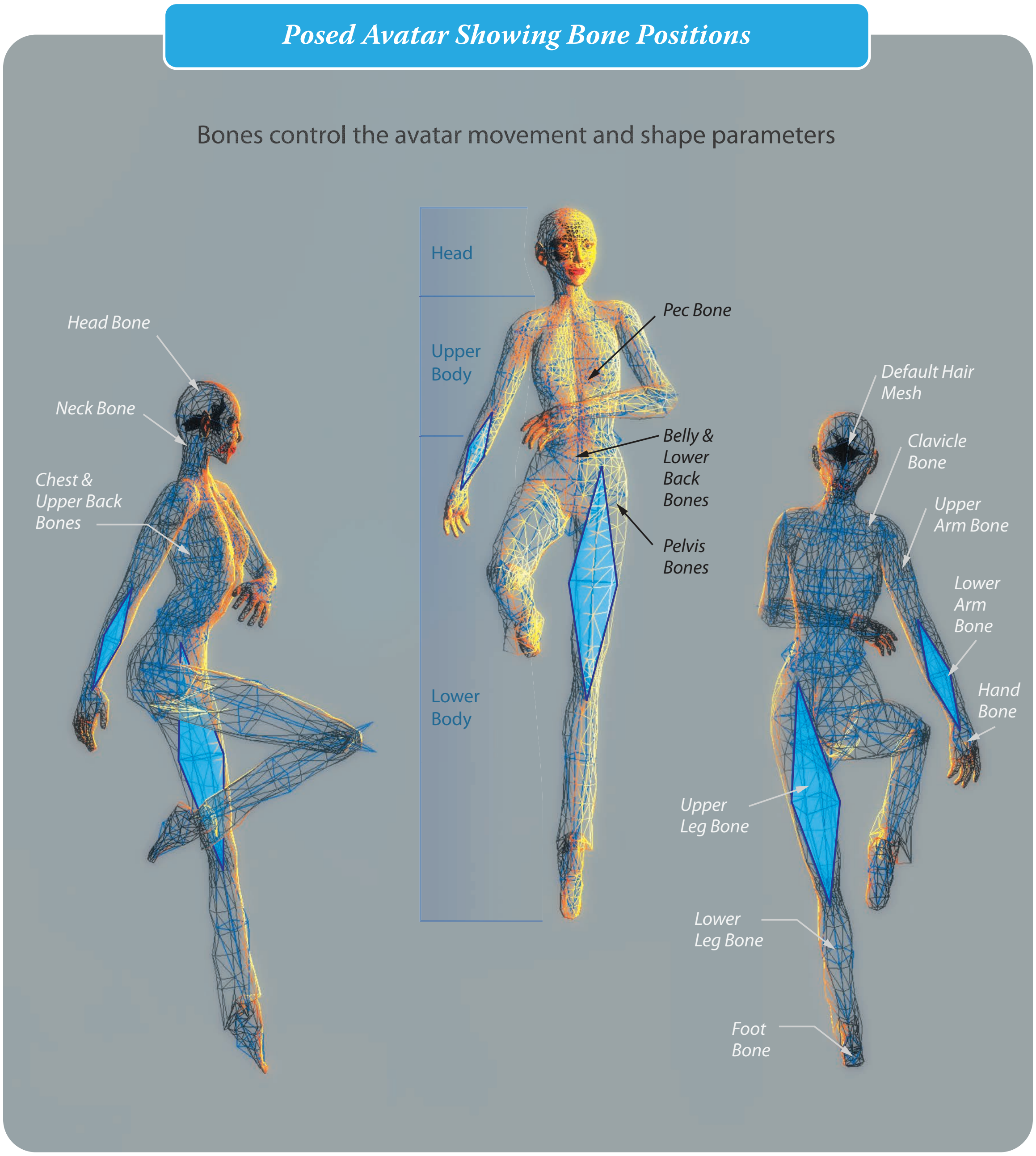


FIGURE 10.4 Rigged mesh and skeleton of a basic avatar, three views.

there are series of blue octahedrons (double 4-sided pyramids) aligned along your avatar's body, roughly in the place of where your real major bones are. There are also some additional parts that provide for body movement like a physics-based bounce and body proportion adjustments for the breasts, waist, and buttocks. To see how these bones affect your avatar's shape, go into the My Appearance menu and try adjusting your body height and width while you are looking at your avatar in wireframe mode. As you can see, the bones move when you adjust the body shape to new parameters. These bones are not just the jointed skeleton that moves in response to an animation cycle; they are also "handles" that allow you to move sections of your avatar's mesh and to sculpt a new shape for them. Mesh clothing can also be linked to your avatar's bones, to allow for a proper fit and movement of the fabric as your avatar moves [8]. Skeletal animation is the standard way to animate characters in real-time environments like gaming and virtual worlds. As you can see in Figure 10.4, the skeleton in your avatar is named and connected much like your own real one.

There is a hierarchy of influence in these bones. The central parts of the avatar's body, containing the pelvis, belly, and chest, are the major controllers of gross body movement; in other words, transform the position of one of these and the whole body will change its location. If you move a part of the body that is a minor part of the hierarchy, such as a hand or foot, then only the forearm or lower leg may move significantly. This kind of hierarchy is often described in "parent" and "child" terms, meaning that the parent is a major member of the hierarchy (like the pelvis), and the child is a minor member (a hand) and will respond to movement in the parental element. The skeleton is connected to the avatar mesh in a process called *skinning*. In this process, the vertices and faces of the avatar mesh are associated with one or more bones that will define the movement of the joint in that part of the avatar [9]. For an animator, this relationship is crucial, as they typically will move the avatar from pose to pose in order to create an animation loop. This animation process relies on a system called *inverse kinematics*, which automatically calculates the joint angles of the bones when the avatar is posed, while also preventing unnatural or impossible joint movement [10]. These joint angles and the transformations are recorded and become part of the animation data set that creates movement for the avatar. To understand a little bit more about skinning and its limitations with our current avatar animation, hold out one of your arms and perform a bicep curl by moving your hand toward your shoulder. Make the following observations as you perform this:

- Observation 1—The hand moves much farther in space than your forearm or upper arm does, and the forearm moves more than the upper arm does
- Observation 2—The muscle that is contracting in your upper arm, the biceps, is getting shorter and fatter as it brings your forearm and hand up near your shoulder

Observation 1 demonstrates that, like your real bones, the collision bones in your avatar's arm can influence the positions of the avatar mesh parts to greater and lesser extents. In a rigged animated character, this is referred to as *bone weight*, meaning the weight of influence a bone has on each part of the limb that contains it. If you use Blender, you can show this data as a heat map (see Figure 10.5) to clearly indicate the parts of the body that are rigged and how much they will be influenced by the bone movement being created by the animator. Note, this also relates to parts of the avatar that are influenced by the physics-affected bones in the breasts and buttocks that respond to movement.

Observation 2 shows us the shortcomings of our avatar animation in virtual worlds to date. When our avatar moves its body, we do not see changes in the position or shape of an underlying musculature system. That kind of animation is currently beyond the parameters of virtual world avatars. It may become possible in the near future when the client viewers can display morphing body shapes for avatars in response to a given animation they are performing.

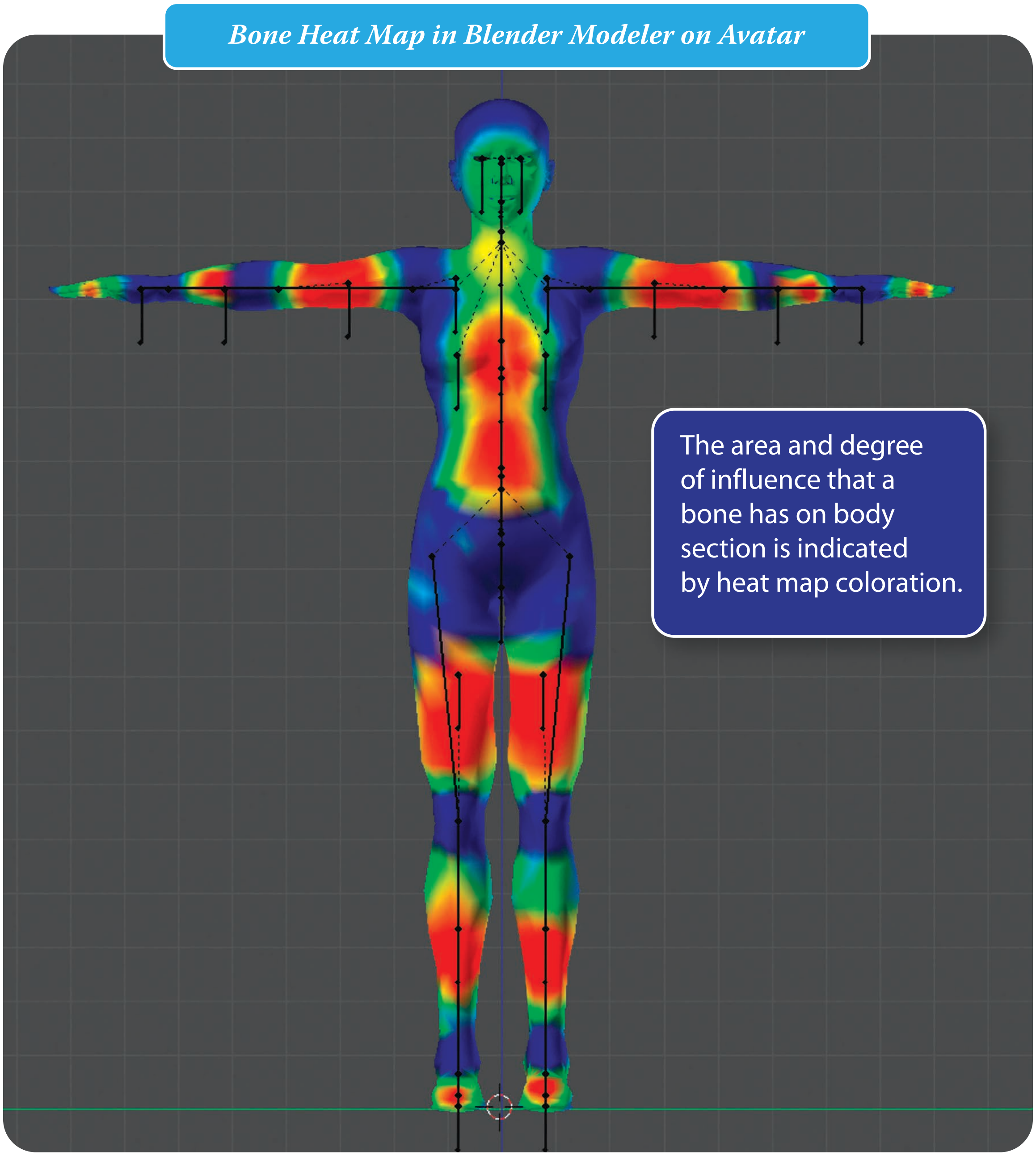

Bone Heat Map in Blender Modeler on Avatar
The area and degree of influence that a bone has on body section is indicated by heat map coloration.

FIGURE 10.5 Standard avatar in Blender 3D modeler showing a bone heat map.

10.3.1 Overview of Rigged and Fitted Mesh Clothing

Avatar clothing design has developed considerably since the early days of virtual worlds. As you can see from Figure 10.6 the avatar has moved from wearing graphic-/texture-based garments with prim or sculpty attachments (on the left) to completely rigged mesh outfits (on the right) that move with the avatar.

There are many content creators in Second Life who specialize in clothing for avatars, and their clothing is available in the Second Life Marketplace. However, if you are developing your own grid and the content for it in OpenSim, you may end up having to make your own line of clothing for your avatars. Fortunately this process is well documented both in the Second Life Wiki, http://wiki.secondlife.com/wiki/Clothing_Tutorials, and on the YouTube.com website under the keyword search for "second life + clothing + rigged mesh." You will find additional information including tips for getting the best results on the Fitted Mesh wiki page here: http://wiki.secondlife.com/wiki/Mesh/Rigging_Fitted_Mesh.

10.3.2 Tools and Plugins for Creating Rigged Meshes

Some specialty tools have been devised to simplify the process of creating avatar and avatar clothing meshes specifically for the Second Life and OpenSim avatars. For Blender there is Avastar, available for download at http://blog.machinimatrix.org/. Another plugin that is widely available is SLAV from Wiz Daxter (http://wiz-bg.blogspot.com/p/slav-3ds-max.html). This is offered as a free demo version, so you can test it before buying.

10.4 COMPARISON OF 3D MESH-MAKING PROGRAMS; SOME THOUGHTS ON CHOOSING ONE

There is no getting around it: If you want to create custom meshes for your own usage, you will need to learn how to use one or more of these programs, or hire someone who does. In Table 10.2 is a comparison of the most fully featured 3D programs available for content creators in virtual worlds. You will probably note that the price for this software ranges from nothing (freeware like Blender) to costly (professional software like 3ds Max), and there are some in the middle of this price range. The high prices can be ameliorated with a school ID if you are a faculty or student member and need the program for classes.

However you decide to proceed, take the time to read all the specifications for the software before you decide to purchase it, and try the demo versions, so you are assured of getting the most useful package for your money.

10.5 PROJECT: BUILDING A GAME-BASED ENVIRONMENT WITH COMPONENT MESHES

Now it is time for building the major architectural elements for your game-based virtual environment. Tim Widger and I labored for months to create these elements and we hope you find them useful for developing all sorts of games and promoting emergent play behavior. For the purposes of the project steps listed in the following, we will specify exact coordinates, so that these buildings will line up correctly on the terrain that was created for the project in Chapter 9, Section 9.6, "Project: Creating a Four-Region, Game-Based Terrain." Should you decide to freestyle design your game-based environment, the village and quay should be placed on a slope, but the other buildings can be positioned on any flat place in the terrain you would like to use. See the following instructions for linking the castle and village components into one piece with a central alignment cube. All of the building components are available for download from http://www.anncudworthprojects.com/ in the Extending VWD Downloads section. Each one of these building components (created as COLLADA files, e.g., BLDG_Castle_1.dae) has been made as a multipart linked object with an

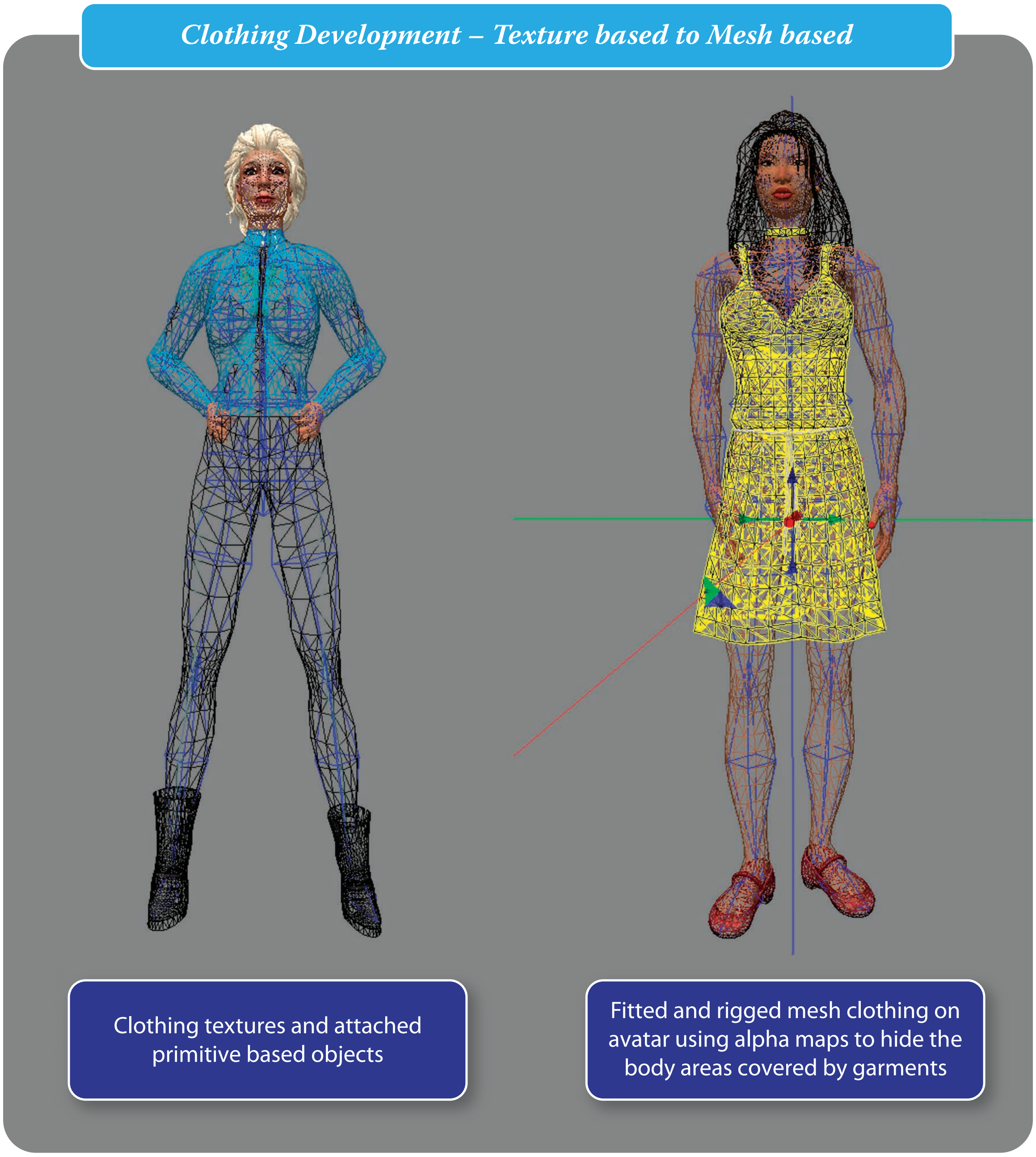

Clothing Development – Texture based to Mesh based
Clothing textures and attached primitive based objects
Fitted and rigged mesh clothing on avatar using alpha maps to hide the body areas covered by garments

FIGURE 10.6 Images of avatar clothing development.

TABLE 10.2
Comparison of Five Mesh-Making Programs, by Price and Workflow

Software Name	Price for Standard User	Import/Export Options	Recognizes Bones	Recognizes UV Mapping	Types of Workflow
3ds Max (version 2009 and up)	Monthly subscription $185 ($3675 for perpetual license)	Via .FBX plugin	Yes	Yes	Character, clothing, and mesh creation
Maya (version 2009 and up)	Monthly subscription $185 ($3675 for perpetual license)	Via .FBX plugin	Yes	Yes	Character, clothing, and mesh creation
Blender (version 2.49 and up)	Free	COLLADA.dae	Yes	Yes	Character, clothing, and mesh creation
Cinema 4D (version R11 and up)	$3695 to buy	COLLADA.dae	Yes	Yes	Character, clothing, and mesh creation
DAZ Studio (version 2.0 and up)	Free	COLLADA.dae	Yes	Yes	Creation of background images with DAZ content or use software to develop imported avatar meshes

alignment cube (key prim) set up at its center. Note: Sometimes during the upload process, a multipart object will reorganize and the alignment cube will no longer be the key prim. If you select a newly uploaded building component and find that to be the case, simply use the Build/Edit linked menu to unlink the alignment cube, select the remaining parts of the component, and then the cube, so it becomes the last selected element, and then link them back together. That should make the alignment cube the key prim (it will show a yellow highlight) of your linked object and give you the center position on the entire component so you can locate it correctly. Also as a general note about graphics, during the upload/import of these COLLADA (.dae) files, the alpha transparencies on the graphic/textures may come with the transparencies set to 0%, creating opaque windows. To make the windows semitransparent again, simply go into Build menu, check the Edit linked box and select the zfaces option. Click on the window faces and change the transparency to the level you desire, as shown in the inset of Figure 10.11. You will also want to do that for the "hinge" vertex face that extends out from the "practical" doors, found in the castle, barbican, and two of the village houses. As you can see in Figure 10.16, that triangular face has been selected and made completely transparent.

10.5.1 Loading in the Castle

You will be installing the castle on the northeast region of the four-region, game-based sim. The castle has 14 separate linked components and the alignment cube (key prim) of their linkset should be located at these coordinates, in a standard 256 meter square region: X = 192.91666 meters, Y = 148.91270 meters, Z = 41.26767 meters. Every linked component section of the castle should have a rotation of X = 0, Y = 0, and Z = 317 degrees. In Figure 10.7 is an exploded view of the castle and its 14 component parts, so you can see an overview of the structure that will be assembled.

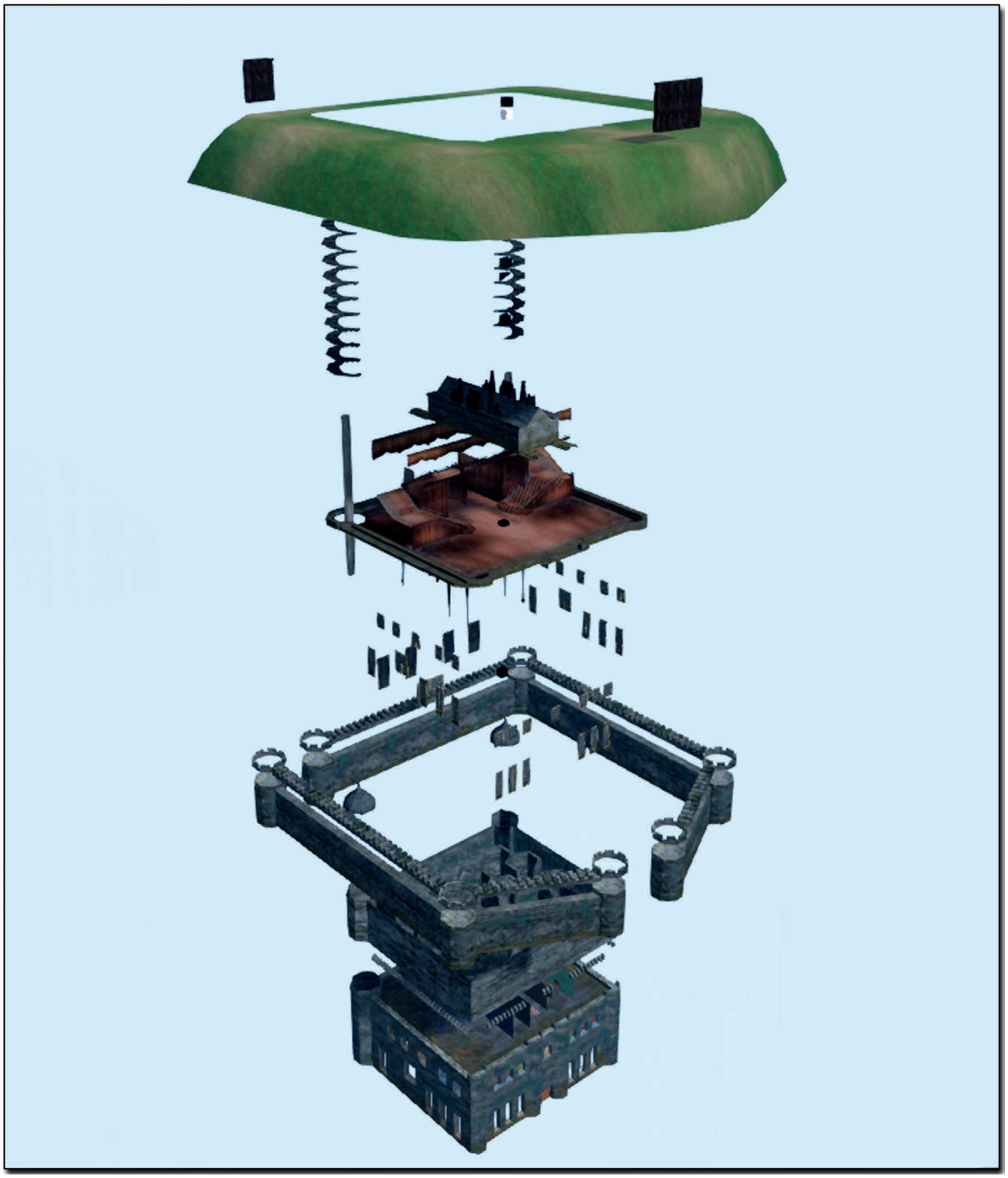


FIGURE 10.7 Exploded view of castle components.

It is not necessary that you assemble the castle in any specific order, as long as the alignment cubes (key prims) for each component are at the same coordinates and rotation. However, for the purpose of a clear explanation, I have chosen an order of assembly that allows the least obstructed view of the castle elements as they are assembled in place on the terrain that you created in Chapter 9, Section 9.6.

Step 1: Dungeon and Tower Roofs (BLDG_Castle_3)

In Figure 10.8 is a screen shot of the dungeon section of the castle being installed in its terrain "excavation" foundation. This folder containing both the COLLADA 3D model file (.dae) and its related graphic/textures is called BLDG_Castle_3. Using the upload menu in the client viewer make sure to include the graphics and set the physics file to be the same file as the model file. See Chapter 2, Section 2.9 and Chapter 7, Section 7.5, for more detailed information on using the upload model menu in the Firestorm viewer.

Step 2: Ground, Internal Stairs, and Roof Shed (BLDG_Castle_13)

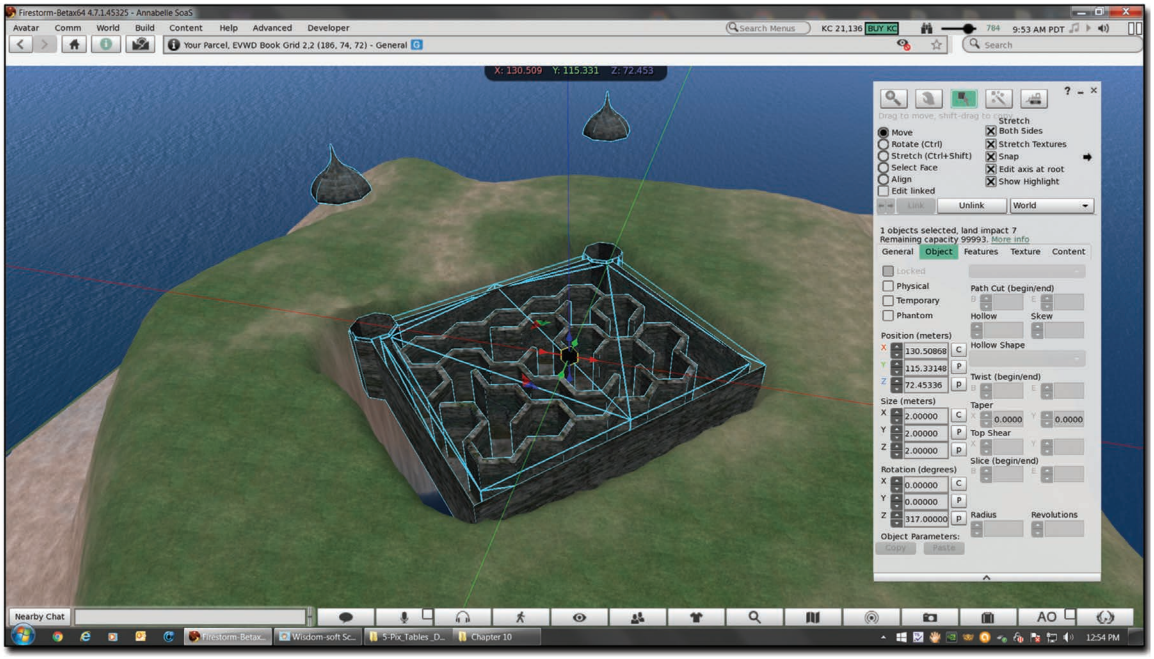
Next, we will add in the ground cover and the tower roofs. Upload the files for BLDG_Castle_13, and place them at the same coordinates. You can see the result in Figure 10.9.

Step 3: Interior Floors and Staircases (BLDG_Castle_8, 9, and 10)

Now that you have the hang of it, we will move a bit faster. In the next image, Figure 10.10, you can see the results of uploading and installing three castle components (8, 9, and 10) that comprise the interior floors and staircases in the castle. Again, install them at the same coordinates, and if they seem misaligned, then check to make sure the key prim is the alignment cube.

Step 4: Internal Walls and Windows (BLDG_Castle_1, 2, 6, and 7)

Four components (1, 2, 6, and 7) make up the walls and windows for the castle, as you can see in top image of Figure 10.11. Once you have gotten them in place, open the Edit/Build menu, check

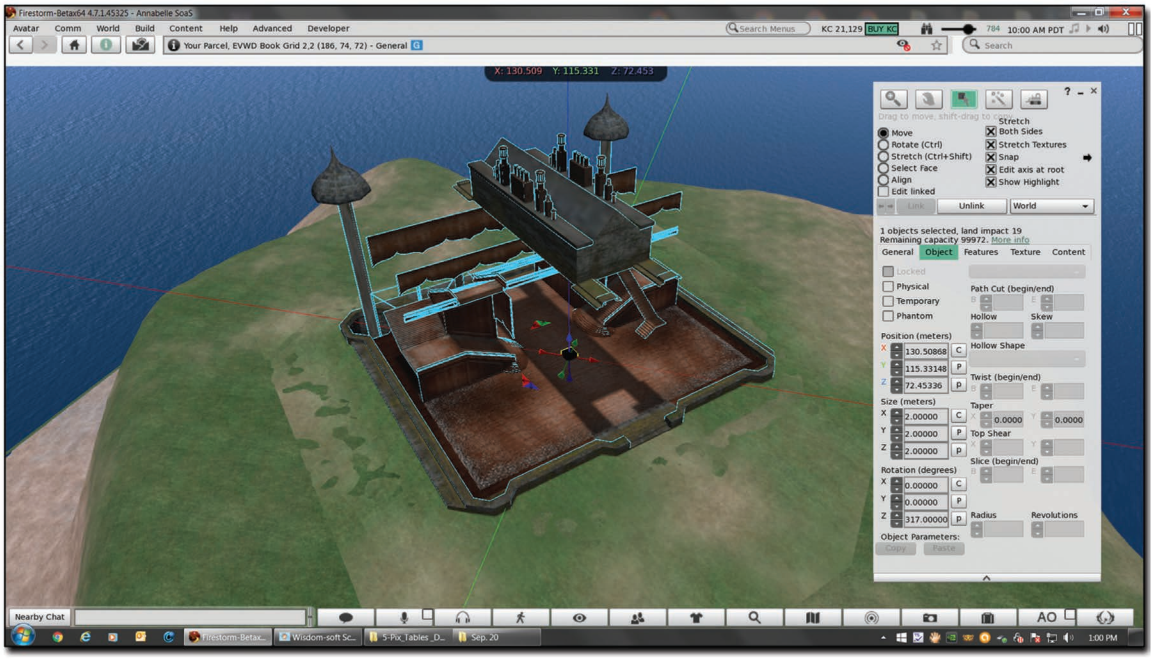

Firestorm-Betax64 4.7.1.45325 - Annabelle SoaS
Avatar Comm World Build Content Help Advanced Developer
Search Menus
KC 21,136 BUY KC
784 9:53 AM PDT
Your Parcel, EVWD Book Grid 2,2 (186, 74, 72) - General G
Search
X: 130.509 Y: 115.331 Z: 72.453
Drag to move, shift-drag to copy
Link Unlink World
General Object Features Texture Content
Nearby Chat
AO
12:54 PM

Move [✓]; Rotate (Ctrl); Stretch (Ctrl+Shift); Select Face; Align; Edit linked
Stretch: Both Sides [✓]; Stretch Textures [✓]; Snap [✓]; Edit axis at root [✓]; Show Highlight [✓]

1 objects selected, land impact 7
Remaining capacity 99993. More info

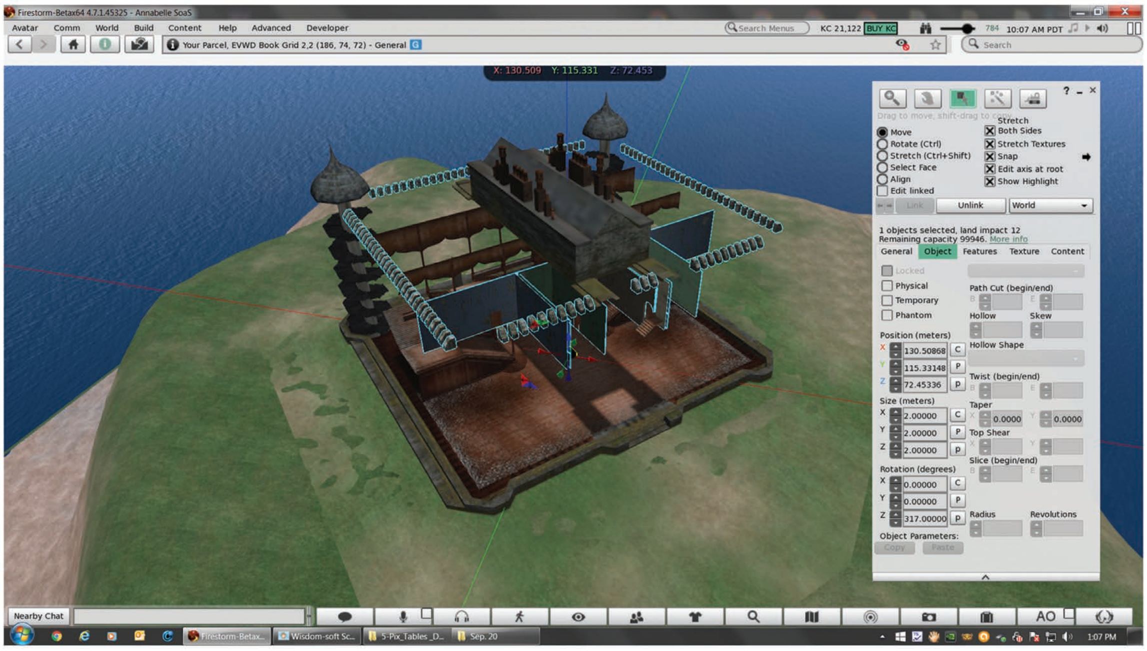
Locked; Physical; Temporary; Phantom
Position (meters): X 130.50868, Y 115.33148, Z 72.45336
Size (meters): X 2.00000, Y 2.00000, Z 2.00000
Rotation (degrees): X 0.00000, Y 0.00000, Z 317.00000
Path Cut (begin/end); Hollow; Skew; Hollow Shape; Twist (begin/end)
Taper: X 0.0000, Y 0.0000
Top Shear; Slice (begin/end); Radius; Revolutions
Object Parameters: Copy Paste

FIGURE 10.8 Castle assembly: dungeon, step 1.


Firestorm-Betax64 4.7.1.45325 - Annabelle SoaS
Avatar Comm World Build Content Help Advanced Developer
Your Parcel, EVWD Book Grid 2,2 (186, 74, 72) - General
X: 130.509 Y: 115.331 Z: 72.453
General Object Features Texture Content
Nearby Chat

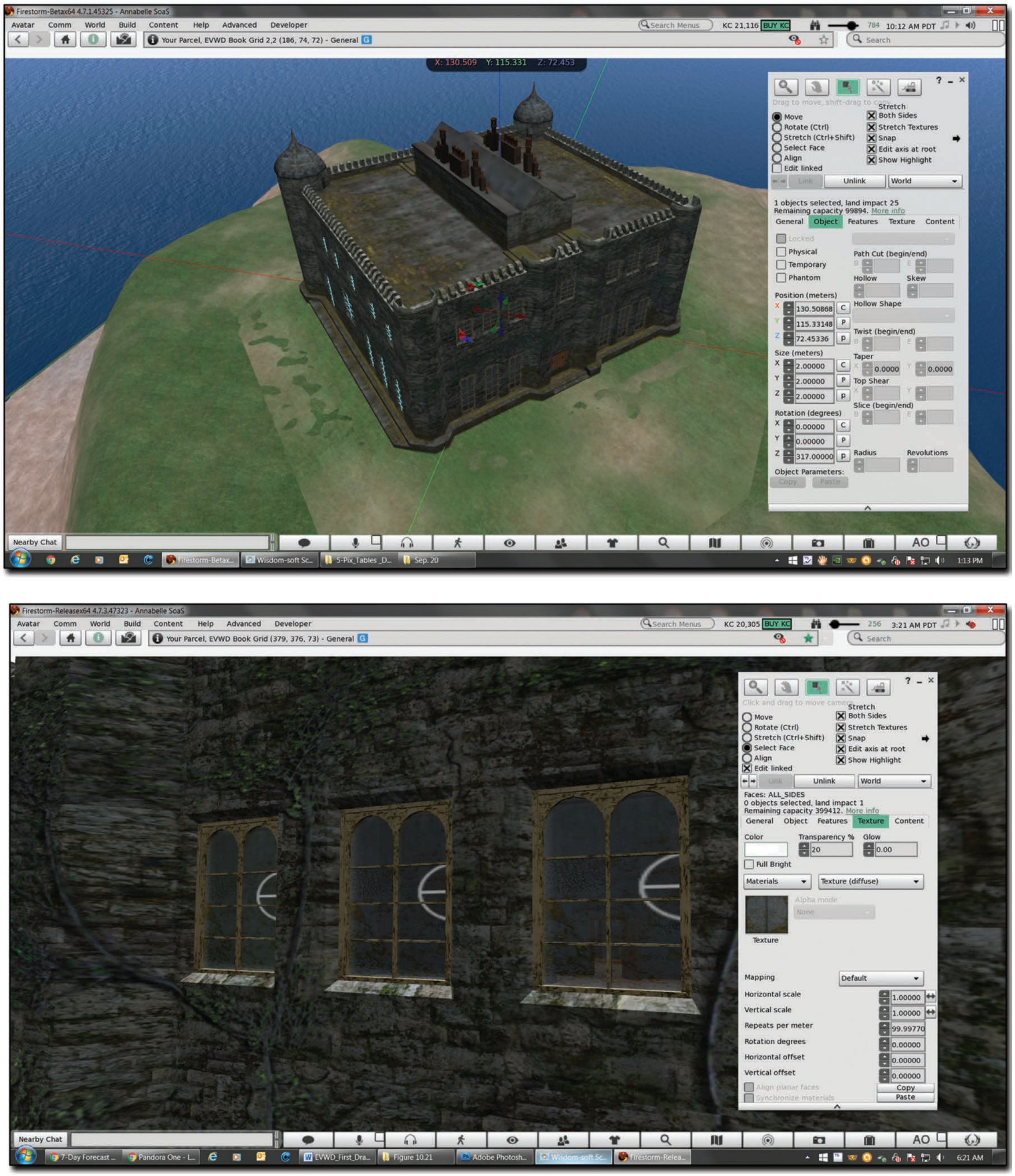
Move [✓]
Stretch Both Sides [✓]
Stretch Textures [✓]
Snap [✓]
Edit axis at root [✓]
Show Highlight [✓]
World
1 objects selected, land impact 19
Remaining capacity 99972. More info
Position (meters) X 130.50868 Y 115.33148 Z 72.45336
Size (meters) X 2.00000 Y 2.00000 Z 2.00000
Rotation (degrees) X 0.00000 Y 0.00000 Z 317.00000
Taper X 0.0000 Y 0.0000

FIGURE 10.9 Castle assembly: ground, internal stairs, and roof shed, step 2.


Firestorm-Betax64 4.7.1.45325 - Annabelle SoaS
Avatar Comm World Build Content Help Advanced Developer
Your Parcel, EVWD Book Grid 2,2 (186, 74, 72) - General
X: 130.509 Y: 115.331 Z: 72.453
General Object Features Texture Content
Nearby Chat

Move [✓]
Stretch Both Sides [✓]
Stretch Textures [✓]
Snap [✓]
Edit axis at root [✓]
Show Highlight [✓]
World
1 objects selected, land impact 12
Remaining capacity 99946. More info
Position (meters) X 130.50868 Y 115.33148 Z 72.45336
Size (meters) X 2.00000 Y 2.00000 Z 2.00000
Rotation (degrees) X 0.00000 Y 0.00000 Z 317.00000
Taper X 0.0000 Y 0.0000

FIGURE 10.10 Castle assembly: interior walls, staircases, and crenellation, step 3.


Firestorm-Betax64 4.7.1.45325 - Annabelle SoaS
Avatar Comm World Build Content Help Advanced Developer
Your Parcel, EVWD Book Grid 2,2 (186, 74, 72) - General
X: 130.509 Y: 115.331 Z: 72.453
Nearby Chat
Firestorm-Releasex64 4.7.3.47323 - Annabelle SoaS
Your Parcel, EVWD Book Grid (379, 376, 73) - General
Nearby Chat

Drag to move, shift-drag to copy
● Move ○ Rotate (Ctrl) ○ Stretch (Ctrl+Shift) ○ Select Face ○ Align ☐ Edit linked
Stretch: ☒ Both Sides ☒ Stretch Textures ☒ Snap ☒ Edit axis at root ☒ Show Highlight
Link Unlink World
1 objects selected, land impact 25
Remaining capacity 99894. More info
General [✓ Object] Features Texture Content
☐ Locked ☐ Physical ☐ Temporary ☐ Phantom
Position (meters) X 130.50868 Y 115.33148 Z 72.45336
Size (meters) X 2.00000 Y 2.00000 Z 2.00000
Rotation (degrees) X 0.00000 Y 0.00000 Z 317.00000
Path Cut (begin/end) Hollow Skew Hollow Shape Twist (begin/end) Taper X 0.0000 Y 0.0000 Top Shear Slice (begin/end) Radius Revolutions
Object Parameters: Copy Paste

Click and drag to move camera
○ Move ○ Rotate (Ctrl) ○ Stretch (Ctrl+Shift) ● Select Face ○ Align ☒ Edit linked
Stretch: ☒ Both Sides ☒ Stretch Textures ☒ Snap ☒ Edit axis at root ☒ Show Highlight
Link Unlink World
Faces: ALL_SIDES
0 objects selected, land impact 1
Remaining capacity 399412. More info
General Object Features [✓ Texture] Content
Color Transparency % 20 Glow 0.00
☐ Full Bright
Materials Texture (diffuse) Alpha mode None
Texture
Mapping Default
Horizontal scale 1.00000
Vertical scale 1.00000
Repeats per meter 99.99770
Rotation degrees 0.00000
Horizontal offset 0.00000
Vertical offset 0.00000
☐ Align planar faces ☐ Synchronize materials
Copy Paste

FIGURE 10.11 Castle assembly: windows and setting the transparency, step 4.

the Edit linked box, and go into the Select Face mode. Then set each cluster of window faces to the transparency you desire under the Texture tab/Transparency setting, as shown in the bottom image of Figure 10.11.

Step 5: Props and Their Physics Shapes (BLDG_Castle_11 and 12)

The next components are the props and their related physics shapes. These props are optional; they are simply décor inside of the castle rooms. As you can see in Figure 10.12, they contain furniture elements and such. There is a special physics shape that you should attach on the upload for BLDG_Castle_12, because these props can be used as seating elements.

Step 6: Barbican (the Outside Wall) and Crenellation (the "Teeth" on Top of Wall) (BLDG_Castle_4 and 5)

These two components make up the outside walls of the castle environs. As you can see in Figure 10.13, they combine to make up a crenellated fortification wall around the castle grounds.

Step 7: Castle Doors and Finishing Up (BLDG_Castle_14)

This is the last component for the castle assembly. As shown in the top image of Figure 10.14, this adds the doors for the walls surrounding the castle.

The bottom image of Figure 10.14 illustrates the next step. Once all of the castle components have been placed, and all of the alignment cubes are at the same coordinates, you have two options: (1) unlinking and deleting all of the alignment cubes (Build menu/Edit linked) in each of the 14 components, and then linking all of the castle components together, letting the key prim be chosen by the system from one of the remaining elements; (2) unlinking and deleting all but one of the alignment cubes, and then making that the key prim of the entire castle in one big linkset. Option 2 will maintain a central point for you should you need to move

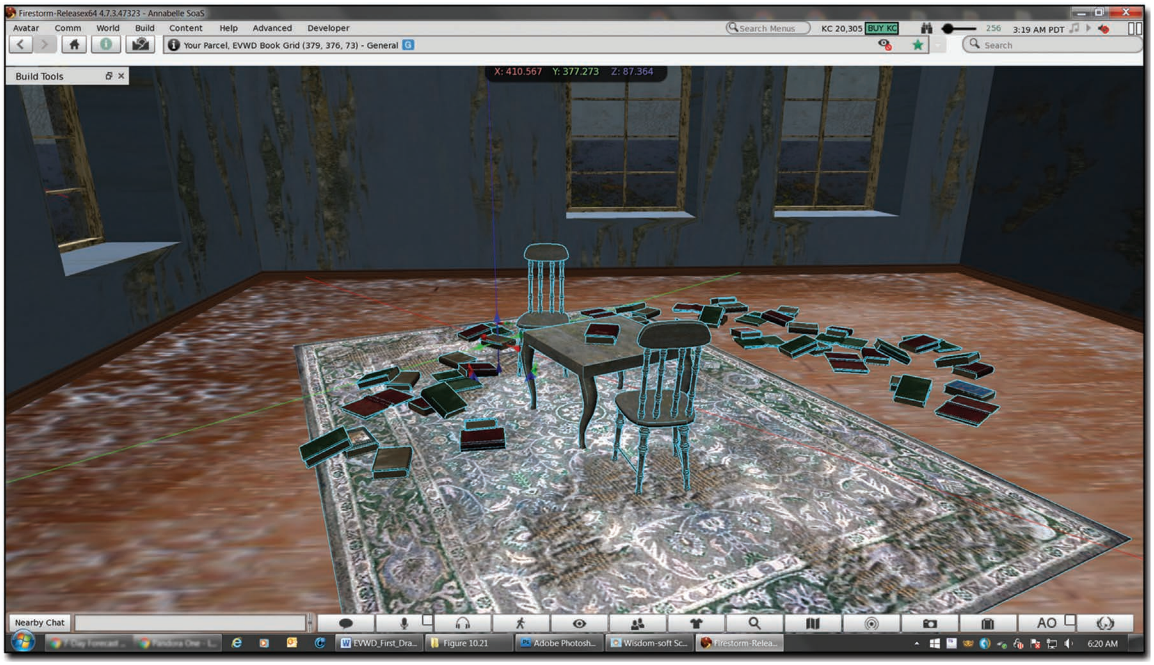
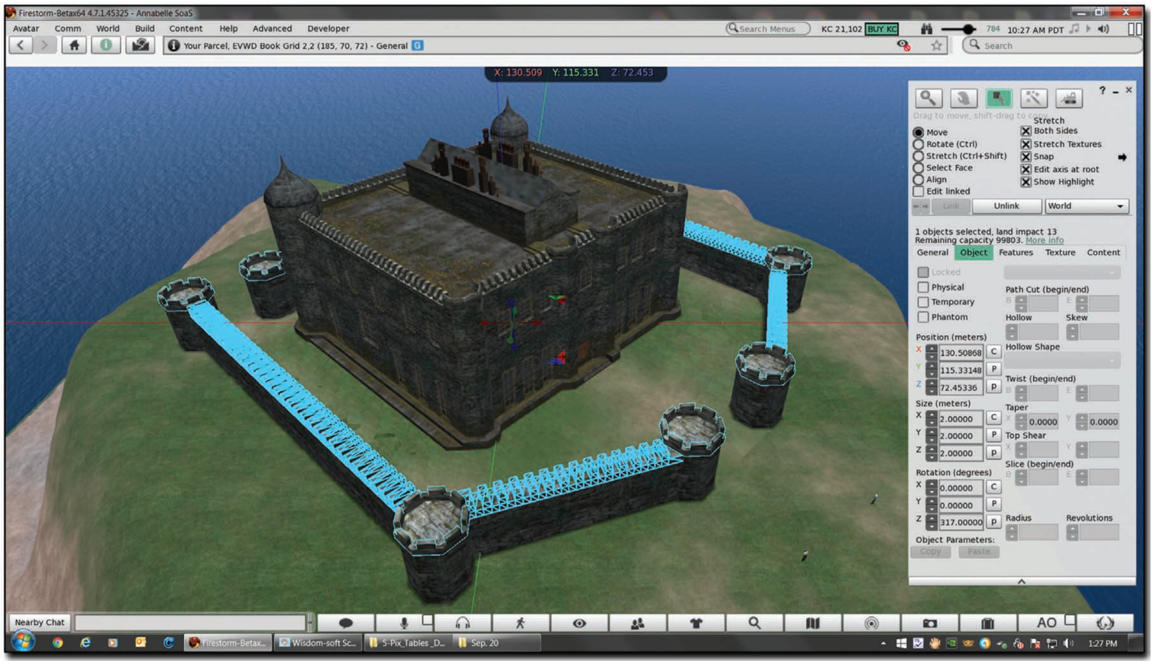

Firestorm-Releasex64 4.7.3.47323 - Annabelle SoaS
Avatar Comm World Build Content Help Advanced Developer
Search Menus
KC 20,305 BUY KC
3:19 AM PDT
Your Parcel, EVWD Book Grid (379, 376, 73) - General
Search
Build Tools
X: 410.567 Y: 377.273 Z: 87.364
Nearby Chat
AO
6:20 AM

FIGURE 10.12 Castle assembly: props, step 5.


Firestorm-Betax64 4.7.1.45325 - Annabelle SoaS
Avatar Comm World Build Content Help Advanced Developer
Your Parcel, EVWD Book Grid 2,2 (185, 70, 72) - General
X: 130.509 Y: 115.331 Z: 72.453
Nearby Chat
10:27 AM PDT
1:27 PM

Drag to move, shift-drag to copy
● Move
○ Rotate (Ctrl)
○ Stretch (Ctrl+Shift)
○ Select Face
○ Align
☐ Edit linked

Stretch
☒ Both Sides
☒ Stretch Textures
☒ Snap
☒ Edit axis at root
☒ Show Highlight

Link | Unlink | World

1 objects selected, land impact 13
Remaining capacity 99803. More info

General | Object | Features | Texture | Content

☐ Locked
☐ Physical
☐ Temporary
☐ Phantom

Position (meters): X 130.50868, Y 115.33148, Z 72.45336
Size (meters): X 2.00000, Y 2.00000, Z 2.00000
Rotation (degrees): X 0.00000, Y 0.00000, Z 317.00000
Taper: X 0.0000, Y 0.0000

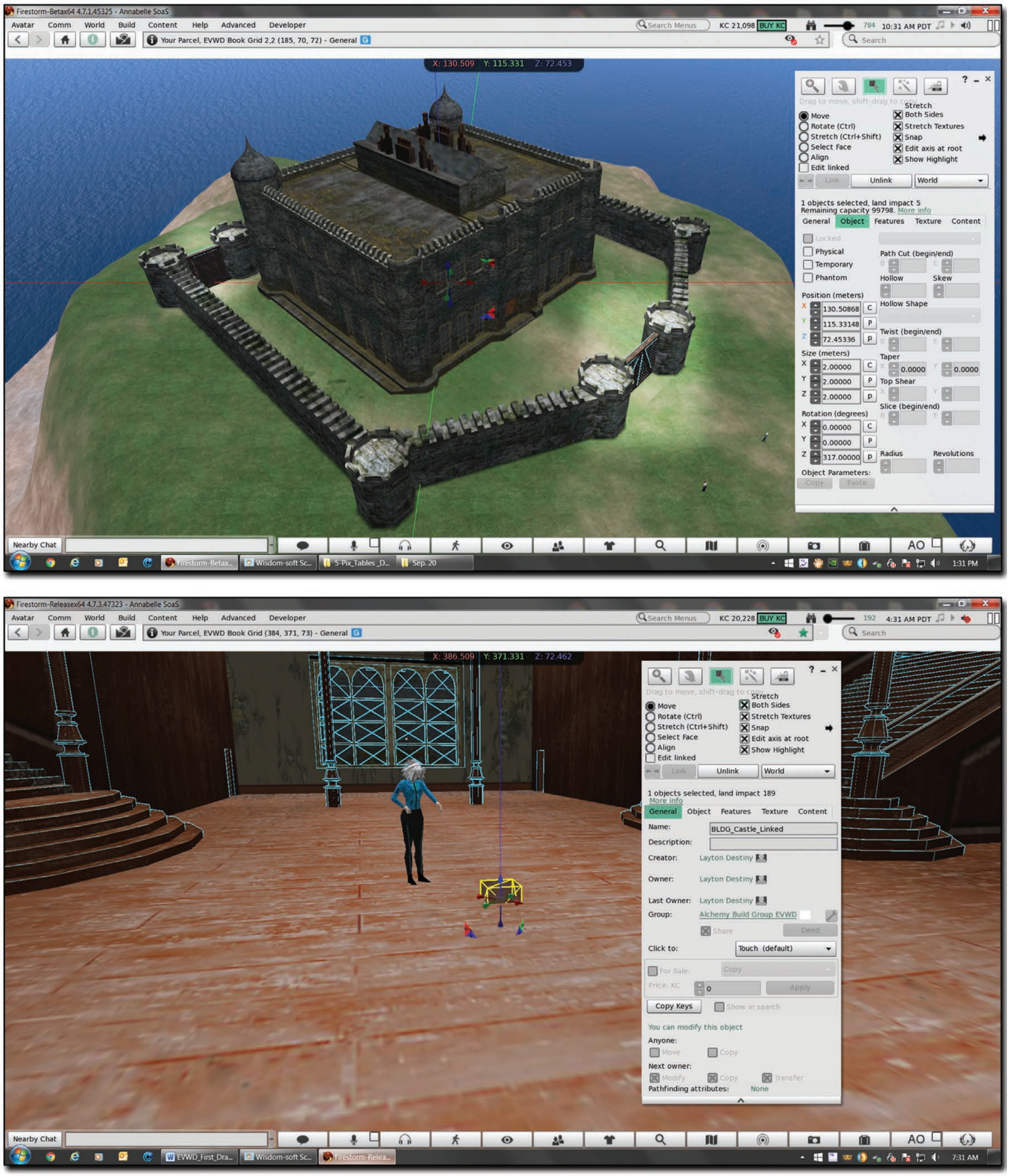
FIGURE 10.13 Castle assembly: barbican, step 6.

the castle somewhere else on the region. Since the remaining alignment cube is large, it is recommended that you scale it down, move it up near the first floor ceiling so it will not interfere with any movement inside the castle. You will also want to make its graphic/texture completely transparent so it disappears.

10.5.2 Loading in the Quay and Village

Now let us do the installation of the nine components that make up the village portion of the four-region game-based sim. This will be installed on the southeast region and the village coordinates are X = 192.91666 meters, Y = 48.91270 meters, and Z = 41.26767 meters. The rotation is X = 0, Y = 0, and Z = 258.10001 degrees. Every alignment cube, again make sure it is the key prim in the linkset, should be put at these coordinates and rotation. When you are uploading the 3D COLLADA files for this section, most of the model files are also to be used as the physics files, except for the village steps, piers, and props. Do not forget to include the textures in the upload menu settings.

Step 1: Quay (BLDG_Village_1)
As you can see in Figure 10.15, this component, (BLDG_Village_1), creates the massive stone quay that supports the village houses on the hillside and the street between them.
Step 2: Houses (BLDG_Village_2, 3, and 4)
The next three components, shown in the top two images in Figure 10.16, are known as BLDG_Village_2, 3, and 4. These meshes will complete the house elements on each side of the main street. The interiors of two of them are completed: The Mousehole Inn on the lower, southeast


Firestorm-Betax64 4.7.1.45325 - Annabelle SoaS
Avatar Comm World Build Content Help Advanced Developer
Your Parcel, EVWD Book Grid 2,2 (185, 70, 72) - General
KC 21,098 BUY KC
10:31 AM PDT
X: 130.509 Y: 115.331 Z: 72.453
Move
Rotate (Ctrl)
Stretch (Ctrl+Shift)
Select Face
Align
Edit linked
Stretch
Both Sides
Stretch Textures
Snap
Edit axis at root
Show Highlight
Link Unlink World
General Object Features Texture Content
Nearby Chat
Firestorm-Releasex64 4.7.3.47323 - Annabelle SoaS
Your Parcel, EVWD Book Grid (384, 371, 73) - General
KC 20,228 BUY KC
4:31 AM PDT
X: 386.509 Y: 371.331 Z: 72.462
Copy Keys
Nearby Chat

1 objects selected, land impact 5
Remaining capacity 99798. More info

Locked
Physical
Temporary
Phantom

Position (meters)
X 130.50868
Y 115.33148
Z 72.45336

Size (meters)
X 2.00000
Y 2.00000
Z 2.00000

Rotation (degrees)
X 0.00000
Y 0.00000
Z 317.00000

Object Parameters:

Path Cut (begin/end)
Hollow Skew
Hollow Shape
Twist (begin/end)
Taper X 0.0000 Y 0.0000
Top Shear
Slice (begin/end)
Radius Revolutions

1 objects selected, land impact 189
More info

Name: BLDG_Castle_Linked
Description:
Creator: Layton Destiny
Owner: Layton Destiny
Last Owner: Layton Destiny
Group: Alchemy Build Group EVWD
Share Deed
Click to: Touch (default)
For Sale: Copy
Price: KC 0 Apply
Show in search
You can modify this object
Anyone: Move Copy
Next owner: Modify Copy Transfer
Pathfinding attributes: None

FIGURE 10.14 Castle assembly: castle doors and finishing up, step 7.

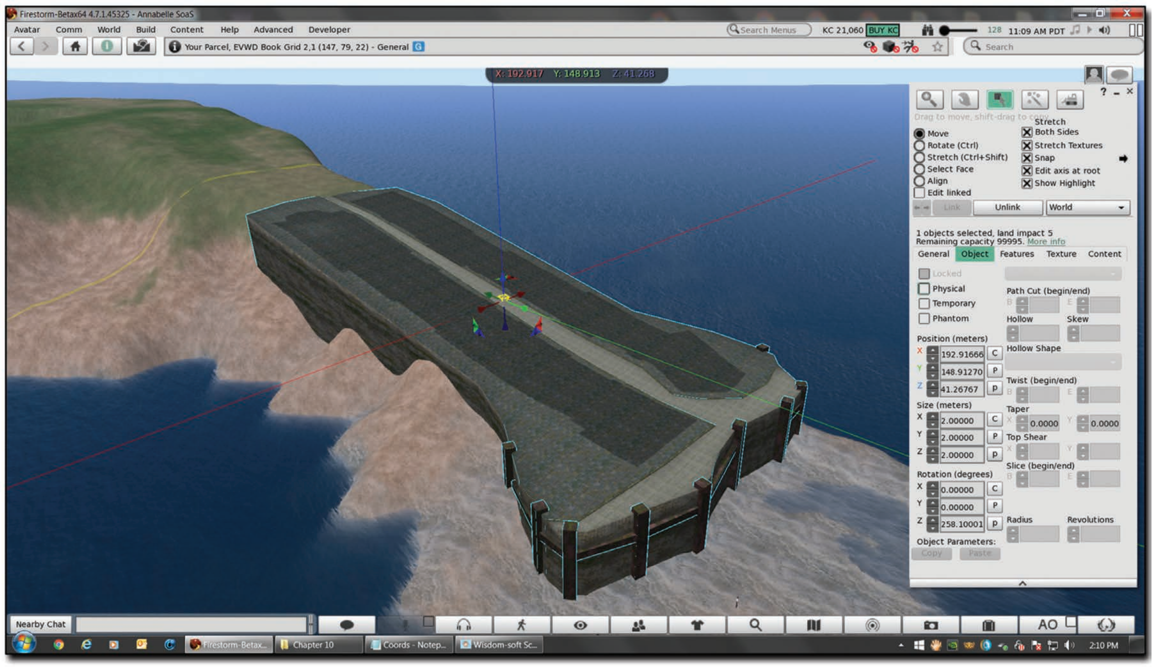


FIGURE 10.15 Quay assembly, step 1.

corner of the village; and one of the houses on the upper northwest corner, which we can call the Mayor's House. Once you have gotten them in place, open the Edit/Build menu, check the Edit linked box and go into the Select Face mode. Then set each cluster of window faces to the transparency you desire under the Texture tab/Transparency setting, just as you did for the castle windows on the hill. You will also want to do that for the "hinge" vertex face that extends from the rotatable doors, and make that face completely transparent, as you can see in bottom image of Figure 10.16.

Step 3: Exterior Quay and House Steps (BLDG_Village_5 and 6)

The next two components will add in the exterior stairs on the houses, as you can see in Figure 10.17. Make sure to include the separate physics file for these elements, so they will function properly with the avatars.

Step 4: Piers and Chains on Walkways (BLDG_Village_7 and 8)

The next set of components will install the piers and their connecting chains all around the village walkways. BLDG_Village_7 is the piers component and has its own special physics file. BLDG_Village_8 has 16 separate COLLADA files to complete the chains that surround the village, and may take some time to upload. Installing this part of the village is optional; leave it off if you want to keep the polygon count lower. In Figure 10.18, you can see what it looks like when all the components for the piers and chains are in place.

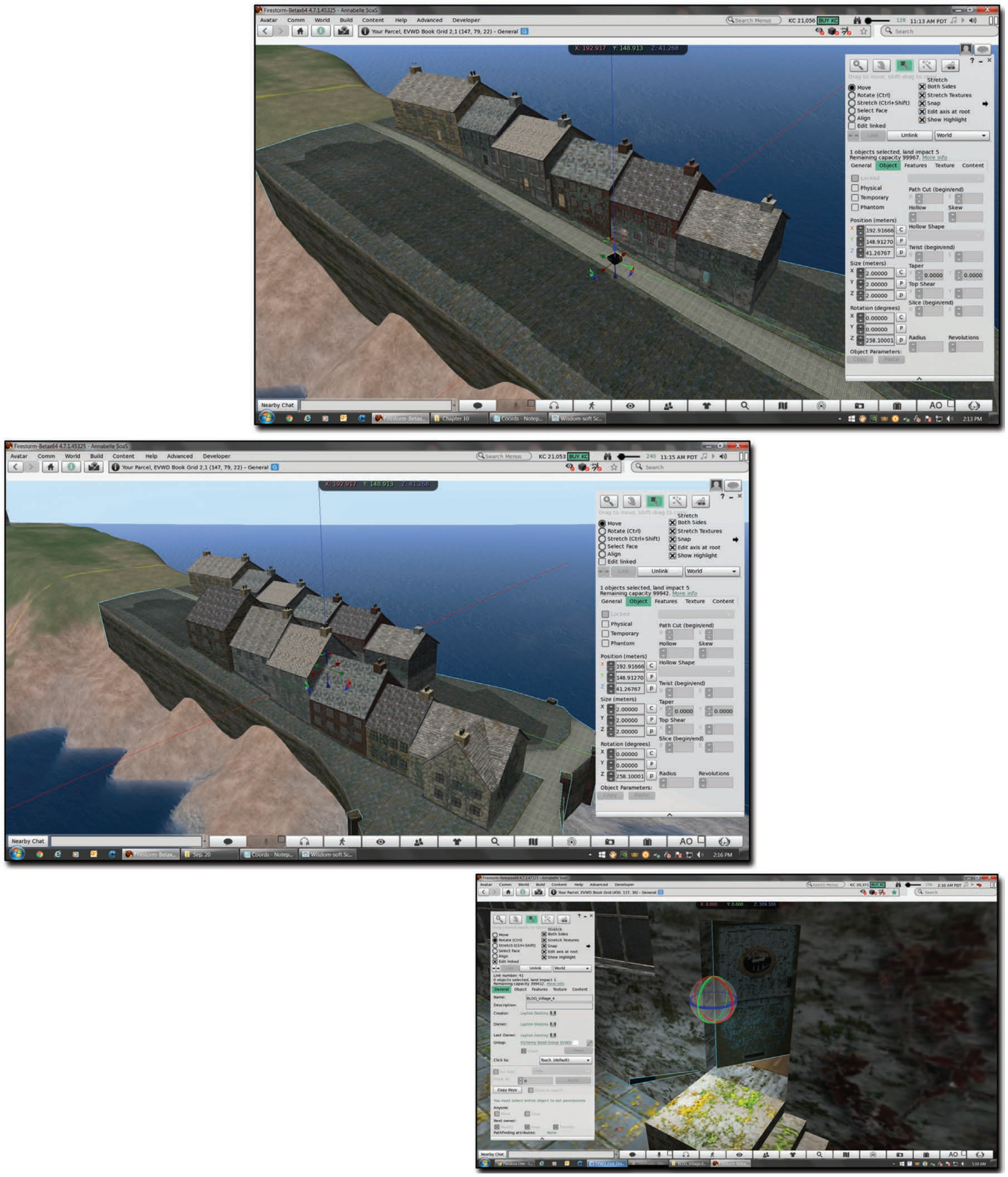
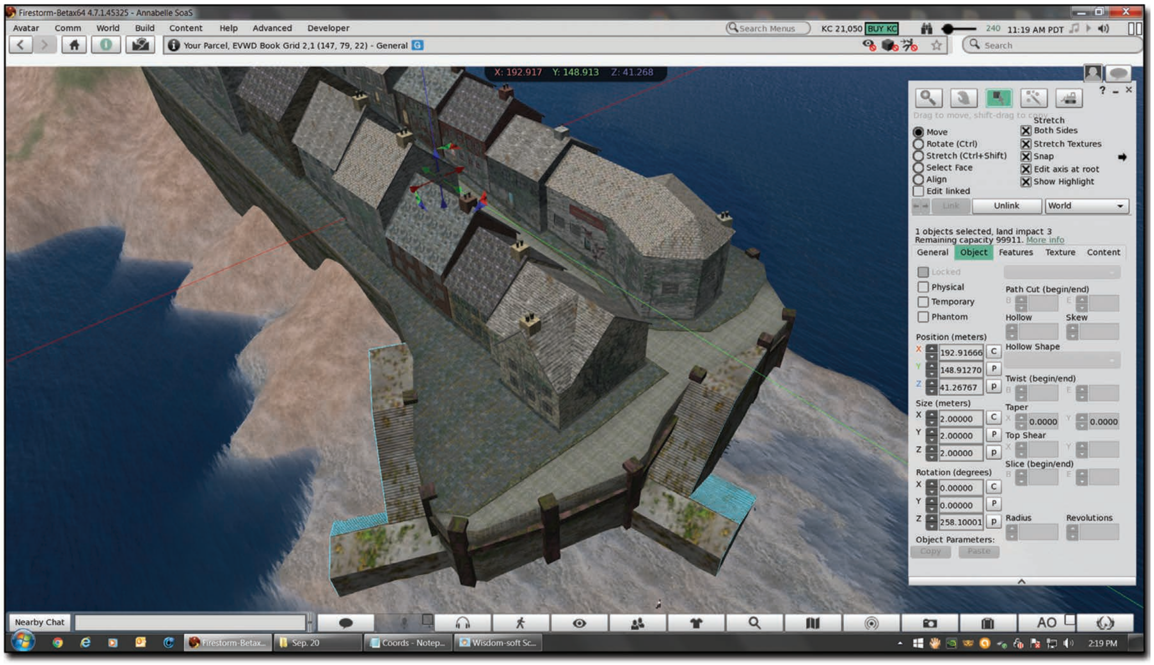


FIGURE 10.16 Village assembly and transparency settings, step 2.


Firestorm-Betax64 4.7.1.45325 - Annabelle SoaS
Avatar Comm World Build Content Help Advanced Developer
Your Parcel, EVWD Book Grid 2,1 (147, 79, 22) - General
KC 21,050 BUY KC
11:19 AM PDT
X: 192.917 Y: 148.913 Z: 41.268
Nearby Chat
2:19 PM

[● Move] Rotate (Ctrl) Stretch (Ctrl+Shift) Select Face Align
Edit linked
Stretch: [✓ Both Sides] [✓ Stretch Textures] [✓ Snap] [✓ Edit axis at root] [✓ Show Highlight]
Unlink World

1 objects selected, land impact 3
Remaining capacity 99911. More info

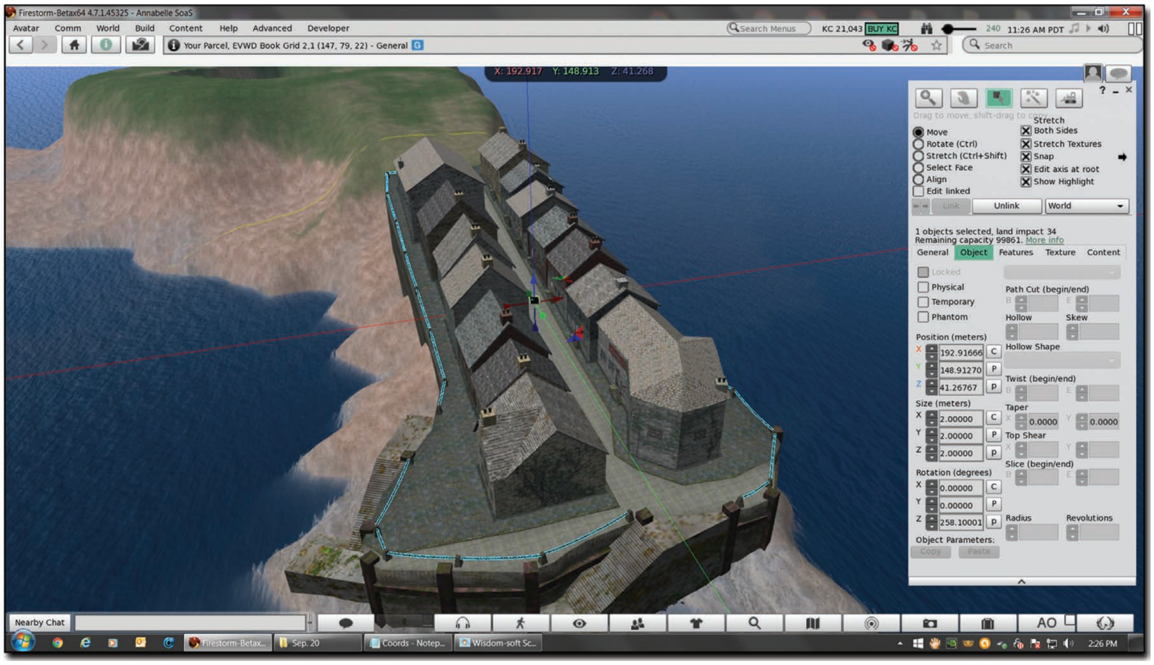
General **Object** Features Texture Content

Locked
Physical
Temporary
Phantom

Position (meters) X	192.91666
Position (meters) Y	148.91270
Position (meters) Z	41.26767
Size (meters) X	2.00000
Size (meters) Y	2.00000
Size (meters) Z	2.00000
Rotation (degrees) X	0.00000
Rotation (degrees) Y	0.00000
Rotation (degrees) Z	258.10001
Taper X	0.0000
Taper Y	0.0000

Path Cut (begin/end) Hollow Skew Hollow Shape Twist (begin/end) Top Shear Slice (begin/end) Radius Revolutions
Object Parameters:

FIGURE 10.17 Village assembly: exterior steps, step 3.


Firestorm-Betax64 4.7.1.45325 - Annabelle SoaS
Avatar Comm World Build Content Help Advanced Developer
Your Parcel, EVWD Book Grid 2,1 (147, 79, 22) - General
KC 21,043 BUY KC
11:26 AM PDT
X: 192.917 Y: 148.913 Z: 41.268
Nearby Chat
2:26 PM

[● Move] Rotate (Ctrl) Stretch (Ctrl+Shift) Select Face Align
Edit linked
Stretch: [✓ Both Sides] [✓ Stretch Textures] [✓ Snap] [✓ Edit axis at root] [✓ Show Highlight]
Unlink World

1 objects selected, land impact 34
Remaining capacity 99861. More info

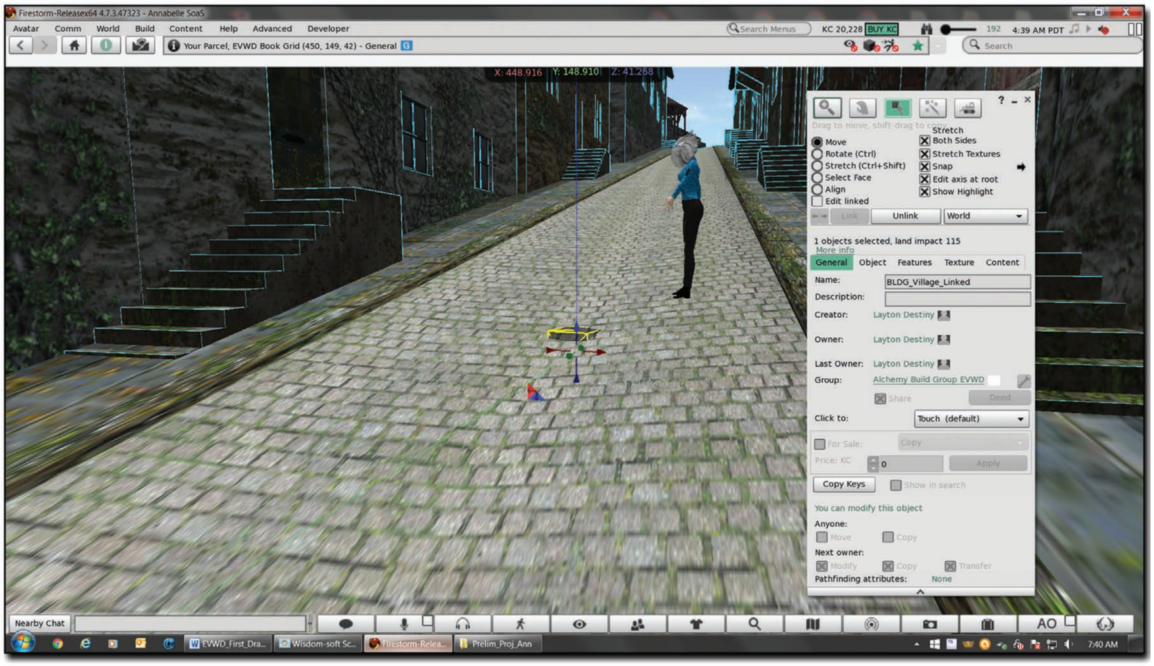
General **Object** Features Texture Content

Locked
Physical
Temporary
Phantom

Position (meters) X	192.91666
Position (meters) Y	148.91270
Position (meters) Z	41.26767
Size (meters) X	2.00000
Size (meters) Y	2.00000
Size (meters) Z	2.00000
Rotation (degrees) X	0.00000
Rotation (degrees) Y	0.00000
Rotation (degrees) Z	258.10001
Taper X	0.0000
Taper Y	0.0000

Path Cut (begin/end) Hollow Skew Hollow Shape Twist (begin/end) Top Shear Slice (begin/end) Radius Revolutions
Object Parameters:

FIGURE 10.18 Village assembly: piers and chains, step 4.


Firestorm-Releasex64 4.7.3.47323 - Annabelle SoaS
Avatar Comm World Build Content Help Advanced Developer
Your Parcel, EVWD Book Grid (450, 149, 42) - General
KC 20,228 BUY KC
4:39 AM PDT
X: 448.916 Y: 148.910 Z: 41.268
Drag to move, shift-drag to copy
Nearby Chat

Move [✓]
Rotate (Ctrl)
Stretch (Ctrl+Shift)
Select Face
Align
Edit linked

Stretch Both Sides ☒
Stretch Textures ☒
Snap ☒
Edit axis at root ☒
Show Highlight ☒

Link | Unlink | World

1 objects selected, land impact 115
More info

General | Object | Features | Texture | Content

Name: BLDG_Village_Linked
Description:
Creator: Layton Destiny
Owner: Layton Destiny
Last Owner: Layton Destiny
Group: Alchemy Build Group EVWD
☒ Share Deed
Click to: Touch (default)
For Sale: Copy
Price: KC 0 Apply
Copy Keys Show in search
You can modify this object
Anyone:
Move Copy
Next owner:
☒ Modify ☒ Copy ☒ Transfer
Pathfinding attributes: None

FIGURE 10.19 Village assembly linking and street props, step 5.

Step 5: Props and Décor in the Village (BLDG_Village_9)

The next components are the village street props and their related physics shapes. These props are optional; they are simply décor for the village environs. They contain sidewalk elements and such. There is a special physics file that you should use during the upload for BLDG_Village 9, because these props are being used on the walkways and need collision boundaries.

As you can see in Figure 10.19, once all of the village components have been placed, and all of the alignment cubes are at the same coordinates, you again have two options: (1) unlinking and deleting all of the alignment cubes (Build menu/Edit linked) in each of the separate components and then linking all of the village components together, letting the key prim be chosen by the system from one of the remaining elements; or (2) unlinking and deleting all but one of the alignment cubes, and then making that the key prim of the entire village in one big linkset. Option 2 will maintain a central point for you should you need to move the village somewhere else on the region. Since the remaining alignment cube is large, it is recommended that you move it up near the rooftops and make it completely transparent, so it will not interfere with any movement in the streets of the village.

10.5.3 Adding the Buttercross, Forge, and Stables Outbuildings

The next sets of components are individual outbuildings that are distributed across the landscape. Just north of the village is a buttercross, or town market meeting place. The coordinates for this component are X = 180.11084, Y = 17.76016, and Z = 65.37387, no rotation on any axis.

To the east of the castle on the same region is the forge outbuilding, and the coordinates for this are X = 15.45880, Y = 94.83426, and Z = 30.34831. The forge is rotated in Z = 250.00 degrees. To the west of the forge is the stables outbuilding, just over the regional border on the northwest region. The coordinates for the stable are X = 235.91023, Y = 145.26024, and Z = 30.91986. The stables outbuilding is rotated at Z = 270.00 degrees. In Figure 10.20, you can see screenshots of these three outbuildings in their places. The top image shows the location of the buttercross, and in the bottom image of Figure 10.20 you can see where the stable and forge have been placed.

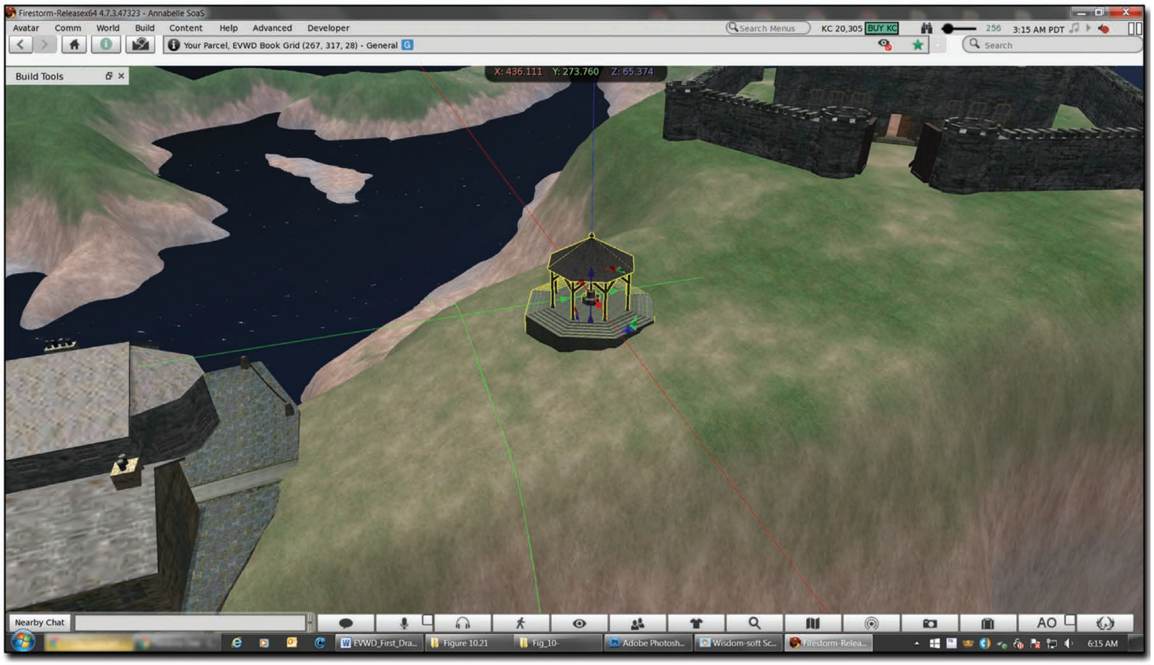
10.5.4 Smoothing the Terrain for a Perfect Fit

Once you have placed all the components, you may discover that some of the terrain is poking up where you do not want it to be. If you feel the need to lower the terrain a bit, open the Build menu and go into the Terrain Edit tools (the little bulldozer icon). With a very low setting in a small selection area, lower the terrain, as shown in the bottom image of Figure 10.21. Note: The ground cap that comes with the tower can be tinted temporarily if you need to have increased contrast between that ground cover and the virtual terrain.

10.5.5 Detailing the Environment with Landscaping

The last phase of this project is open to your individual creativity. We have included four trees with the COLLADA files of this project, and they can be distributed throughout the game-based sim over the landscape. These trees were made with Snappy Tree, an online tree generator found at http://www.snappytree.com/; and the branch graphic/textures were made with CanTree, another online tree generator found at http://arnaud.ile.nc/cantree/. The trunk graphic/textures, which were procedurally generated in Filter Forge, can be found in the Chapter 10 download section. These graphics and individual photos of some of our local tree leaves were utilized in the CanTree engine so the branches will match the Snappy Tree generated model. Once you upload the trees, you will need to upload and assign these graphic/textures to the branches and trunks using the Build menu/Edit linked/Texture tab settings. If you decide to make your own trees, keep the polygon count as low as possible (vertices around 3000–5000) and the branch images at 512 pixels × 512 pixels maximum. Otherwise, you will not be able to see the forest for the trees' server demands in your game-based sim. In Figure 10.22 is a screenshot of some initial landscaping around the castle.

As you do the landscaping on your game-based sim, keep an eye on the performance statistics. Remember you need to keep some spare server capacity available for the game elements you may choose to add in later.


Firestorm-Releasex64 4.7.3.47323 - Annabelle SoaS
Avatar Comm World Build Content Help Advanced Developer
Your Parcel, EVWD Book Grid (267, 317, 28) - General
Build Tools
X: 436.111 Y: 273.760 Z: 65.374
Nearby Chat

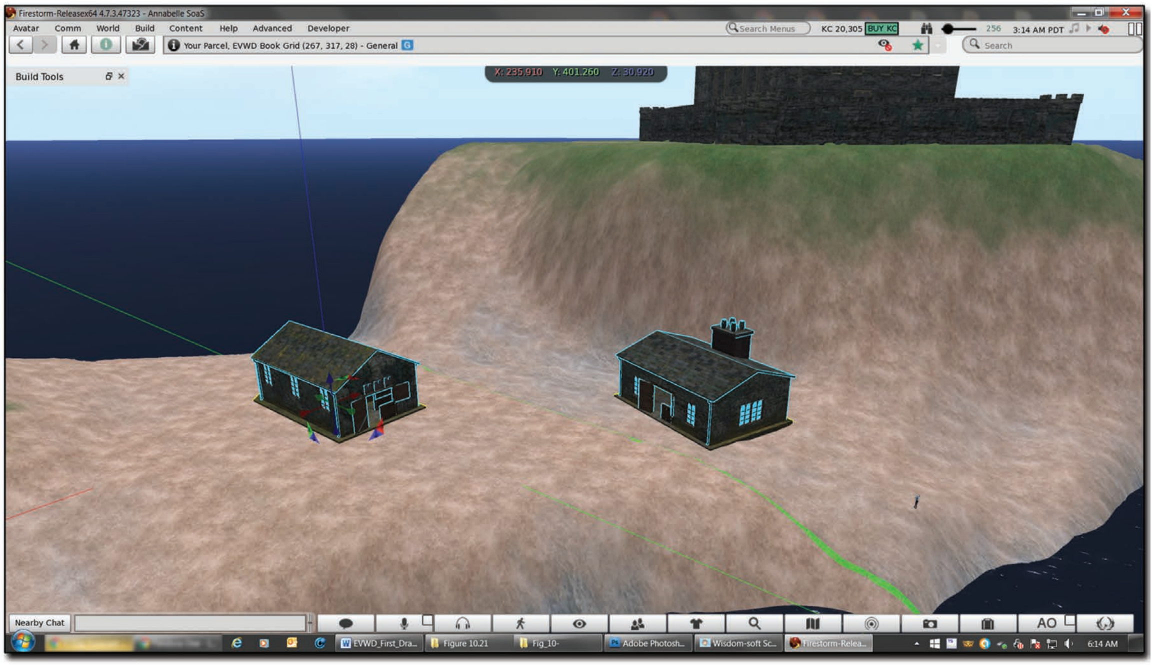

Firestorm-Releasex64 4.7.3.47323 - Annabelle SoaS
Avatar Comm World Build Content Help Advanced Developer
Your Parcel, EVWD Book Grid (267, 317, 28) - General
Build Tools
X: 235.910 Y: 401.260 Z: 30.920
Nearby Chat

FIGURE 10.20 Screenshots of buttercross, forge, and stable installation.

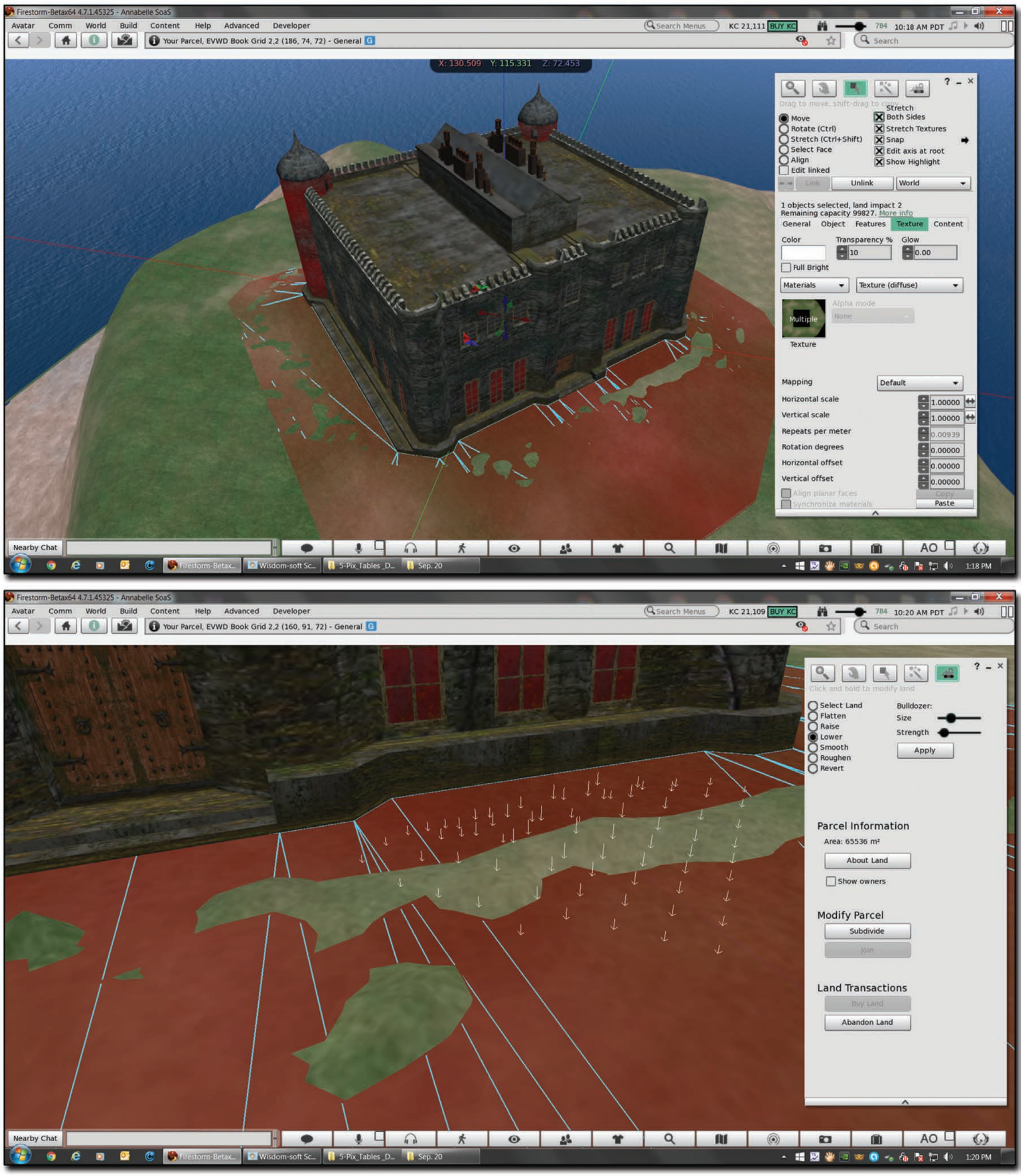


FIGURE 10.21 Using the terrain tools to smooth out the terrain around the castle.

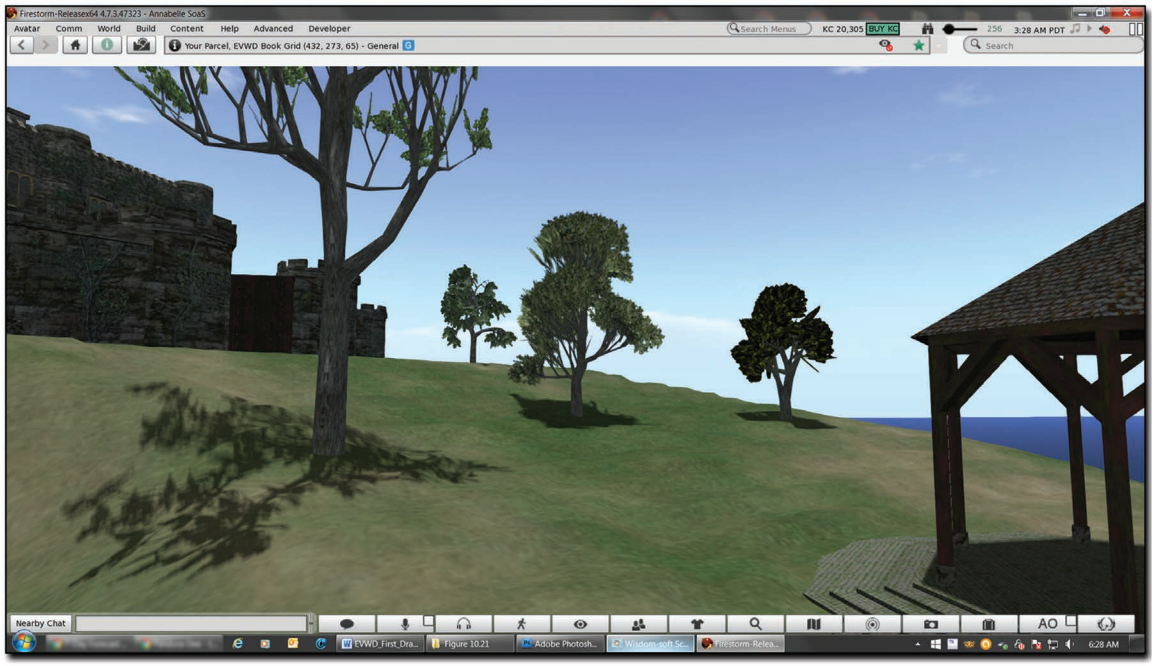

Firestorm-Releasex64 4.7.3.47323 - Annabelle SoaS
Avatar Comm World Build Content Help Advanced Developer
Your Parcel, EVWD Book Grid (432, 273, 65) - General
KC 20,305 BUY KC
3:28 AM PDT
Nearby Chat
AO

FIGURE 10.22 Starting some landscaping with four trees.

10.6 CHAPTER SUMMARY AND FINAL THOUGHTS

By necessity, this chapter has covered a huge subject from a very conceptual, "high concept" point of view. Obviously there is much, much more to the topic of meshes and how they are used in virtual game-based environments. If you have an interest in becoming a mesh expert, then take the time to learn one or more 3D modeling programs, such as 3ds Max, Blender, or Maya. There is a ton of information about them online and many tutorials about how to use them for creating virtual world content. In closing, here are some important things to remember about using meshes:

- Take your time and choose mesh-making methods that work with your project workflow.
- Hire an expert if you need one.
- Not everything needs to be 3D; think of distant objects as possibly being 2D images instead.
- Constantly test your meshes to make sure they are rezzing correctly and are behaving properly with the physics engine, and not taxing the server too much.
- Explore and combine different methods of creating and developing meshes, for instance, utilize scanned models as a base for creating more complex builds that contain other types of meshes (see Chapter 7).

REFERENCES

1. Jon Brouchoud, "Mesh Imports Coming to Second Life Q2," *Arch Virtual*, May 3, 2010, accessed May 17, 2014, http://archvirtual.com/2010/05/03/mesh-imports-coming-to-second-life-q2-2010/.

2. *Wikipedia*, s.v. "COLLADA," accessed February 27, 2015, http://en.wikipedia.org/w/index.php?title=COLLADA&oldid=646922759.
3. *Wikipedia*, s.v. "Photogrammetry," accessed February 28, 2015, http://en.wikipedia.org/w/index.php?title=Photogrammetry&oldid=645271406.
4. *Wikipedia*, s.v. "Digital Sculpting," accessed March 1, 2015, http://en.wikipedia.org/w/index.php?title=Digital_sculpting&oldid=623690864.
5. *Wikipedia*, s.v. "ZBrush," accessed March 1, 2015, http://en.wikipedia.org/w/index.php?title=ZBrush&oldid=648416581.
6. *Wikipedia*, s.v. "3D-Coat," accessed March 1, 2015, http://en.wikipedia.org/w/index.php?title=3D-Coat&oldid=637943735.
7. Treinando Pesado, "How to Make 3d Scan with Pictures and the PPT GUI," *ATOR* (blog), December 28, 2012, accessed February 25, 2015, http://arc-team-open-research.blogspot.com/2012/12/how-to-make-3d-scan-with-pictures-and.html.
8. "Mesh/Rigging Fitted Mesh," *Second Life Wiki*, accessed March 6, 2015, http://wiki.secondlife.com/w/index.php?title=Mesh/Rigging_Fitted_Mesh&oldid=1191101.
9. *Wikipedia*, s.v. "Skeletal Animation," accessed March 8, 2015, http://en.wikipedia.org/w/index.php?title=Skeletal_animation&oldid=641453035.
10. *Wikipedia*, s.v. "Inverse Kinematics," accessed March 8, 2015, http://en.wikipedia.org/w/index.php?title=Inverse_kinematics&oldid=630661339.

This page intentionally left blank

11 Designing with Advanced Materials and Animated Graphic/Textures

I love mirrors. They let one pass through the surface of things.

Claude Chabrol

11.1 OVERVIEW OF DESIGNING WITH ADVANCED MATERIALS AND ANIMATED GRAPHIC/TEXTURES

Graphics, shaders, and materials used on meshes create the unique visual identities of the avatars, objects, and buildings of your virtual world. The color, shine, roughness, and patterns in the graphics you use contribute their specific visual qualities to the overall appearance of the surface. In the Build/Edit menu under the Texture tab, there are three major categories of graphic/texture layers for you to utilize in your creations:

- The Texture (diffuse) layer is a graphic/texture used to bring in the tones and shades of the object's surface, and can be photorealistic or illustrative.
- The Bumpiness (normal) layer will create a visual impression of the bumps and wrinkles on the surface without adding additional geometry to the underlying mesh.
- The Shininess (specular) layer is often derived from the Texture (diffuse) graphic; this could be used to add a special layer of highlights or shiny pattern to the surface.

All the objects in your virtual environment can utilize those layers of graphic/textures to expand the spirit of your design and enhance the message of a virtual environment. In addition to these qualities, the Texture (diffuse) layer on your object can be animated by scripting. Some of the basic effects in this category are sliding movement (e.g., running water), rotation (e.g., pinwheels or windmill blades), ping-pong movement (e.g., opening and closing of curtains), and frame animation that is made from a series of images (e.g., flames flickering). The usage of these kinds of advanced materials requires that the graphic/textures be applied to the mesh with precision registration, and this can be done with the use of UV mapping and UV unwrapping features that are available in outside 3D modeling software such as Blender or 3ds Max. In this chapter, you will explore the following concepts related to Advanced Material textures that contain Texture (diffuse), Bumpiness (normal), Shininess (specular), and animated graphics:

- Graphic/textures, shaders, and materials function as a team to create the surface look of a mesh object.
- UV mapping and UV unwrapping is an essential skill necessary to getting proper registration of your advanced materials on the mesh object.

- Virtual world client viewers can affect the look of a graphic/texture significantly, so your design should consider their settings and usage by the visitor.
- Shininess (specular) layers can embellish the surface, adding dimension and highlights to the appearance of an object if the Advanced Lighting Model is activated.
- Alpha modes are important parts of Texture (diffuse) layers and can be used in a variety of ways.
- Movement, animation, and video-based media are also part of Texture (diffuse) graphics in the virtual environment.

If you need to review the basics of materials and shaders used in a virtual environment, please refer to Chapter 6, Section 6.6 of the book *Virtual World Design*. Throughout the next sections, we are going to focus on the concepts and techniques involved with developing and applying these graphics/textures to create advanced materials. At the end of this chapter, you will continue on the construction of the game-based environment that was started in Chapter 4. You will be adding Bumpiness (normal) and Shininess (specular) layers as well as an animated texture on the Fairy Bridge provided.

11.2 THE STRUCTURE OF AN ADVANCED MATERIAL USED IN VIRTUAL ENVIRONMENTS

Essentially the formula for a material is

Graphic/texture + Shader (built in rendering parameters) = Material

Graphic/textures can take many forms and each one can be influenced by the shader differently to create a specific effect in the material. As you can see in Figure 11.1, there are some interesting kinds of graphics that can be used in a virtual environment, in addition to the standard photographic image. The Materials Project (and its related viewer) was launched in mid-2013, allowing resident creators of Second Life to utilize custom Bumpiness (normal) and Shininess (specular) graphic/textures on their creations in the Build/Create/Texture menu. Firestorm, Singularity, and other client viewers promoted the spread of advanced material usage throughout OpenSim. Prior to this improvement, creators could only use some built-in stock Bumpiness (normal) textures, which were usually not specific to any custom objects they were building. Now you can bring in your own Bumpiness (normal) graphic generated to match the Texture (diffuse) graphic you are using for your creations. In other words, your custom rock wall will have a matching surface relief or cragginess on it. You can also create and upload a Shininess (specular) graphic, which will allow you to control the shininess and highlights of the object's surface. As with all of the more complex rendering effects, these materials can only be displayed when the Advanced Lighting module is turned on in the viewer (under Preferences/Graphics in the Firestorm viewer), so using them will demand more processing from your graphics card. When these special textures are applied to an object, primitive, or imported mesh in your virtual world, you can affect the way they are displayed in the viewer by using the Build/Texture menu options to change the settings called opacity, glow, brightness, and underlying color. These four settings in the Build/Edit/Textures menu represent the Shader part of the equation:

Graphic/texture + Shader = Material

All of these shader settings combined with the graphic/texture layers—Texture (diffuse), Bumpiness (normal), and Shininess (specular)—will affect the way materials in your virtual environment are being displayed to the visitor through their client viewer.

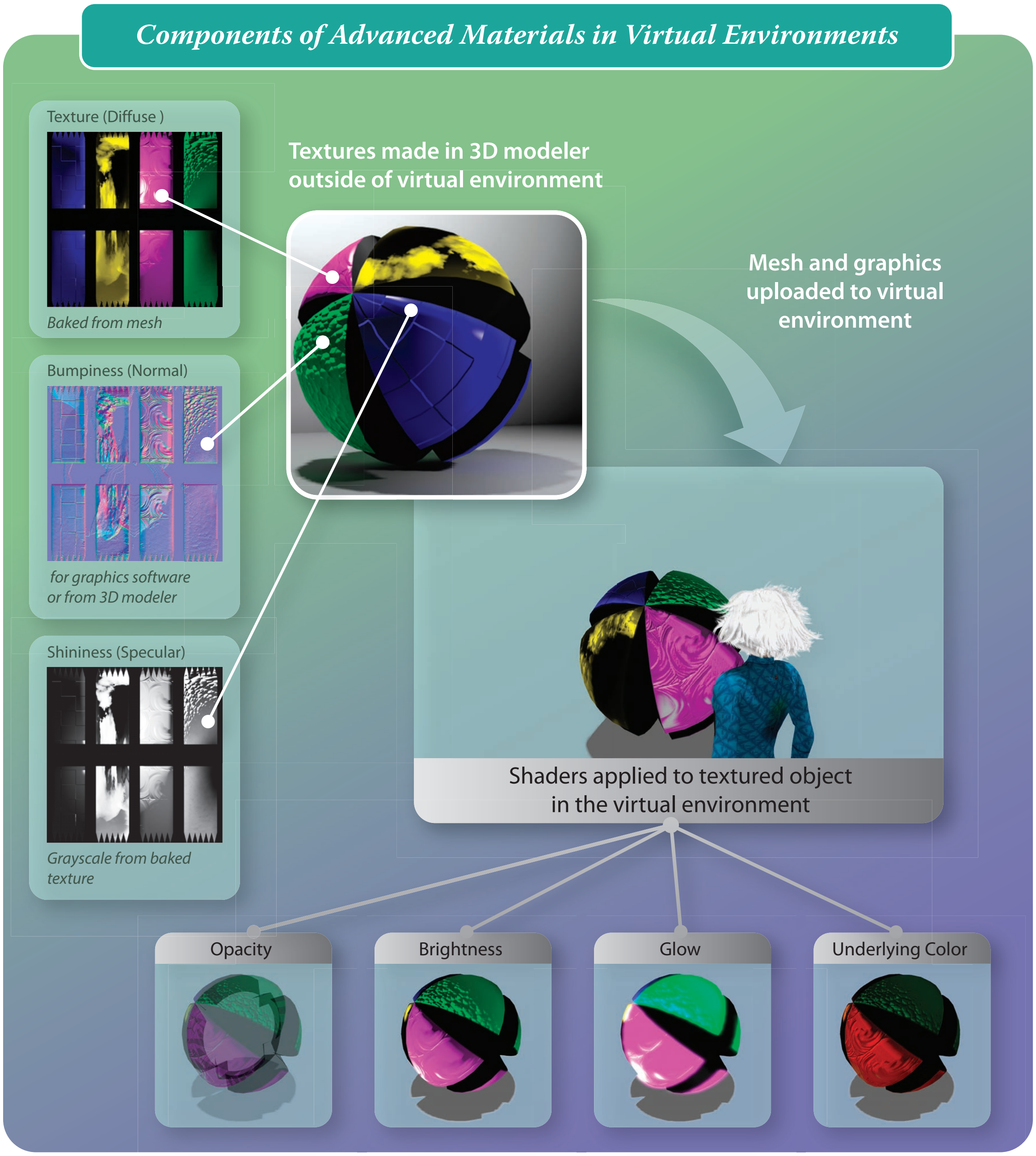

Components of Advanced Materials in Virtual Environments
Texture (Diffuse)
Baked from mesh
Bumpiness (Normal)
for graphics software or from 3D modeler
Shininess (Specular)
Grayscale from baked texture
Textures made in 3D modeler outside of virtual environment
Mesh and graphics uploaded to virtual environment
Shaders applied to textured object in the virtual environment
Opacity
Brightness
Glow
Underlying Color

FIGURE 11.1 The interrelationship of graphic/textures and shaders on a mesh object in the virtual environment.

11.3 DESIGNING WITH ADVANCED MATERIALS IN MIND

Here are guidelines you should keep in mind when you are designing a virtual environment full of objects with advanced materials on them:

1. Graphic size should always be 512 pixels by 512 pixels or lower, unless you have a huge object that has to be highly detailed, such as multipart meshes that share one texture over many faces. Other exceptions may be signage with lots of legible text or a graphic/texture that will be subdivided into frames for animation on a mesh face inworld.
2. Streamline your graphic creation system to utilize one or two programs in which to make all of your graphics. For instance, if you are working from a photograph-based graphic in a 2D program like Photoshop or GIMP, then make the Bumpiness and Shininess graphics in that program by utilizing the plugins that are available. If you are working in a 3D program, then use the built-in Bumpiness (normal) and Shininess (specular) graphic generators available with it, and sweeten them in your favorite 2D graphics program, if necessary.
3. Ask yourself if the Bumpiness (normal) and Shininess (specular) graphic application on each object is really necessary for the overall aesthetic of the environment. While most objects in the real world have some sort of shine and surface quality (rough/smooth) to them, copying all of that into a virtual environment may just lag the performance of the sim and negatively affect the visitor's experience on it. Be judicious in your use of these graphics and strive for maximum visual impact with minimal simulation server processing needs.

11.4 UV UNWRAPPING AND UPLOADS OF MESHES WITH ADVANCED MATERIALS

When you are designing with advanced material, one of the most powerful methodologies to understand and utilize involves the usage of UV unwrapping. With this you can "unwrap" the skin of any mesh model you have created, flatten it out, paint detail and decorations on it, and rewrap it back on the surface of the model inworld. This is a very artful technical process, and there are many ways to achieve it through 3D modeling tools available to use. In the following section is a short introduction to the workflow that two designers use in the 3D modeling programs 3ds Max and Blender, as they create the mesh model, unwrap and work with the UVW mapping, bake the surface detail into them, and then upload the graphics and mesh into a virtual environment. For comparative purposes, the "push/pull" face modeling software, SketchUp, has been included.

11.4.1 Overview of the Workflows and Software

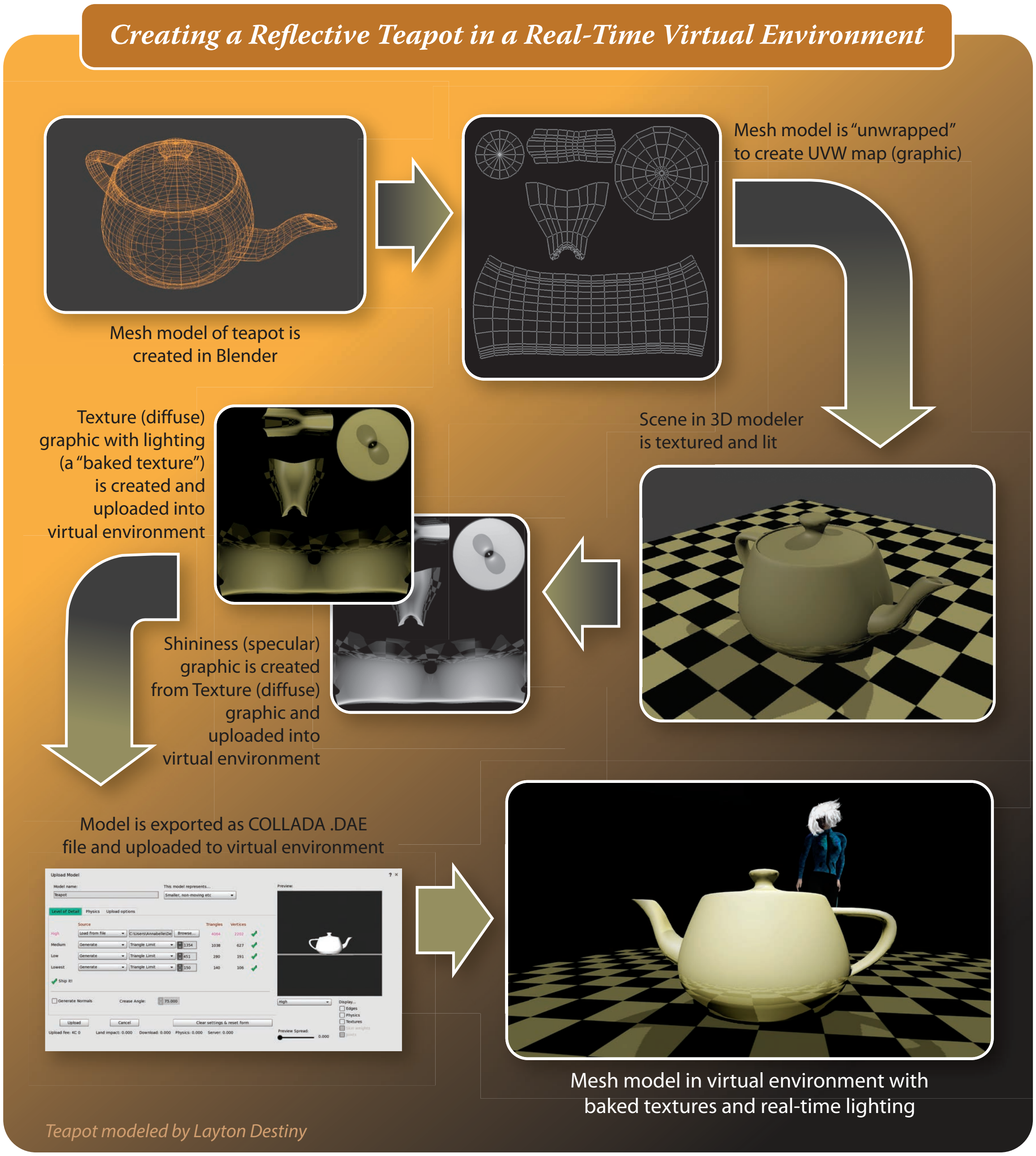
In Table 11.1 is an overview of the workflow methodology used in creating the components for a prelit surface graphic and mesh that will be uploaded into the virtual environment. As you can see, the process is almost identical when you compare 3ds Max and Blender, which use a process called *UVW mapping* and *UVW unwrap*. The standard SketchUp program, which uses face mapping, will not provide you with a complete set of tools for creating prelit surfaces; it doesn't offer a way to unwrap your mesh models.

11.4.2 The Reflective Teapot Scene: A Breakdown

In Figure 11.2 is a series of images showing a simple still-life scene created in Blender by Layton Destiny, our ace model maker, and then moved into the virtual environment. The initial model, a basic teapot, is made in Blender, and reflection-creating materials are applied. The lower half of the teapot is reflecting the

TABLE 11.1
Overview of Mesh UV Unwrap/Import Methods: Comparing 3ds Max, Blender, and SketchUp

	3ds Max	Blender	SketchUp	Notes
1	Make or obtain mesh model	Build or acquire your mesh model	Make or obtain mesh model	
2	Apply materials and light the model	Mark seams for UV unwrap	Apply graphics to the faces of the model	Consider what kind of real-time lighting you will be using in the virtual environment as you light this model
3	If model is a complex form add the Unwrap UVW map modifier to the mesh, and organize the map in the UVW editor, otherwise Render to Texture will do it automatically	Select all vertices and press U and select your unwrap method	Export mesh as COLLADA (*.DAE); set the export to include graphics which are face mapped, not UVW mapped	Set the SketchUp export to include graphics; 3ds Max and Blender will not include graphics in their COLLADA export
4	Use Render to Texture menu to render out the baked graphic as a UV map	Add a blank UV material to your UV map (New in the UV/Image editor)	–	–
5	Upload mesh model into virtual world	Add your lights	–	–
6	Upload the baked UV map graphic into virtual world	Select your mesh and press Bake in the Properties/Render menu	Upload Mesh model into Virtual World	–
7	To reunite the mesh and its baked graphic: Add UV map graphic into Texture (diffuse) slot in the Build/Edit menu of Client Viewer	Save your baked image	Graphics should come along with mesh, no need for separate upload	Any changes or tweaks to the baked UV map should be done in Photoshop before this step
8		Select your mesh and save as .dae (File/Export/COLLADA) using the Second Life Static operator preset for static mesh and the Second Life Rigged operator preset for rigged meshes	–	–
9		Upload mesh and map to your virtual environment and put map into the Texture (diffuse) slot	–	–


Creating a Reflective Teapot in a Real-Time Virtual Environment
Mesh model of teapot is created in Blender
Mesh model is "unwrapped" to create UVW map (graphic)
Scene in 3D modeler is textured and lit
Texture (diffuse) graphic with lighting (a "baked texture") is created and uploaded into virtual environment
Shininess (specular) graphic is created from Texture (diffuse) graphic and uploaded into virtual environment
Model is exported as COLLADA .DAE file and uploaded to virtual environment
Mesh model in virtual environment with baked textures and real-time lighting
Teapot modeled by Layton Destiny
Upload Model
Level of Detail
Physics
Upload options
Preview:
Browse...
Upload
Cancel
Clear settings & reset form
Display...
Edges
Physics
Textures

Model name: Teapot

This model represents... Smaller, non-moving etc

	Source			Triangles	Vertices	
High	Load from file	C:\Users\Annabelle\De		4064	2202	✓
Medium	Generate	Triangle Limit	1354	1038	627	✓
Low	Generate	Triangle Limit	451	280	191	✓
Lowest	Generate	Triangle Limit	150	140	106	✓

✓ Ship it!

☐ Generate Normals Crease Angle: 75.000

High

Upload fee: KC 0 Land impact: 0.000 Download: 0.000 Physics: 0.000 Server: 0.000

Preview Spread: 0.000

FIGURE 11.2 Teapot methodology, Blender build.

checkerboard pattern of the tabletop. This kind of surface reflection is not currently available in a real-time rendered environment like a virtual world, but we can bring in graphics that imitate the effect and deepen the quality of the virtual scene. When the surface of the teapot looks good, its graphic coordinates are unwrapped and flattened using the tools specific to this purpose (see Table 11.1). After that, the corresponding graphic is *baked* or rendered to preserve all this surface detail, the reflected checker floor, the highlights, the surface shine, and so forth. Finally, the graphic and mesh are uploaded and reunited in the virtual environment and placed on a built surface that has the same checker graphic that was used in the Blender model. When the Advanced Lighting is turned on, some basic shininess is added to the teapot texture, and some projector lights are included in the scene to create shadows on the checkered surface from the teapot, the result inworld is comparable to the original scene in Blender.

11.4.3 Build Your Knowledge, Develop the Art of UV Unwrapping and UV Maps

The best way to get comfortable with making UV maps and unwrapping them properly from a mesh model is to do some. Online are numerous tutorials for whatever 3D software you choose to use. Dive in and give it a try; you can get some fantastic results!

11.5 VIRTUAL WORLD VIEWERS AND ADVANCED MATERIALS

The first thing you should test before proceeding any further with this chapter is your viewer's functionality with these advanced materials. Follow the steps in this mini-project to test your results. In Figure 11.1 are images to guide you along the way.

11.5.1 Mini-Project: Test Your Viewer for Advanced Materials

1. Download the Chapter 11 test mesh and materials called Sphere.dae, Sphere_diffuse_map.tga, Sphere_normal_map.png, and Sphere_specular_map.png from http://www.anncudworthprojects.com/ under the Extending VWD tab, and upload this model and the three graphics into your avatar's inventory. Note: On the mesh sphere upload, you may want to scale up the sphere, if this is for a group demo.
2. Make sure your viewer has the Preferences/Graphics/Advanced Lighting module turned on, and that it is set to cast shadows from the sun and moon.
3. In an empty clear area, such as a sandbox, drag the sphere from your inventory onto the ground to rezz it. Apply the Sphere_diffuse_map.tga in the Texture (diffuse) slot. With that in place, the sphere's eight projected surfaces should exhibit four very different textures. Check your results by comparing it with the images in Figure 11.1.
4. In the pull-down menu of the Build/Edit/Texture tab, activate the Bumpiness (normal) menu and drag the Sphere_normal_map.png graphic into the slot provided. Look at the sphere from several angles, change the lighting/environmental settings, and notice how the textures on the surface of the sphere look more detailed.
5. Change over to the Shininess (specular) tab and add Sphere_specular_map.png into the slot provided. Using the World/Sun Position/Midday menu, set the time of day to Noon, and rotate the ball to see how the specular highlights from the sun move across the surface of the ball. This can be made more noticeable by changing the color under the specular settings to red or some other color that contrasts with the Texture (diffuse) base color. Try adjusting the glossiness and environment settings as well to see how that affects the specular highlights.

6. Also notice that you have some alpha mode options with the Texture (diffuse) menu. Since your Texture (diffuse) graphic on this sphere has an alpha channel, you can activate these settings. Toggle through them and see what kind of differences they make to the appearance.
7. Now go to Midnight, create a primitive 1 meter away and make it a projector light source using the Features tab in the Build/Edit menu and a simple white square texture. Take notice of how these advanced materials change under the projector light and how the specular highlights move across the surface as you rotate the sphere.

As you can see, these graphics add lots more visual detail and specular reactivity to the surface of a simple object. Open your statistics bar (Advanced/Performance Tools/Statistics Bar), and check to see if viewing an object like this affects the frame rate displayed on your computer. Zoom in and out, and change the lighting to see if anything negatively affects the performance.

11.6 UNDERSTANDING BUMPINESS (NORMAL) GRAPHICS AND WHAT THEY CONTRIBUTE TO A MATERIAL

Normal is a funny word in the English language and can mean conformity or even naturally occurring. However, in mathematics, the word normal means perpendicular. The normal of a geometric triangular plane is a line coming from the face and extending out in a perpendicular direction. In the real world, you might think of the Eiffel Tower as the normal line on the geometrical face of the Paris landscape. That famous perpendicular tower points upward from the surface of the Champ de Mars while its four legs stand on the plane. If you were to represent the earth as a geodesic sphere created from numerous triangular polygons and gave every major city such a tower, then the normals of a sphere would stand out in clear relief, pointing out in all directions from the curving geometrical surface of your globe. Knowing the direction these normals point in relationship to the main light source allows your computer to calculate how the surface of the sphere should be shaded; creating a sphere that is brightly lit on one side and gradually darkening toward the other side as the light fails to reach those surfaces. This normal information can be captured within a normal graphic, which essentially converts the spatial information, the x, y, and z coordinates of the surface normal into the color coordinates on the RGB color scale, and creates a flat graphic image from it. In Figure 11.3 you will see a screengrab of a 3D model of crumpled paper created in 3ds Max and the normal graphic made from it. Inworld, the main light source (sun/moon) is set at an angle to cast some shadows on the sides of the surface bumpiness. A normal graphic has its limits however. It cannot create great mountainous extensions from an otherwise flat plane; it is best used to add surface roughness and detail to a simplified mesh structure. In the bottom left image of Figure 11.3, notice how the normal graphic created does not show the shadowing but does show the overall shape of the forms. As you can see in the bottom right image, the effect is enhanced when the normal graphic is combined with a Texture (diffuse) graphic that has the same shadows baked onto its surface.

11.6.1 Bumpiness (Normal) Graphic Creation Methodologies

Essentially Bumpiness (normal) graphics can be generated from two kinds of sources: 2D images or 3D models. How you organize your graphics *pipeline* or workflow depends on from where you are starting. In Figure 11.3 is an illustration of the 2D–3D–2D workflow. As you can see in the top images, the initial source for the Texture (diffuse) graphic is a 2D photographic image of the crumpled paper captured with a digital camera. The initial image of the crumpled paper was cropped and slightly enhanced for contrast and brightness in Photoshop and then uploaded into 3ds Max and applied to a very thin box model. This image was also used

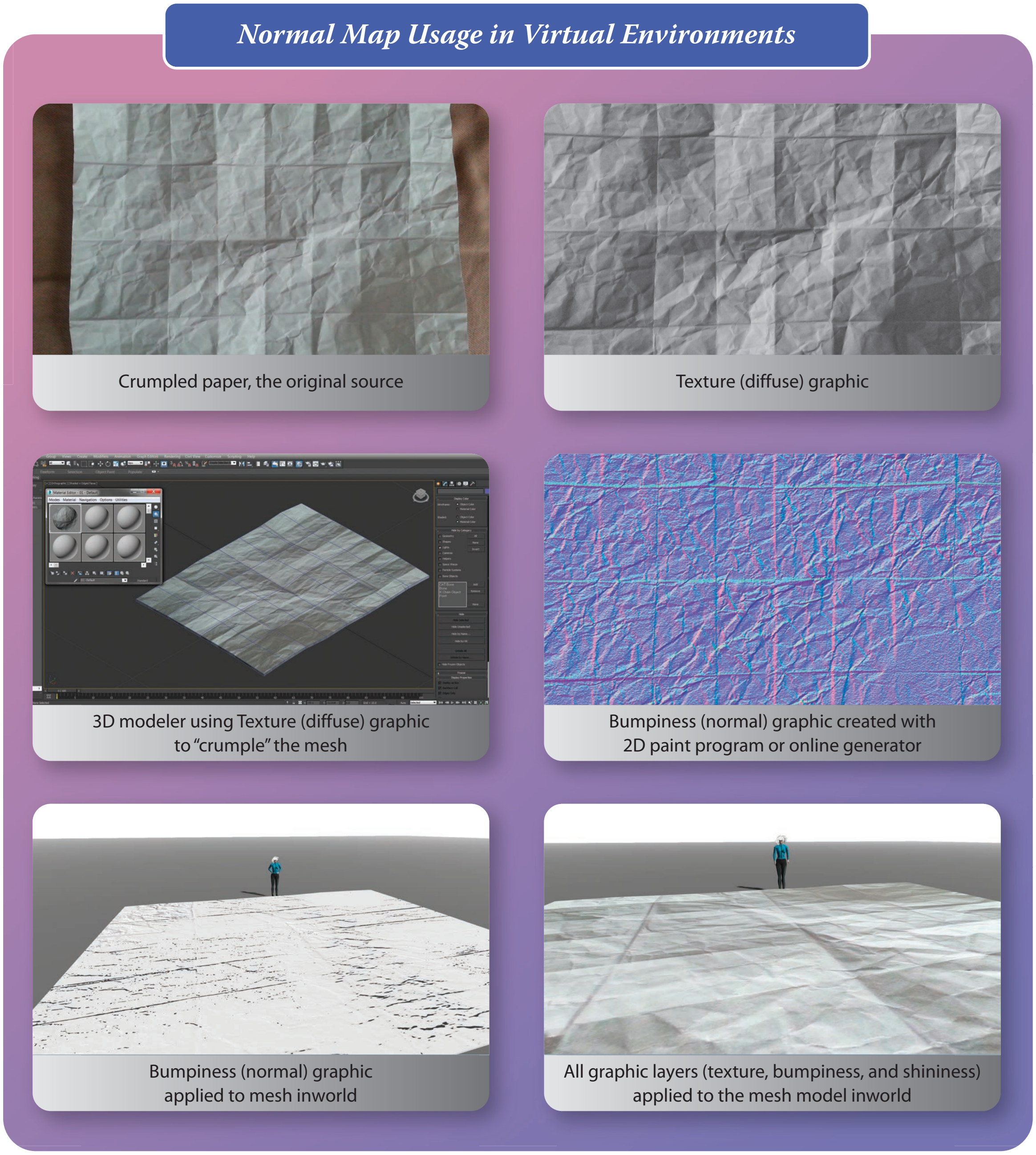

Normal Map Usage in Virtual Environments
Crumpled paper, the original source
Texture (diffuse) graphic
3D modeler using Texture (diffuse) graphic to "crumple" the mesh
Bumpiness (normal) graphic created with 2D paint program or online generator
Bumpiness (normal) graphic applied to mesh inworld
All graphic layers (texture, bumpiness, and shininess) applied to the mesh model inworld

FIGURE 11.3 Creating Bumpiness (normal) graphic/textures.

on the model as a “displacement modifier,” acting much like a heightmap would act on terrain, causing the many vertices of the box model to warp and shift like a crumpled form. From this point, the crumpled paper model in 3ds Max was lit, and a graphic was rendered with the light and shadows created on the surface by lights in the 3D modeling environment. This kind of image is referred to as baked. The initial baked graphic was flattened for export and use in the Texture (diffuse) layer on a mesh in the virtual world environment. The baked graphic was also brought into Photoshop, and a Bumpiness (normal) graphic was generated, as you can see in the middle image of Figure 11.3. There is also another approach to this process. The steps for model making and baking an image in 3ds Max could have been skipped. The initial photograph could have been loaded into a 2D graphics editor such as Photoshop or GIMP, and used to generate a Bumpiness (normal) graphic with the related plugins, but the lighting would have been set at what was captured in the initial picture. The simplest workflow for this in Adobe Photoshop is to utilize a graphic plugin like NVIDIA texture tools available for download from https://developer.nvidia.com/nvidia-texture-tools-adobe-photoshop. For creators who use GIMP, there is a normal map (Bumpiness) graphic generating plugin found at https://code.google.com/p/gimp-normalmap/. There is even a way to create Bumpiness (normal) graphics without any 2D graphics programs. Assuming you already have a clean grayscale heightmap to work with, you can utilize an online Bumpiness (normal) graphic creator such as Normal Map Online (http://cpetry.github.io/NormalMap-Online/), and generate it then and there. Shadermap (http://shadermap.com/home/) and Crazy Bump (http://www.crazybump.com/) are low cost programs that will do the job too. 3D modeling software like Blender (http://www.blender.org/) and 3ds Max (http://www.autodesk.com/products/3ds-max/overview) have built-in Bumpiness (normal) graphic generators, and tutorials for using them can be found online.

11.7 UNDERSTANDING THE SHININESS (SPECULAR) LAYER AND WHAT IT CONTRIBUTES TO A MATERIAL

When the movie companies want to create a spectacular night shot in the dark alleys of Manhattan, New York, they spray the street surface with water. By doing this, the movie crew instantly creates a whole environment full of specular highlights. All the glows from light sources such as neon signs and streetlights are reflected on the sidewalks and roadway, and where puddles have been created, a mirrorlike effect occurs. The brightest reflection of light is the specular highlight, and as the environment starts to dry off, you will see this fade away until all that remains is the look of a matte surface with some low reflectance here and there. Specular graphics work the same way as spraying something with water or oil—they bring out the shine. They can be used to great effect on the surfaces of your virtual props, vehicles, and clothing, as demonstrated in this simple example, a case study for making a sequined toy bear (http://wiki.secondlife.com/wiki/Case_Study_-_Sequined_bear_using_materials). In this particular instance, the specular graphic for the sequins was made first, and then a Bumpiness (normal) graphic was created from that, using MS Paint as the graphics program. Simple but effective, as you can see in the final results shot. More complex projects will likely require a more advanced graphics program like Photoshop or GIMP.

11.7.1 Shininess (Specular) Graphic Creation Methodologies

OK, without getting too “chicken or egg” about it, you will need to decide on the best workflow for creating your Shininess (specular) graphic and where that falls into your Bumpiness (normal) graphic creation process. In Figure 11.4 is one workflow in 3ds Max demonstrating how to go from a real object to a digital version of it, and how a custom Shininess (specular) graphic (five-point star highlights) is used. As you can see, much like the previous figure, this methodology starts in the real world with an image for reference.

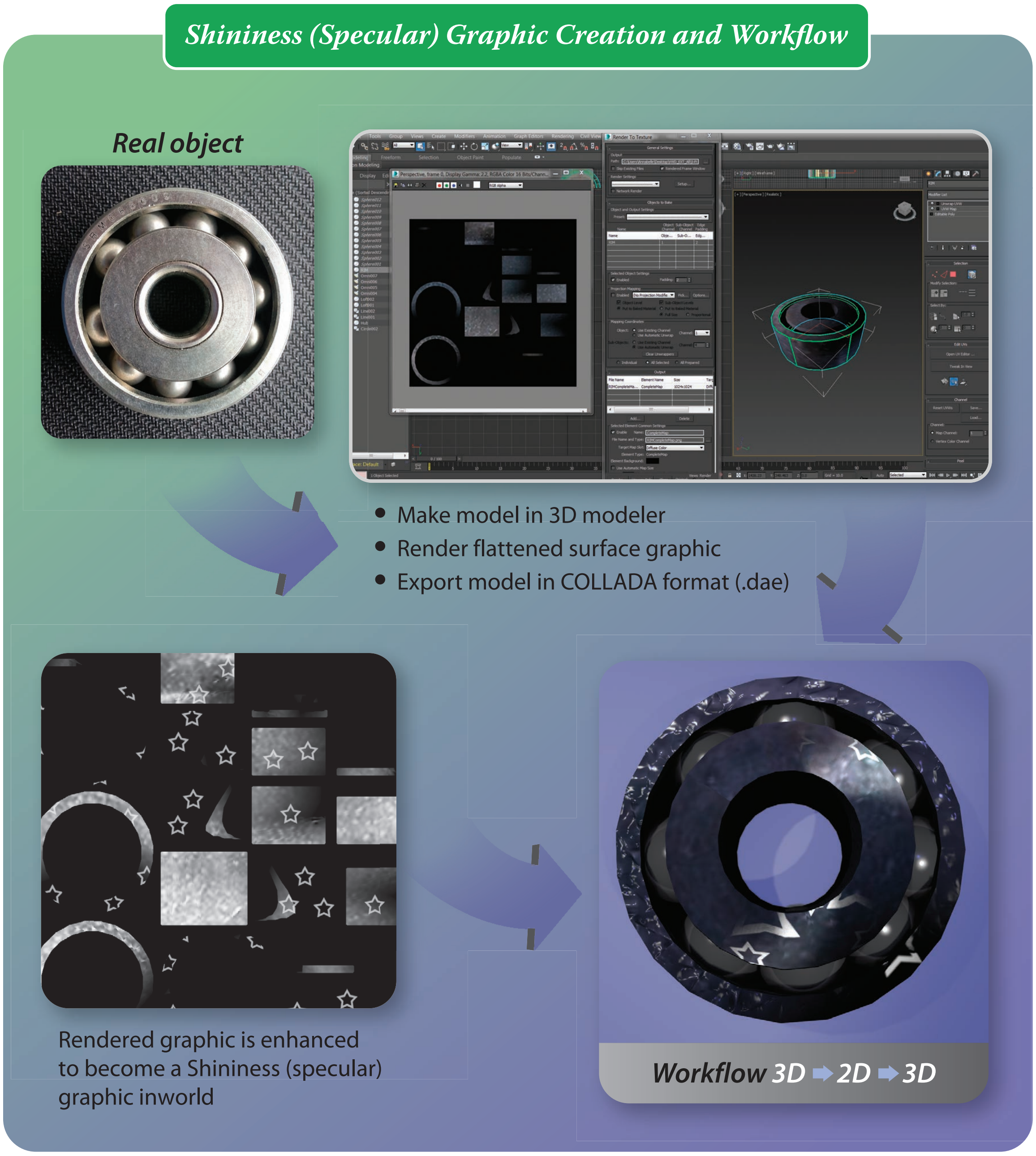

Shininess (Specular) Graphic Creation and Workflow
Real object
• Make model in 3D modeler
• Render flattened surface graphic
• Export model in COLLADA format (.dae)
Rendered graphic is enhanced to become a Shininess (specular) graphic inworld
Workflow 3D ➡ 2D ➡ 3D

FIGURE 11.4 Shininess (specular) graphic/texture creation methodologies.

The object is modeled in 3ds Max, and the graphic is baked, flattened, and exported as a 2D element. The baked graphic was transformed into a grayscale image, and "stamped" with white stars. The model, baked Texture (diffuse) graphic, a Bumpiness (normal) graphic, and the Shininess (specular) graphic with the white stars were all uploaded into the virtual environment. In the bottom left picture is the result. We see a virtual object that exhibits shiny five-point stars when it is lit with a projector light. Think of what this could do for a brocade jacket or a mosaic floor! This is just one way to approach Shininess (specular) graphics and texture usage; you will probably find the way that works best for your project with some experimentation.

In practice, the creation of Shininess (specular) graphics is more of an art than a science, as you will discover when you try making one by desaturating the Texture (diffuse) graphic to gray scale and applying it as a Shininess (specular) texture. This almost works in some cases, and fails in most. There are often too many grays in the desaturated image to generate good clear specularity on the surface of your object. Adjusting the levels to clip these gray values, and pushing the contrast will help clarify a mushy looking specular graphic. However, good specular graphics start with a clear concept of how the lighting highlights should look, how they flicker across the surface, and what kind of color they should display on the surface of the object you are modeling. Pick up something metallic from your desk and hold it under a desk light, move it around and see where the highlights travel to, and then try to imagine how you would make a Shininess (specular) graphic to show this flickering quality on a virtual version of this object. Notice how when light hits a matte or diffuse material surface, the light is bounced back in a scattered pattern, but when it hits a shiny or specular surface, the light comes back to our eyes with direct rays, and often colored with the same tint as the object. To observe this phenomenon in real life, get a handful of coins. As you hold each piece over the light source and look at the highlight, you will notice that the highlight is the same color as the coin. This color influence can be added to the Shininess (specular) layer for any textures you think should exhibit this characteristic by changing the specular color in the Build/Textures/Specular menu. Now look at a piece of shiny fabric like silk or leather. Notice that the highlights are the color of the light emanating from the light source; the object's material is not influencing the color of them at all. Close observation of the patterns and colors of the highlights on real-life objects will help you create specular graphics and choose shader settings in the Build/Edit/Textures menu that are visually engaging and delightful to the eye of the viewer.

11.8 NOTES REGARDING BUMPINESS (NORMAL), SHININESS (SPECULAR), AND ALPHA MODE USAGE

As you begin to create and use Bumpiness (normal) and Shininess (specular) textures on the objects in your virtual environment try to keep these notes in mind:

- Typically the Bumpiness (normal) graphic size is very small, usually 256 pixels by 256 pixels, and is tiled across the surface of an object. A large surface graphic that will not be tiled will need a commensurately large resolution Bumpiness (normal) graphic to correctly display the bumpiness without any undesirable artifacts on your surface.
- As you learn to use these graphic layers, start small. Use a Bumpiness (normal) and Shininess (specular) layer on avatar-sized objects like clothing, weapons, or vehicles first, and then scale up to bigger things.
- Try to include a moving light source in the virtual environment you are building, so Bumpiness (normal) and Shininess (specular) textures you have created are shown off to their best advantage. Remember, the Advanced Lighting Model must be activated under the Preferences/Graphics menu.

- Create your Shininess (specular) graphics with the overall color and sources of lighting in mind.
- Set up a layered template in your 2D graphics program so you can work with the Texture (diffuse), Shininess (specular), and Bumpiness (normal) layers all in one file.
- Understand the lighting model parameters and settings available for the materials in your virtual environment. The visual texture and lighting model of a material is created with the use of bidirectional reflectance distribution functions (BDRF) [1]. These functions are controlled with the settings provided by the client viewer. In Second Life and OpenSim, a control menu for these adjustable settings is provided in the Build/Edit/Textures menu. This menu includes these settings: Diffuse Color, Diffuse Texture, Specular Color, Specular Texture, Specular Exponent ("Glossiness"), Environment Intensity, and Bumpiness (normal).

11.8.1 A Few Notes about Alpha Mode Options in the Virtual Environment

Alpha modes do not have to be a great mystery. Some of these can create quite stunning effects, so you should experiment with them. However, bear in mind these additional layers of information applied to a material will increase the demands on graphics processing. Consider the visual impact necessary and use these alpha modes sparingly.

11.9 SCRIPTING FOR ANIMATED GRAPHIC TEXTURES IN A VIRTUAL ENVIRONMENT

As you probably know, there are many small scripts that can influence the behavior of a Texture (diffuse) graphic on the surface of an object. By using the `llSetTexture` functions, graphics can slide, ping-pong, rotate, and change background color. Another sort of special graphic/texture is the animated version. This multi-image graphic utilizes a special LSL script to generate a moving image on the surface of your object. Like the paper flip (or flick) books you probably had as a kid, this graphic creates the illusion of movement by displaying a sequence of images in rapid succession. These graphics and the scripts that run them are handy for all sorts of things like flickering flames, moving water, flapping wings, and spinning mechanical parts. In the project at the end of this chapter, you will use one on the abutments of a fairy bridge to create an animated detail.

11.9.1 Scripting an Animated Graphic from Still Images

To create a good smooth animation from a series of images on the Texture (diffuse) graphic, you need to decide these three things:

1. The number of columns these sequential images will have on your scripted texture
2. The number of rows these sequential images will have on your animated texture
3. The speed that these frames will be displayed (defined as frames per second in the LSL script)

Figure 11.5 is a sample script for animating a graphic texture. As you can see from the function `llSetTextureAnim`, the script will be calling for this information right away. Now you will have to decide what is more important: smoother animation or more resolution. The largest graphic for objects that you can upload is currently 1024 pixels by 1024 pixels. Setting up for a four-frame animation (2 × 2) cuts the resolution for each image to 512 pixels by 512 pixels. If you need a 16-frame animation (4 × 4), each frame shrinks to 256 pixels by 256 pixels. To maintain acceptable image fidelity with the frame number increasing

```
//This script is a modified version of the graphic animation script that Robin
 (Sojourner) Wood provides at her texture tutorial display and library in Second Life.
//Located at http://slurl.com/secondlife/Livingtree/127/99/25, this is well worth the
 visit if you are interested in learning about graphics and how they are used inworld.
default
{
 state_entry()
 {
 llSetTextureAnim(ANIM_ON | LOOP, ALL_SIDES,4,4,0,0,10.0);
 //ALL_SIDES means to animate all the faces of the object (in this case, all.)
 //4 is the number of columns in the grid (first one in the string)
 //4 is the number of rows (second one in the string)
 //10 is the number of frames per second (last one in the string)
 //make the numbers match the texture grid you set up,
 //Maintain all comma positions or a Syntax Error will appear.
 }
}
```

FIGURE 11.5 A Sample Script for Animating a Graphic/Texture.

and the resolution dropping, you will need to scale down the size of the object that will have this animated graphic on it. In the project at the end of this chapter is a step-by-step guide for creating an animated graphic texture for your game-based sim.

11.9.2 Creating Animated Graphics from GIF Animations and Video Media

Another approach is to convert a Graphics Interchange Format (GIF) file into an animated graphic texture that you can apply to the surface of your object. This can work very well when you create the GIF from a short series of nice clear images or drawings. Remember, the more frames you need to create the loop, the lower in resolution they will be on the surface of the object.

With your GIF file in hand, you can easily convert it into an animated graphic using the handy application that Fred Beckhusen has on his Outworldz website. It is a tiny program (for the Windows OS only) called GIF_2_SL_Animated_Texture_Program, and you can download it from http://www.outworldz.com/Secondlife/Posts/Gif/Create-Gif-in-Second-Life.htm. He also includes an LSL script to run the animated graphic texture on his site: http://www.outworldz.com/cgi/freescripts.plx?ID=67. Both of these are handy tools to keep around for your animated graphic needs. If you want to put a more extensive animated loop onto your surfaces, then consider making a video and applying it as a media graphic. You can also set up any of your graphics (including those used on the terrain) to access that media by using the World/Location Profile/Place Profile/About Land/Media tab and linking the URL to the web-based media you have made. It would be wise to include some sort of text or symbol on that graphic/texture that reminds visitors to make sure the media settings on the top right of the viewer bar are set to “Show all” so that they see the animation video you are bringing into the virtual environment, not just the media graphic. In the next section is a project that brings a Bumpiness (normal) graphic, Shininess (specular) graphic, and animated graphics together into one structure—a fairy bridge for your game-based sim.

11.9.2.1 Tip: How to Play Media from YouTube on the Face of Your Objects

If you desire to display only your YouTube video (without the surrounding webpage) as a media graphic on the face of an object in Second Life and OpenSim, then you will need to change the standard format of the link, which looks like this https://youtu.be/GV1J9QWwVOw, to this format http://www.youtube.com/embed/GV1J9QWwVOw.

11.10 PROJECT: ADDING ADVANCED MATERIALS TO THE FAIRY BRIDGE

In this project you will practice some essential skills with advanced materials. You will learn the process for adding custom Bumpiness (normal) and Shininess (specular) graphic/textures to a fairy bridge provided with the content of this book, and the process for creating an animated graphic/texture that is activated by the LSL script in Figure 11.5.

Before we start, let us review some aspects of the Advanced Materials settings that are available with the client viewer (we used Firestorm 4.7.3 Release ×64 with OpenSimulator support). In Figure 11.6 is a screen grab of the Build/Edit menu, under the Texture tab. As you can see there are two rows of options for you on the menu. On the top row you can change the underlying Color, set the Transparency, and shift the Glow settings. In the lower row, there are two dropdown menus. On the left is one for choosing either a Media texture, or a standard Material texture. On the right side is a three-part menu for choosing channels for your graphic/textures: (1) Texture (diffuse), (2) Bumpiness (normal), and (3) Shininess (specular). Below that is a "drag-and-drop" window for changing the current graphic/texture on the selected faces of the object. As you can see in Figure 11.6,

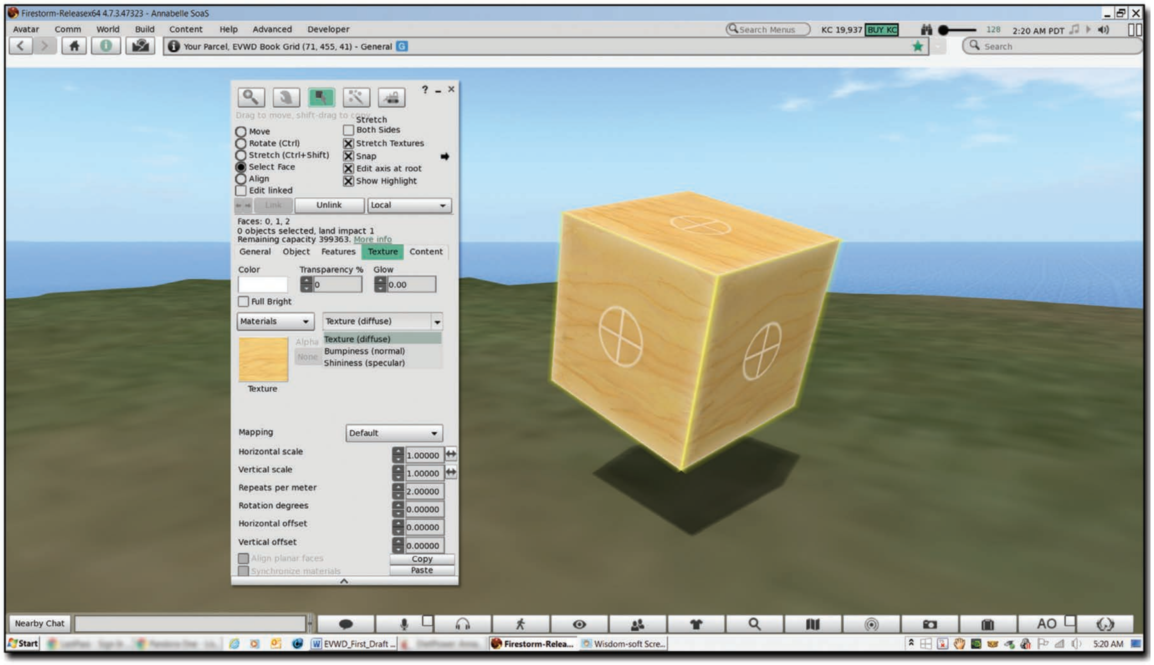


FIGURE 11.6 Basic Texture Edit menu (Firestorm).

by using the Select Face button in the upper section, the cube object on the right has all of its faces selected. The default selection is for all faces, but you can select the faces individually by holding down the Shift key and clicking on the face you want while the Select Face option is chosen. We will utilize this feature to apply the custom Bumpiness (normal) and Shininess (specular) graphics to the fairy bridge in the next section.

Now let us take a quick peek at the Bumpiness (normal) and Shininess (specular) graphic/texture options. These look best when the Advanced Lighting Model is activated under the Avatar/Preferences/Graphics menu, so make sure that is turned on. You may also want to set the light sources to Sun/Moon + Projectors, so you can make a graphic/textures test model yourself. In Figure 11.7, you will see the same cube with a troll's head graphic/textures applied to the three visible sides. Also included in this setup is another primitive object that has been made into a projector light source, using the options under the Features tab in the Build/ Edit menu. (Note: A mostly transparent tapered box with a slight glow is included in the demo model so the "beam" of the projector can be seen for illustration purposes.) Look at the difference in the three selected box faces under midnight and dusky environmental lighting conditions:

- The top face has just a troll's head graphic/texture applied only in the Texture (diffuse) channel, and the Emissive mask has been selected, so the white parts are very bright and the black areas are very dark under these Midnight lighting conditions (top image).
- The left face has the troll's head graphic/texture applied in the Texture (diffuse) channel, and in the Bumpiness (normal) channel, the "suction" bump texture option was chosen. You can see how the curved edges of that bumpiness graphic/texture are catching the environmental lights (middle image).
- The right face of the cube has the troll's head graphic/texture applied in the Shininess (specular) channel only, and you can see it shows most brightly where the projector light is hitting the white sections of the Shininess (specular) graphic/texture, so the troll's head is only partially seen and looks ghostly (bottom image).

Make this little model yourself, and try various graphics on it, including some with transparent alpha channels in them. All sorts of possibilities will open up to your imagination as you play.

OK, now that you have become more familiar with the options on the Build/Edit/Texture menu, let us proceed to adding some custom Bumpiness (specular) and Shininess (specular) graphic/textures to the fairy bridge.

11.10.1 Adding Bumpiness and Shininess Graphic/Textures to the Fairy Bridge

Upload the BLDG_Fae_Bridge model content to your virtual environment. This initial upload should include the mesh model (BLDG_Fae_Bridge.dae) and its physics file (BLDG_Fae_Bridge_Physics.dae), as well as its five Texture (diffuse) graphics, which are attached automatically when the textures are included in the Upload menu. Overall, there are 15 graphic/textures used on the five major face groups of the Fairy (Fae) Bridge. Five of these graphic/textures are for the Texture (diffuse) channel, five are for the Bumpiness (normal) channel, and the remaining five are for the Shininess (specular) channel. Now is a good time to make sure that your Advanced Lighting module is turned on under Avatar/Preferences/Graphics in your client viewer.

Here is a list of the five fairy bridge face groups and their graphic/textures:

- Abutments (foundations of the bridge on each side)
  - BLDG_Fae_Bridge_Abutment.tga (applied on upload of mesh)
  - BLDG_Fae_Bridge_Abutment_Norm.tga
  - BLDG_Fae_Bridge_Abutment_Spec.tga

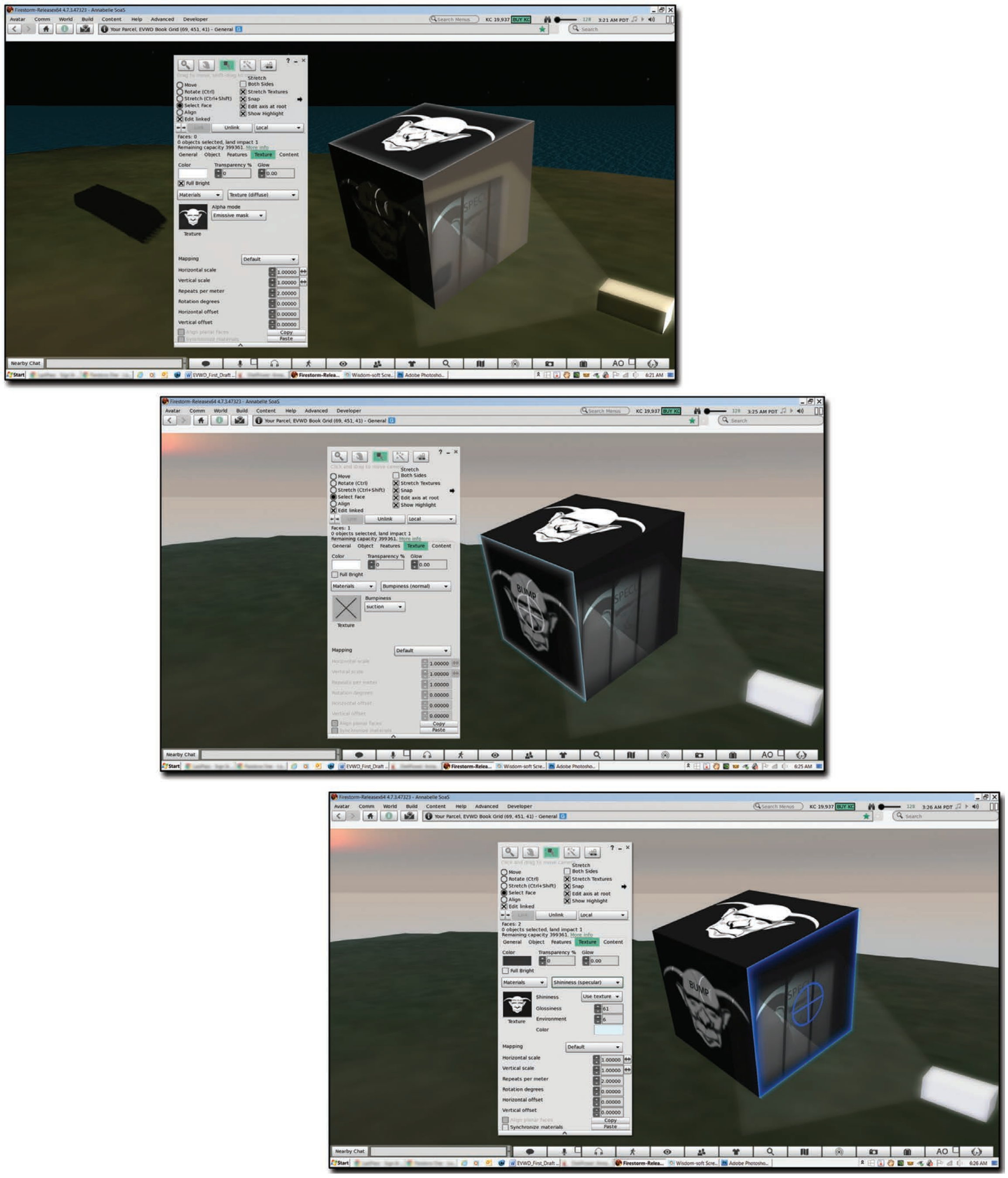


FIGURE 11.7 Advanced Material options, the troll's head variations.

- Cobbles (walkway of the bridge)
  - BLDG_Fae_Bridge_Cobbles.tga (applied on upload of mesh)
  - BLDG_Fae_Bridge_Cobbles_Norm.tga
  - BLDG_Fae_Bridge_Cobbles_Spec.tga
- Deck (top side railing faces)
  - BLDG_Fae_Bridge_Deck.tga (applied on upload of mesh)
  - BLDG_Fae_Bridge_Deck_Norm.tga
  - BLDG_Fae_Bridge_Deck_Spec.tga
- Deck2 (underneath part of the walkway and side faces)
  - BLDG_Fae_Bridge_Deck_2.tga (applied on upload of mesh)
  - BLDG_Fae_Bridge_Deck_2_Norm.tga
  - BLDG_Fae_Bridge_Deck_2_Spec.tga
- Quoins (decorative edges to abutments and sides of bridge)
  - BLDG_Fae_Bridge_Quoins.tga (applied on upload of mesh)
  - BLDG_Fae_Bridge_Quoins_Norm.tga
  - BLDG_Fae_Bridge_Quoins_Spec.tga

Once you have uploaded the fairy bridge, look it over to make sure that all the surfaces have their Texture (diffuse) graphic/textures applied to the mesh model. Got them all? Good! Now, we will focus on the application of the Bumpiness (normal) and Shininess (specular) graphic/textures to the mesh faces. In Figure 11.8, are five views of the fairy (fae) bridge showing the location of each face group on the mesh model. (Note: For a clearer understanding of what each face group comprises, the nonselected face groups have been made semitransparent in each progressive image.) From the top reading clockwise, the images show you the face groups in the following order: abutments, cobbles, deck, deck2, and quoins. To apply the Bumpiness (normal) and Shininess (specular) graphic/textures to each face group, you will select them by using the Select Face option at the top of the Build/Edit/Texture tab menu. By holding down the Shift key, you can select the face groups individually. Then by using the dropdown menus (shown in Figure 11.7) you will apply the respective Bumpiness and Shininess textures that you uploaded separately (Avatar/Image/Upload). The best approach is to apply all of these textures in the order shown in Figure 11.8, with their default settings in the Build/Edit/Texture menu. Then you can adjust the settings for the cobbles, deck, and quoins face groups to the levels of Bumpiness and Shininess that you prefer for your virtual environment. These are the most visible elements of the fairy bridge and would show off their advanced materials to greatest advantage. The remaining parts of the bridge should match settings in Bumpiness, but you may want to dial down the Shininess somewhat to provide a more interesting contrast overall.

OK, now that you have become an advanced materials ninja, let us review the process for developing a graphic/texture that will be animated with scripting.

11.10.2 Making an Animated Graphic/Texture: Step by Step

11.10.2.1 Step 1: Setting Up the Story Your Animation Will Tell

An animated graphic/texture can provide you with the mechanism to create moving parts like turning wheels and flickering flames. It can also display a sequence of events. You can also use this approach to tell a short story with your graphic. For this project, we will go through a methodology for creating an animated graphic that displays a metamorphosis of a Troll's face into a Green Man (a pagan deity who represents plants and the

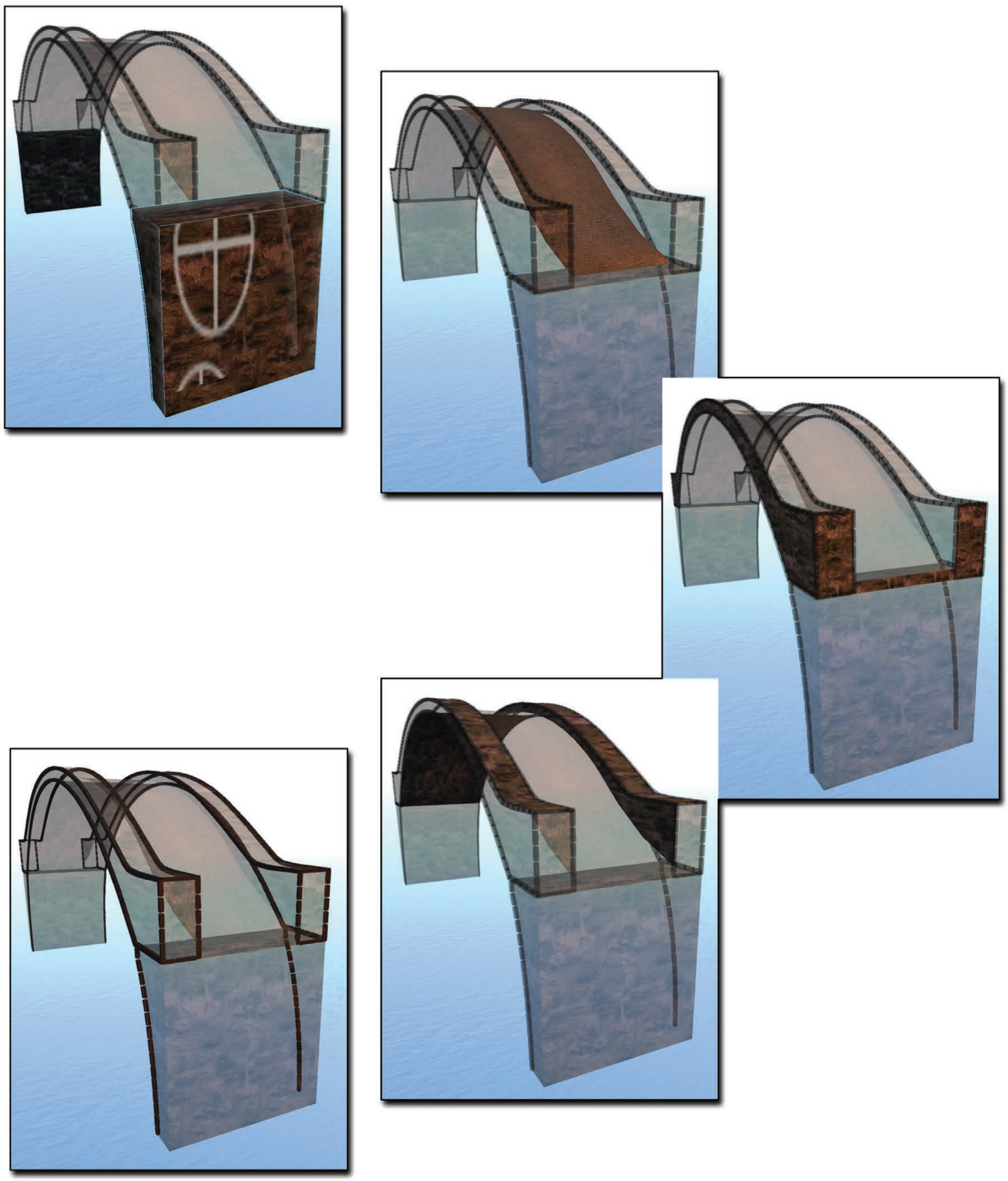


FIGURE 11.8 Applying Bumpiness and Shininess graphic/textures to the fairy bridge.

life of the world). If you want to start with something simpler, then you can tell the story of a cube turning into a sphere, or the growth of some bars on a chart.

11.10.2.2 Step 2: Plan for the Number of Key Frames Your Animation Will Need

A *key frame* is one image that represents a major change, such as the starting and ending positions of a character in the action and plot of an animation. For instance, in the story of a troll turning into a Green Man, the most important frames, the key frames, are the images of the troll's head before he changes at all, and the image of his head after the metamorphosis is complete and he has become a Green Man. Metaphorically speaking, key frames are like pins on a map, and the rest of the frames in the animated graphic show the process of going from one pin to another across the map. In this animation, we want to show a return from Green Man back to troll, so there will be three key frames:

Key frame 1—Just the troll's head.
Key frame 2—The troll's head after he has transitioned into a Green Man, growing branches and leaves from his head.
Key frame 3—Autumn and winter have come, the leaves and branches around the Green Man's head have died, snow has fallen, and we are just about back to the first key frame with a troll's head.

Key frame 3 should be equal in its "visual difference" from key frame 1, just as key frame 2 is equal in its visual difference from key frame 3, so that there is a smooth transition from troll to Green Man and back to troll overall as the animation plays. Take the time to write down a little description for each one of your frames, starting with the key frames, so that you know what visual differences each image will need to complete the animation. The troll-to-Green Man animation graphic/texture is going to be made of 16 frames. These will be laid out in your graphics program and scaled down into one graphic that is 1024 pixels by 1024 pixels. Each frame will tell 1/16 of the story. If you are doing a simpler animation, then you can use four or eight frames, and the story would be told using frames that display a much larger visual difference or tell a simpler story as they progress through the sequence. When an animation graphic script such as the one in Figure 11.5 is included in the content of your primitive and running inworld, each frame area of the animated graphic/texture (comprised of 4, 8, 16, etc. frames) is read sequentially and displayed on the face it has been applied to, going left to right and top to bottom, as shown next:

1-2-3-4
5-6-7-**8**
9-10-11-12
13-14-**15**-16

The potential key frames for a 16-frame animation are shown in the bold font.

11.10.2.3 Step 3: Creating the Images for the Animation—The Process

There is an overview of this process in Figure 11.9, which we will refer to throughout this project. In Figure 11.9, you can see the process used for creating the individual frames of the troll-to-Green Man animation graphic/texture. Working from real-life research (photo in the top left of Figure 11.9), a series of pencil sketches were generated, digitized, and brought into a 3D Modeler (3ds Max) as guidelines for building the branches that would grow out of the troll's face (middle of Figure 11.9). Layer by layer, using the line guides from the sketches, a series of 2D branchy shapes were created around the troll's head

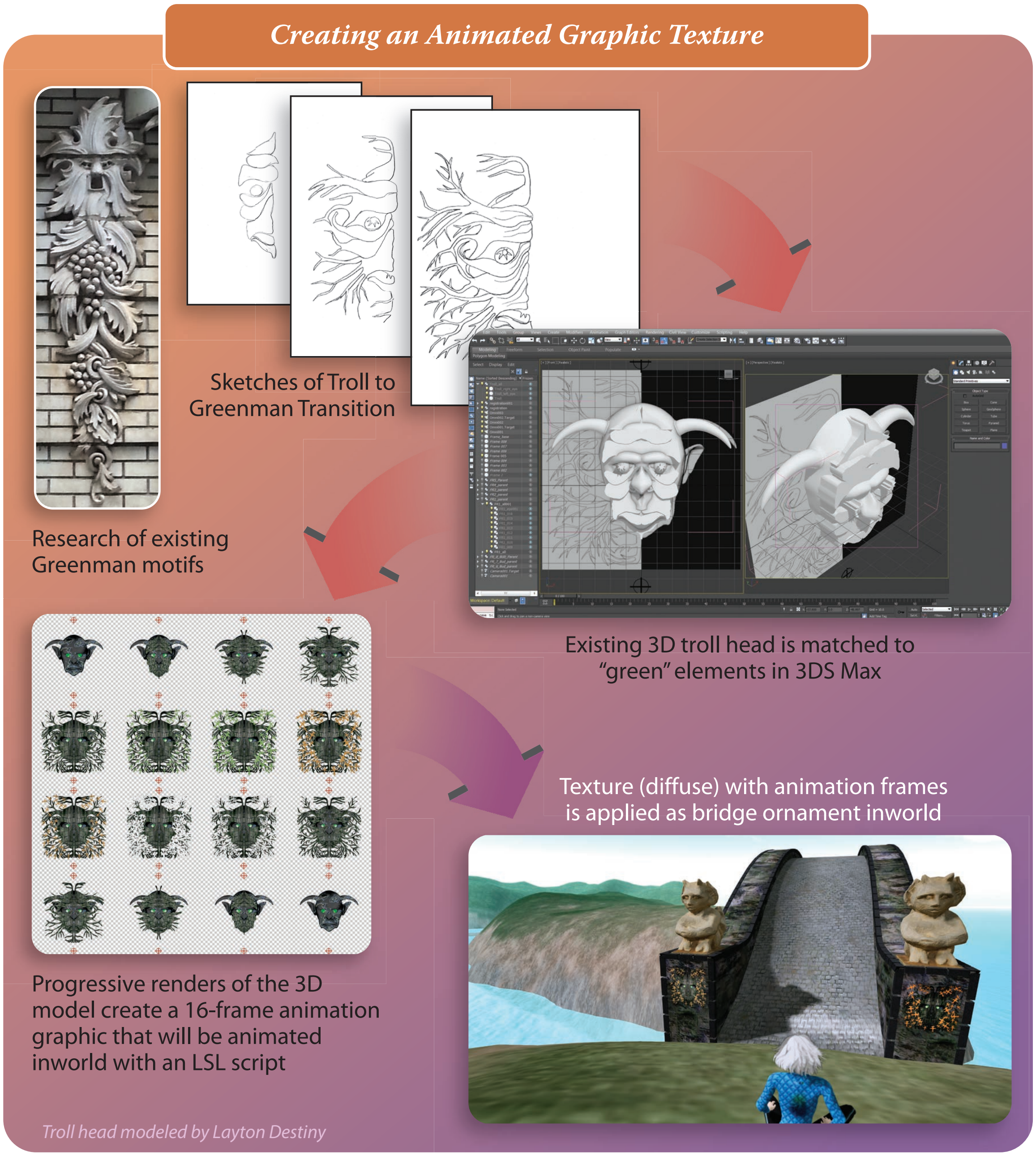

Creating an Animated Graphic Texture
Research of existing Greenman motifs
Sketches of Troll to Greenman Transition
Existing 3D troll head is matched to "green" elements in 3DS Max
Progressive renders of the 3D model create a 16-frame animation graphic that will be animated inworld with an LSL script
Texture (diffuse) with animation frames is applied as bridge ornament inworld
Troll head modeled by Layton Destiny

FIGURE 11.9 Process for making a troll-to-Green Man animated graphic/texture.

model and extruded to give them depth. These layers were given a procedural bark graphic/texture and a leaf graphic/texture made from photos taken of real oak leaves. These oak leaf graphics were changed in scale and color so that they appear to transition from spring leaf buds to summer leaves to autumn foliage in the animation. There are even a couple of frames that have snowflakes in them to show a winter look on the Green Man's branches. Once the layered sequence of forms was completed, these models were lit, and each frame of the progression was rendered as a name_of_animation_frame#.png file (at 1200 pixels by 1200 pixels each) with a transparent background and 3D registration marks. Overall our goal is to get a stable, unmoving troll's head showing the progression of branches growing out of it, as they enlarge, change seasons, and then disappear. As you can see, registration marks (circle and crosshairs) were added in the 3D model and rendered into the frame images to help with the registration of frames in the Photoshop image (middle of Figure 11.9). Then these frames were laid out in sequence to create the graphic/texture layout discussed in the next section (bottom left of Figure 11.9). Note: It is not necessary to go to such an extensive build for your first animated graphic/texture. To try this as a simple project, use an object inworld against a white background. Move and modify it to make your animation frames from snapshots saved to your disk.

11.10.2.4 Step 4: Layout the Animation Frames on the Graphic in Photoshop

All the animation.png frames were layered in sequence into one "Super scale" (4800 pixel by 4800 pixel) file in a graphics program (Photoshop), as shown at the bottom left of Figure 11.9. It is very useful to set up a grid in your graphics program and snap to it while you are doing this step, because the registration of the sequential images has to be perfect. Perfect alignment prevents a "jump" in the image while the animation plays. Once the frames are all laid out and aligned perfectly, you can flatten the image into one layer with a transparent background, and reduce the overall size to 1024 pixels by 1024 pixels.

To reiterate the guidelines for graphic/texture efficiency inworld:

- The largest graphic/texture for objects that you can upload is currently 1024 pixels by 1024 pixels.
- A four-frame animated graphic/texture (2 × 2) cuts the resolution for that size image to 512 pixels by 512 pixels. If you need a 16-frame animation (4 × 4), each frame shrinks to 256 pixels by 256 pixels.

To maintain acceptable image fidelity with the frame number increasing and the resolution dropping, you will need to carefully consider the scale of the object that will have your animated graphic/texture on it and the average distance it will be viewed from. For example, an eight-frame animation graphic of flickering bonfire flames would probably look good on a box primitive with these dimensions: x = 1 meter, y = 0.05, meter and z = 1 meter. Three of these wide and thin boxes can be located at the same x,y coordinates and set with different rotation on the z-axis (a common way of faking a 3D object before meshes came along). The addition of some particle scripts to make sparks and drifting smoke to the object would add to the illusion. For more intricate animations, with many smaller moving elements, you will probably want to have more frames on the graphic/texture and apply it to a smaller object (primitive) inworld. The animated troll-to-Green Man animated graphic/texture will be 16 frames and should not be applied to any surface larger than 0.75 meters to the side.

11.10.3 Scripting and Applying the Animated Graphic Inworld

OK, now that you have made the graphic for use with the graphic animation script, it is time to test it inworld.

11.10.3.1 Step 5: Upload the Graphic—First Test on a Primitive

Using the Avatar/Upload/Image menu bring the graphic/texture into your avatar's inventory. Make a box primitive, change five of its sides to solid black, and leave the front face with the default "plywood" texture. Now create a new script in its contents and copy/paste code from Figure 11.5 into that new script. The default wood graphic/texture on the primitive should start moving once the script is saved. Then apply the animated graphic/texture you made to the front face of the box primitive and rejoice at your moving animation on the box face.

11.10.3.2 Step 6: Tuning the Animated Graphic and Incorporating It into a Build

Under the Build/Edit/Texture tab, set the Texture (diffuse) channel to Alpha blending, so that the background is transparent, set the Bumpiness (normal) channel to Brightness, and the Shininess (specular) channel to use the same graphic that the diffuse channel is using. Tweak the animation scripting as necessary for the speed and number of faces that your animated graphic/texture will be displaying during its animation. Reduce the scale of the box, and place some copies of it on the fairy bridge as magical moving ornamentations, as shown in Figure 11.10.

11.11 CHAPTER SUMMARY AND FINAL THOUGHTS

When you are making graphic/textures for a virtual environment, thinking superficially is a good thing. With the right combination of Texture (diffuse), Bumpiness (normal), and Shininess (specular) graphic/textures, a very simple mesh will have a rich, visually complex look without making big demands on your

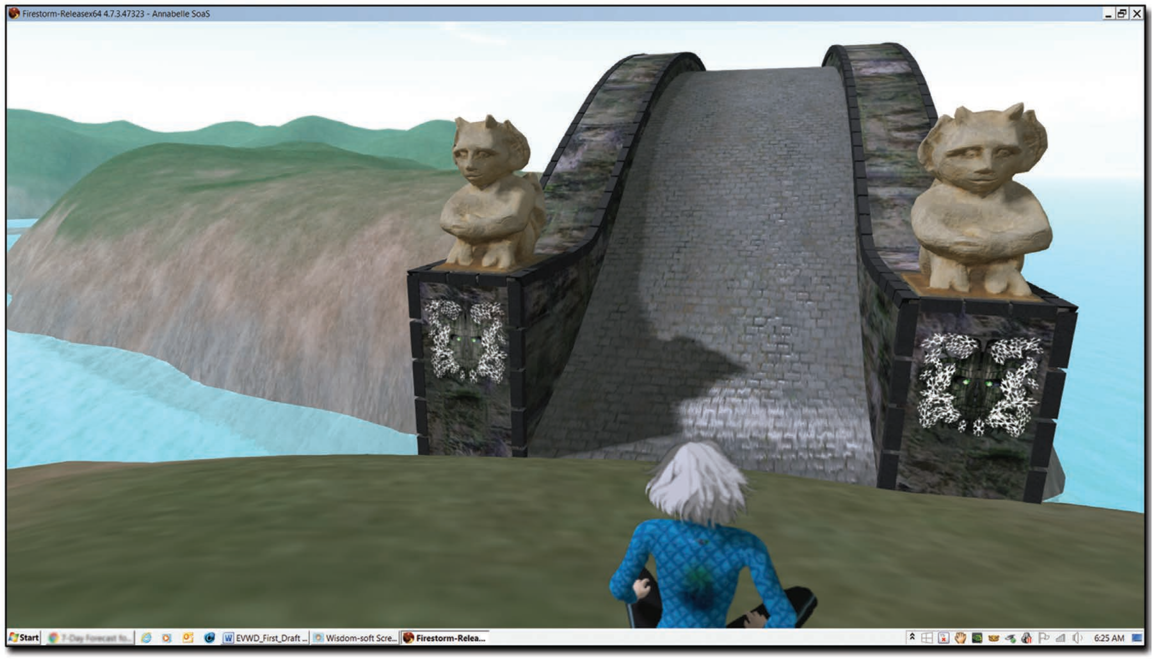


FIGURE 11.10 Final look of troll-to-Green Man animated graphic/texture on bridge.

computer's graphics processor. Make many observations of objects in real life and take pictures of interesting surfaces you see to build up your visual vocabulary and stock image collection. Doing that will greatly increase your design capacity and lower the actual creation time, because you will know just what kind of surface and highlighting you want to achieve, or at least have a ready reference to it. There are many 2D and 3D programs, plugins for them, and online sites to utilize in the creation of graphic/textures and advanced materials. You will find a rather extensive list at http://wiki.secondlife.com/w/index.php?title=Texture_Tools&oldid=1187720. Keep your first projects modest in scope and scale, so that you can rapidly change the settings and experiment with all sorts of combinations. Eventually, you will settle on some favorites that can be used throughout each virtual environment to support your overall artistic vision.

REFERENCE

1. "Second Life's Light Model for Materials," *Second Life Wiki*, accessed February 11, 2015, http://wiki.secondlife.com/w/index.php?title=Second_Life%27s_light_model_for_materials&oldid=1179347.

12 World Building and Design with Procedural Techniques

Now I held in my hands a vast methodical fragment of an unknown planet's entire history, with its architecture and its playing cards, with the dread of its mythologies and the murmur of its languages, with its emperors and its seas, with its minerals and its birds and its fish, with its algebra and its fire, with its theological and metaphysical controversy. And all of it articulated, coherent, with no visible doctrinal intent or tone of parody.

Jorge Luis Borges, "Tlön, Uqbar, Orbis Tertius"

12.1 OVERVIEW OF WORLD BUILDING AND DESIGN WITH PROCEDURAL TECHNIQUES

One of the biggest challenges for a designer is to create a virtual world that has visible and audible content that we believe in, and that we understand because it resonates with our real-world experience. One way to visualize a new world while you are in the process of creating it is to make a "Species of the World" model. In this model, you divide what the world will contain into these five parts: "skin," "bones," "mind," "heart," and "soul." Each of these parts represents a large category of world content. For instance, as you will see in Table 12.1, the skin category concerns itself with the terrain and its many aspects and characteristics, and the soul category contains religions and art forms.

If you are thinking of designing a completely new virtual world, creating a "Species of the World" chart is one place to start your list of elements to design or set rules for. Of course, this is a general overview designed to get the broad strokes in place; there is much more to each world you design, and infinite levels of detail that can be added. In the *Dungeon Masters Guide* to *Dungeons & Dragons* (*D&D*), a role-play adventure game that started in the mid-1970s, you will find a useful and comprehensive guide to creating an imaginary world [1]. *D&D* (http://www.ddo.com/en) is still a worldwide phenomenon, and its collaborative open-ended story structure provides an excellent foundation for developing your world-building and storytelling skills. Here are two possible approaches to designing a world for a game-based sim: (1) create just the amount of content and rules necessary for the regional area of the world in which you will be playing the game, or (2) create the entire world and its rules first, and then subdivide it into the areas you will use to play the game and develop those individually. If you are building a multiregion environment for a large grid in a virtual world, you may want to approach it from the second method, and if you are just designing a single region, you may find that the first approach works well.

In this chapter, we will explore ideas for world building and how that relates to procedural techniques. Here are the key concepts:

- World building relates to the creation myth.
- Procedural techniques available in modeling and graphics programs can be used for creating game-based environments.
- Procedural scripts can be utilized in building content within game-based virtual environments.

TABLE 12.1
The 5 Parts in the "Species of the World" and Some of What They Contain

"Body of the World"	"Skin"	"Bones"	"Mind"	"Heart"	"Soul"
Items in that category	Terrain	Forces and motion (tides/earthquake)	Language	Music	Religions
	Mountains	Gravity	Mathematics	Storytelling	Painting, illustration
	Steppes	Planet geology (core/mantle/layers)	Magic	Literature	Sculpture
	Desert	Atmosphere	Industry	Crafts (textiles/ ceramic/metalwork)	Filmmaking
	Oceans	Planetary system	Media	Foods	Writing
	Lakes	Planetary movement and time measurement	Science (chemistry/alchemy)	Philosophy	Supernatural
	Rivers	Biosphere	Architecture	Dance	Poetry
	Geopolitical boundaries	Ecology	Psychology	War	Other planes of existence

At the end of this chapter you will "grow" a fairy tree by utilizing a procedural program based in the Linden Scripting Language (LSL) code. This content is part of the game-based virtual environment you will construct throughout the various projects included in this book.

12.2 WORLD BUILDING AND THE CREATION MYTH

The foundation of a believable world is a strong creation myth. Let us take a look at a few of them. Cosmological stories such as those found in Norse mythology about the enormous ash tree called Yggdrasil, which connects the nine worlds, is one such example of how the world's structure is explained by storytelling [2]. The Native American creation mythology called the Diné Bahane' (Navaho Creation Story) also speaks of multiple worlds, starting with the creation of one people, the Air-Spirit People and their relationship to the four primary colors (yellow, blue, black, and white) and the four cardinal directions (north, east, west, and south).

> It is said that at Tó bił dahisk'id white arose in the east and was considered day. We now call that spot Place Where the Waters Crossed. Blue arose in the south. It too was considered day. So the Niłch'i dine'é, who already lived there, moved around. … In the west yellow arose and showed that evening had come. Then in the north black arose. So the Air-Spirit People lay down and slept. [3]

In her book, *Primal Myths-Creation Myths around the World*, Barbara Sproul describes one of the Chinese creation myths.

> IN THE BEGINNING there was chaos. Out of it came pure light and built the sky. The heavy dimness, however, moved and formed the earth from itself. Sky and earth brought forth the ten thousand creations, the beginning, having growth and increase, and all of them take the sky and earth as their mode. The roots of Yang and Yin—the male and female principle—also began in sky and earth. [4]

Try the following mini project. It will jump-start your own creation myth and get creative thoughts flowing for your game-based sim world.

12.2.1 Mini-Game Learning Block: Developing Your Own Creation Myth

1. Pick your favorite number from the range 1 to 12. (For instance, number 9.)
2. Now pick your favorite flora species and favorite fauna species. (For instance, willow tree and butterfly.)
3. Divide your favorite number by 3, round up to the nearest whole number, and make this the number of genders your world has. (For instance, 9 would become 3, and the world would have three genders.)
4. Write about starting with the universal void and then instantiate the seed of your favorite flora and let it grow into a structure that supports the emergence of the fauna you chose. (For instance, you might write: "Into the void, a single willow seed fell by the banks of the great river of stars, and that seed grew into a graceful humble tree that felt the loneliness of the void most acutely. After 9 million stars had been born, a small caterpillar drifted across the void and landed on the branch of the willow tree and wrapped itself into a cocoon. Slowly a new creature began to form inside the cocoon, and when the tree had almost given up waiting for a result, a small starry butterfly emerged. As it flapped its new wings, the void began to change …")
5. Develop the "initial place" for the world to start. It need not be the final world, but it should be connected in a developmental process to that result. (For instance, you might write: "As the butterfly flapped, it danced on the branches of the willow tree, and bits of bark fell off to form land and mountains, and as the willow fronds touching the river swayed in time to the dance, drops washed over the new land and made oceans and rivers.")
6. Develop the first "representatives" of your species from this action. (For instance, you might say: "Each flap of the butterfly's wings gathered stardust and formed the shape of the first person [gender 1], and the second person [gender 2], and the middle person [gender 3].")
7. Develop a "family history" about how these first people and their offspring populated the new world, and how they created a god (or gods) that became the basis for the religious stories in their world.

If you want to continue building your world, then follow these steps:

- Relate your favorite number to each category in Table 12.1. (For instance, if you picked 9 as your number, the religion in your world has 9 gods, the architecture uses a standard measure based on 9, the music uses a 9 tone scale, and so forth.)
- Imagine how people with these qualities would develop tools, and use them to build their civilizations. (For instance, they often think of tools that have additional parts to function, just as they need three genders to reproduce.)
- Imagine how these people would develop their societies. (For instance, there is a separate caste for the middle gender.)
- Imagine how they interact with the environment. (For instance, they learn early on to manipulate genetics, and from that knowledge can develop custom flora and fauna that serves their purpose.)
- Continue until you have defined the original content from Table 12.1 in terms of your creation myth.

As you no doubt discovered from that mini-game, world creation myths can be spun from just about anything. They outline the "cultural DNA" of the world you are developing for your game-based sim. Worlds are complex, and building them is a challenge. Starting with a creation myth will give you, the designer, a place to "hang your creative hat," an anchor or touchpoint for all the rest of the design ideas that develop when you create a world. In many cases, the world that is built for the game-based sim or virtual environment echoes our own. As a virtual environment designer, finding and exploiting the differences between the newly created fictitious world and the real, familiar world will help you create a powerful experience for the player. For example, suppose you have a preindustrial agrarian world named Botanica. How do the inhabitants of Botanica cope with a whole new set of cultural and political possibilities when a race of giants rises up from the ground one day? Do these giants become gods? Do they become enslaved? Do they coexist with the population? With this approach the designer can bend the familiar framework of humanity's common historical experience, weaving in mythological or paranormal characters to provide the opportunity for a fictional deviation from the "real-world" experience. In Appendix B, there is one example of how this can be accomplished. It is the journal of Dr. Aubrey Wynn. This document was created as the initial world story/creation myth for Inland: Search for the Sy [5]. This kind of memoir/journal/diary device is often used to help the player make a connection between the world they are leaving and the one they are entering as they start to play the game.

12.3 OVERVIEW OF PROCEDURAL TECHNIQUES AND DESIGN

Building content for virtual worlds is hard. Once game developers decided to leave abstractions like texts, dice, and paper maps for world representation behind and started to create 3D virtual environments and game spaces, the work of world building became an incredibly complex and challenging process. Making and displaying all the 3D "stuff" in a world, for example, the detailed environments that help complete the sense of immersion, constantly challenges the designer, and the graphics processing systems in our computers. This has been so from the very early days of video games; going all the way back to the very first role-play games. One option that game developers have is to incorporate procedural systems, or special programs that automate the building process of game content. Although they have not been used consistently throughout video gaming history, procedural systems have been used since the inception of computer-based games to generate game world content. In 1979–1980, a teenager named Richard Garriot created the first computer role-play game called *Akalabeth: World of Doom* (available for a free download at Gog.com, http://www.gog.com/game/akalabeth_world_of_doom). It is generally agreed that *Akalabeth* was the first game to use a procedural technique to generate its game environment [6]. Shortly after that, a space exploration game called *Elite*, created by Ian Bell and David Braben (now available as *Elite Dangerous*; http://www.elitedangerous.com/), was written. This game utilized numbers generated from a Fibonacci sequence to create the contents of eight galaxies [7]. In today's gaming culture we play several sophisticated games that utilize procedurally generated content. Some notable examples are *Minecraft* (https://minecraft.net/), developed by Markus "Notch" Persson in Sweden, and *Dwarf Fortress* (http://www.bay12games.com/dwarves/), created by Tarn and Zach Adams of Bay 12 games. Soon-to-be-released *No Man's Sky* from Hello Games will be entirely generated with procedural techniques. Founder of Hello Games, Sean Murray, says:

> It's actually being generated as I fly around; none of this exists on a disk. None of it exists on the machine, on your PS4 (Play Station 4) or whatever. None of it exists in the cloud. It's just generating from a set of maths functions. It's always generating the same way because of its maths. Like any maths formula you take the same input and put it into that maths formula, and you'll get the same output. And for our game, the formula is really complex; well it's a series of really simple formulas actually, laid on top of each other to create something quite complex, which is the universe…The input is your position. So you give it any position in that universe, and it will create everything that you see around you, that's the output. [8]

The development of a procedural approach to the virtual environment is developing at a rapid pace in the gaming community. The occurrence of emergent behaviors coming out of the game programming itself presents some exciting possibilities, as Zach Tarn discovered when the carp in *Dwarf Fortress* became carnivorous [9].

12.3.1 The Key Aspects of a Procedural Technique

In *Texturing & Modeling: A Procedural Approach*, David S. Ebert describes a procedural technique. He says: "Procedural techniques are code segments or algorithms that specify some characteristic of a computer-generated model or effect" [10]. He goes on to describe four features of procedural techniques, listing them as abstraction, parametric control, database amplification, and flexibility [11]. To paraphrase his descriptions, these terms can be defined as such:

Abstraction—The complex details of a scene are stored as a function or algorithm, thereby saving storage space on the computer.

Parametric control—A numerical parameter can be assigned to a meaningful concept, allowing the designer to modify the qualities of the object by changing the parameters. (For instance, the roughness or smoothness of a mountain range can be altered with a numerical change, sometimes represented with a "slider or spinner" interface.)

Database amplification—Related to parametric control, this allows for the designer to make profound changes by the manipulation of a few small factors.

Flexibility—The essence of an object, a phenomenon, or motion can be changed without constraint by the laws of physics. (For instance, the designer could create a realistic upward moving rainfall.)

If you utilize a 3D modeler like 3ds Max, odds are that you are already using the procedural techniques built into that program. 3ds Max contains procedurally generated materials menus, "built-in" materials such as marbles or wood grains that allow you to select the pattern of marble stone veins or the wood grain repeats and direction via parametric controls. Obviously, these kinds of tools can be extremely useful to a designer, in terms of saving laborious repetitive work, and allowing for quick prototyping and planning. Let us dig a bit deeper to see what kinds of procedural techniques can be employed in the creation of a game-based sim.

12.3.2 Procedural Theories and Techniques Fundamental to Building a Game-Based Sim

The process of creating a virtual environment with procedural programs can be broken into three general categories: terrain, textures, and 3D objects. Each of these procedural techniques utilizes special random number or pattern generators, which will be discussed next.

12.3.2.1 Common Mechanisms of Procedural Techniques for Terrain or Environmental Levels

Procedural programs used for generating terrain height maps or environmental cells, such as the rooms in a dungeon, may have these mechanisms inside them:

- They rely on algorithms based in *fractal* systems to generate "randomness" of height and transitions from various elevations (simulating erosion) across the terrain [12].
- This randomness may also be generated from *Perlin noise* [13].
- The method for defining the spatial relationships between points on the surface of a terrain and the relative altitudes of those points may be based on a *Voronoi diagram* [14].

Examples of these are displayed in the left column of Figure 12.1.

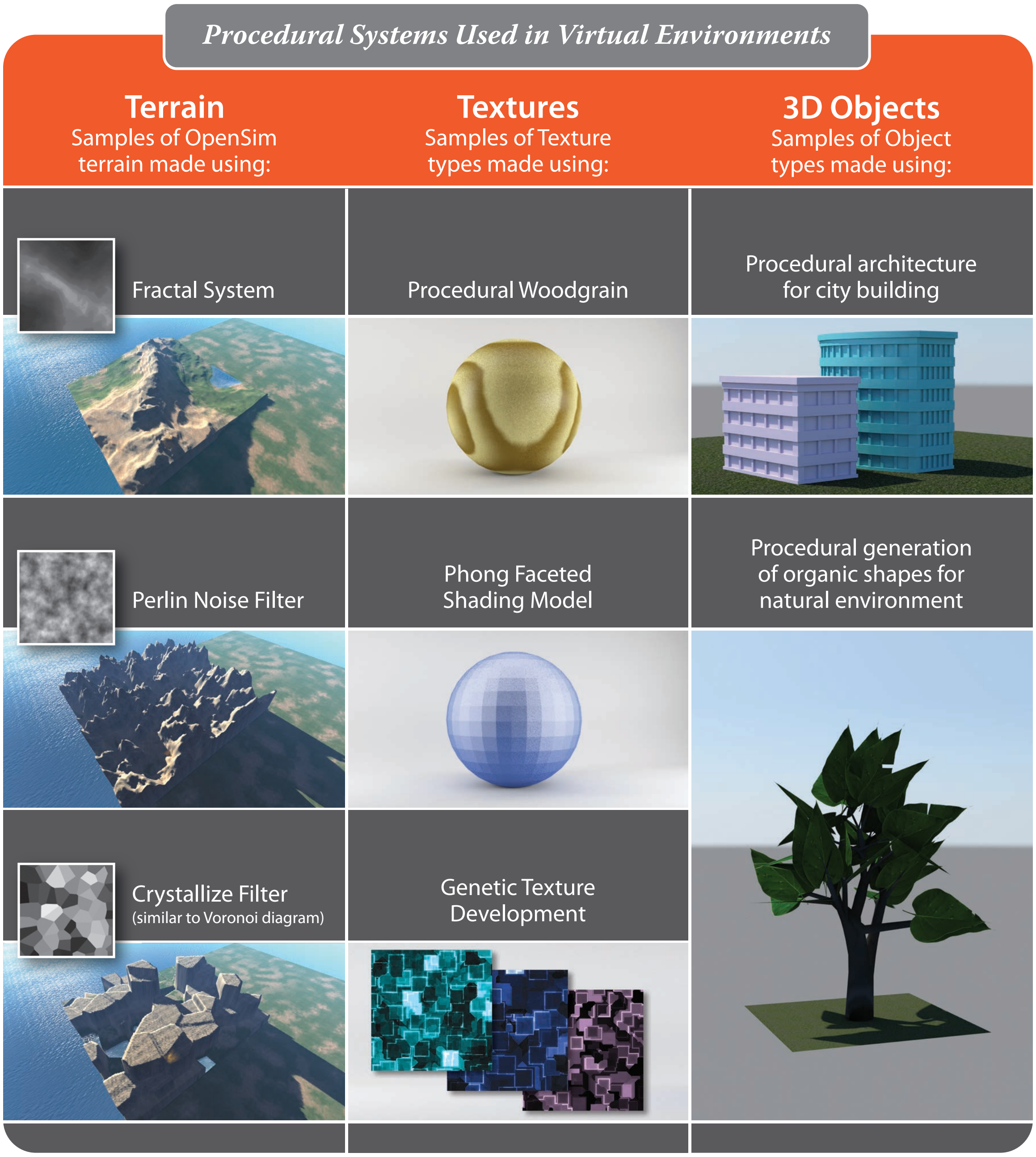

Procedural Systems Used in Virtual Environments
Terrain
Samples of OpenSim terrain made using:
Textures
Samples of Texture types made using:
3D Objects
Samples of Object types made using:
Fractal System
Procedural Woodgrain
Procedural architecture for city building
Perlin Noise Filter
Phong Faceted Shading Model
Procedural generation of organic shapes for natural environment
Crystallize Filter
(similar to Voronoi diagram)
Genetic Texture Development

FIGURE 12.1 Examples of procedural techniques used in terrain, graphic/textures, and 3D objects.

12.3.2.2 Common Mechanisms of Procedural Graphic/Texture Generators

Programs used for generating procedural graphics/textures may have these mechanisms in them:

- Built-in editors can be used to create the randomness on a procedural material, such as a wood grain or marble vein pattern.
- Shading and highlights on the surface of an object can be generated by a procedural texture program called a *shading model* [15].
- They can generate "families" of related textures, related to the initial one by using genetic programming or a *genetic algorithm* [16].

Examples of these are displayed in the middle column of Figure 12.1.

12.3.2.3 Common Mechanisms of Procedural 3D Geometry Generators

- They may use *shape grammars* to spawn the generation shapes and forms that create complexities of architecture and the arrangement of it in a city [17].
- They may use *Lindenmayer systems* (also known as L-systems) for generation of realistic trees and other kinds of vegetation [18].
- In some middleware, premade art or an image is combined with procedurally generated forms to speed up the creation process by reducing the overall calculations. An example would be a tree-generating program that combines its procedurally generated trunk and main tree limbs with images of leafy branches, rather than generating each twig and leaf procedurally [19].

Examples of these are shown in the right column of Figure 12.1.

12.3.3 Everyday Examples of How You Use Procedural Techniques

You may have encountered some unfamiliar terms in the preceding section. The terms *fractal*, *Perlin noise*, *Voronoi diagram*, and *L-system* are not typically in a designer's vocabulary, but they are important to know about. These are names of maths that are used by procedural systems. They are often used within the code of 2D graphics and 3D modeling programs to help us to make unique terrains, graphics, and geometric structures. Let us look at some examples. As you can see in the left column of Figure 12.1, there is a sample of a terrain heightmap created with a fractal generator, and its resulting terrain when loaded into an OpenSim region. Below that is a swatch of Perlin noise, generated by the Filter Forge plugin for Photoshop (https://www.filterforge.com/filters/231.html) and the terrain that results when it is used as a height map. At the bottom of the left-hand column of Figure 12.1, the Adobe Photoshop Filter/Pixelate/Crystalize modifier was added to the Perlin noise image. This crystalize effect looks very similar to a Voronoi diagram [20]. Next to this, is a screenshot of how the crystalized image looks when used as a terrain generating height map in OpenSim. Now, let us consider everyday examples of procedural techniques that we use in creating graphics/textures. GNU Image Manipulation Program (GIMP; http://www.gimp.org/) and Adobe Photoshop both work with plugins that provide many ways to create graphics/textures and patterns. GIMP offers a registry of plugins that work with it (http://registry.gimp.org/) to perform a variety of procedural effects on graphics/textures. Procedural techniques utilize a variety of algorithms to generate their results, for instance, adding code that creates "turbulence" and "displacement" in two colored bands will procedurally generate a graphics/texture with marble veining effects [21]. Another kind of procedural technique produces the subtle shading and highlights on the surfaces of an object. This is called a shading model. There are many examples of this procedural technique across the spectrum of 2D graphics and 3D geometry software.

In the middle column, second image down of Figure 12.1 is an example of the basic “Phong” shader used in 3ds Max, showing how the look of the surface changes as the *diffuse* (the overall base qualities of the surface) and *specular* (the quality of the highlight on the surface) values are displayed on a faceted surface. In the middle column, bottom image of Figure 12.1 are some samples of “genetic graphics/textures” or graphics/texture images created as progressive variants of the base image, with the Filter Forge plugin called “Cyberglow” by Kochubey for Photoshop. This kind of genetic programming can generate “families” of graphics/textures, each successive one developing individual qualities that are related to the initial seed image. Karl Sims (http://karlsims.com/), one of the pioneers in this kind of image creation, has been creating images using genetic programing since the mid-1990s [22]. In the right-hand column of Figure 12.1 there are images showing the output of various kinds of procedural techniques used for the creation of 3D geometry. In the top section there is a screengrab from the 3ds Max modeling software running a script called “Fast Architecture” by Christophe Blattmann (http://www.christopheblattmann.com/). Scripts like this and city creators such as “City Engine” (http://www.esri.com/) have been utilized by many industries for the development of city models, including video game development and level design [23]. At the most fundamental levels, this kind of program that procedurally generates the streets and buildings of an urban environment is using shape grammars to create the 3D forms of the buildings and their arrangement in space. In 1972 George Stiny and James Gips submitted a paper to the International Federation for Information Processing (IFIP) Congress about a new approach to the analysis and design in the visual arts [24]. From that paper about the study of shapes, their arrangement, and the rules (grammars) that can define those shapes and arrangements, the science of shape grammars grew. The successive work of programmers, designers, and architects has developed into the sophisticated procedurally based programs we use today. 3D procedural programs may also use L-systems for the realistic generation of branching structures like trees and other kinds of vegetation. This kind of branch structure defining system was introduced in 1968 by Aristid Lindenmayer (1925–1989), a Hungarian biologist [25]. The bottom image in the right-hand column of Figure 12.1 shows a plant/tree model that was generated by Snappy Tree (http://www.snappytree.com) and modified in 3ds Max. This is an example of the “Art + Procedural” approach to creating procedural content, combining the procedurally generated tree trunk and branching planes with an image that creates the finer details (leaves).

12.4 PROCEDURAL TECHNIQUES USED IN GAME-BASED VIRTUAL ENVIRONMENTS

In this section we will explore some of the techniques and resources you can currently access for building procedural content in virtual worlds.

12.4.1 Building with Fractals in Second Life

Back in 2005, a Second Life resident named Seifert Surface (Henry Segerman in real life; http://www.segerman.org/2ndlife.html) wrote a three-part LSL script that would procedurally generate a series of related forms. This fractal creator, perhaps the first procedural mesh generator in a virtual world, was used by Seifert to create many kinds of sculptures for the science/art scene of early Second Life. The original LSL script can be downloaded from the free LSL section of Outworldz.com for experimentation: http://www.outworldz.com/cgi/freescripts.plx?ID=336.

12.4.2 Non-Player Characters (NPCs) and Clones as Procedural Content

Another way to generate content procedurally is to make clones of your avatar. These individuals can occupy your game-based sim as a non-player character (NPC) to add more “life” and presence to the environment. Typically these avatars are made with cloning scripts in OpenSim (http://opensimulator.org/wiki/NPC) or created as “bots” or scripted agents in Second Life (http://wiki.secondlife.com/wiki/Bot). NPCs based on nonhuman models can be dynamically moved with the Pathfinding functions in Second Life (http://wiki.secondlife.com/wiki/Pathfinding_in_Second_Life) to crawl or fly around your sim. To leverage a procedural approach for making NPCs in your game-based OpenSim region, you would modify your avatar’s appearance and clone it for each of the NPCs you want to create. This database is stored with each NPC and can be called up on demand as you need to populate the sim.

12.4.3 Using Particle Systems as Procedural Content Generators

Not all of the content in your game-based sim will be 3D. Many of the “atmospheric” types of content such as snow, rain, smoke, fog, and fire are usually made from particle systems that emit particles from scripted objects. These are the standard uses for particle systems, but they need not be limited to that. Particle systems and the scripts that create them can be utilized to make terrain-based content such as fields of flowers and snowdrifts, or visual connecters such as anchor chains, kite strings, and laser beams. They have been used to make background and distant content to add a visual depth to the virtual environment, such as moons, comets, and hilltop trees. These systems can also be worn by the avatar as fashionable particle wings or visible “perfume.” If you use Blender to do your modeling, you have a built-in particle system to help you model procedurally, as Andrew Price, the “Blender Guru,” demonstrates in this tutorial: http://www.blenderguru.com/tutorials/how-to-create-a-city/. He utilizes the hair-creating particle system set up on a grid to generate an entire city by using a group of buildings to replace the hair object. Brilliant!

12.4.4 Software and Plugins That Will Help You Work Procedurally

Another fertile source of information about software and plugins for 3D modelers like Blender and 3ds Max is found on the Virtual Terrain Project site (http://vterrain.org/). Its stated goal is “to foster the creation of tools for easily constructing any part of the real world in interactive, 3D digital form.” To that end, not only does the site provide software tools for you to download and try, there is a large database of plugins for other kinds of 3D modeling and terrain-generating programs.

12.5 PROJECT: UTILIZING A PROCEDURAL SCRIPT TO MAKE A FAIRY TREE

In this project you will be creating a fairy tree using some advanced-level scripts developed for this book by Michael Thome (Vex Streeter in the Metaverse). These scripts will work both in OpenSim as well as Second Life. Essentially, this tree is made from one mesh that is copied and rezzed in numerous variations to create its trunk, main limbs, and branches. The leaves displayed on them are created with a particle script that is embedded in each new version of the initial mesh that the script generates. UUID numbers are utilized in the scripts to call in the appropriate foliage and the floating sparkle particle graphics. We have included a foliage and sparkle particle graphics/texture with this project. However, you can also change those graphics/textures out for your own so you can customize the tree.

12.5.1 Designing with a Procedural Generator: Cautions and Downloads

12.5.1.1 Cautions

As with all scripts that create self-generating content, these scripts should be used with caution! Although the branch elements that the build.lsl script creates will switch their internal scripts to listening mode once the tree is finished “growing,” the creation process will cause a spike in server demand, so you want to start carefully and make sure that you are not going to crash the sim. As we go through this process, caution points will be italicized so you know what to watch out for.

12.5.1.2 Getting the Content for Creating the Tree

Everything you will need for making your first tree can be found at http://www.anncudworthprojects.com/ under the Extending VWD Downloads tab. Here is a list of what you will need for making your first tree:

Scripts that build the tree

- builder.lsl—This script tells the child object (the branch) in the contents of the trunk mesh where to manifest the new branches (children in the linkset), listens for grow and clean commands, and runs the dialogue menu for the tree.
- child.lsl—This script tells the new branches how to appear (they slowly fade into full opacity like a dream vision) and how the foliage and fairy sparkle graphics/texture will be shown.

Other components used for making the tree

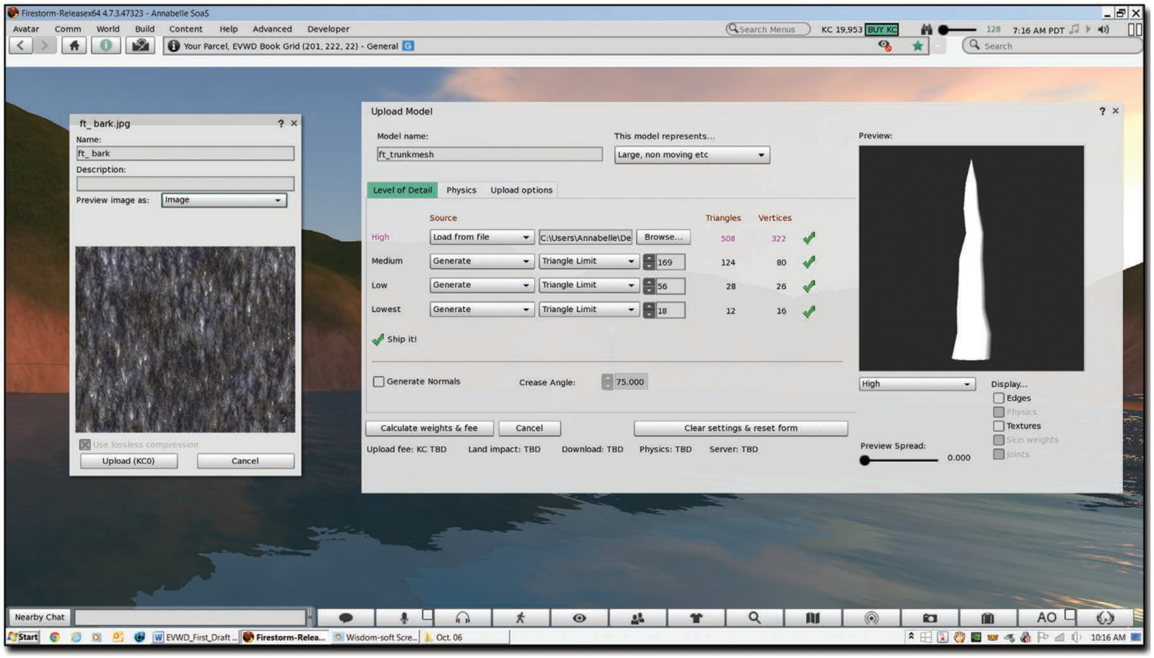
- pattern—This is a text file that is used to record the branch arrangement locations you create and is referred to by the build.lsl script as the tree “grows.”
- trigger.lsl—This script is optional and should be added if you want to make an on/off remote switch for your tree, perhaps as part of the game-based sim you are creating.
- ft_foliage1.tga and ft_sparkle1.tga—These are the basic graphics/texture files included with the tree project.
- ft_trunkmesh.dae—This is a mesh for use in making the trunk and branches of the tree.
- ft_bark.jpg—This is a graphics/texture for the tree trunk and main branches.

Using your favorite client viewer, upload the mesh and graphics/texture elements into a new folder called “Fairy Tree_All.” In Figure 12.2 you can see a screenshot of the uploading process for the mesh trunk and bark graphic/textures.

Also within this folder, create three new scripts and copy/paste the fairy tree scripting code into them to create your own copies of the aforementioned LSL scripts. Make sure to use the “select all” option, so you get every scintilla of Michael’s code. Once you have gotten the scripts into your inventory inworld, you will need to make sure that they will call for the correct particle graphics/textures. To do this, find the “ft_foliage1.tga” particle graphics/texture pattern in your inventory. Right-click on the texture and select from the dropdown menu “copy asset UUID.” This puts the texture’s UUID to your clipboard. Open up the builder script, and on line #18, delete the string number between the quotes, and paste in the UUID from your own inventory. Do the same for line #19 and the ft_sparkle1.tga graphics/texture. If you want to customize these textures, simply replace these UUIDs with your own from graphics/textures you have created. Finally, make a notecard in your fairy tree folder and copy/paste in the information from the “pattern” text file in the same careful manner you used on the scripts. OK, now you have everything you need to make a tree, and it is all inworld with you, so let’s do this thing!

12.5.2 Creating the Trunk and Branches

Step 1: Rezz the ft_trunkmesh, and apply the ft_bark.jpg texture to it. Adjust the tiling, specularity, bump graphics/textures, and so on until you think it looks perfect. In Figure 12.3, there is a screenshot of the trunk at this stage of completion.


Firestorm-Releasex64 4.7.3.47323 - Annabelle SoaS
Avatar Comm World Build Content Help Advanced Developer
Your Parcel, EVWD Book Grid (201, 222, 22) - General
ft_bark.jpg
Upload (KC0)
Cancel
Upload Model
Level of Detail
Physics
Upload options
Calculate weights & fee
Clear settings & reset form
Preview:
Display...
Edges
Textures
Nearby Chat

Name: ft_ bark

Preview image as: Image

Model name: ft_trunkmesh

This model represents... Large, non moving etc

Source				Triangles	Vertices
High	Load from file	C:\Users\Annabelle\De	Browse...	508	322
Medium	Generate	Triangle Limit	169	124	80
Low	Generate	Triangle Limit	56	28	26
Lowest	Generate	Triangle Limit	18	12	16

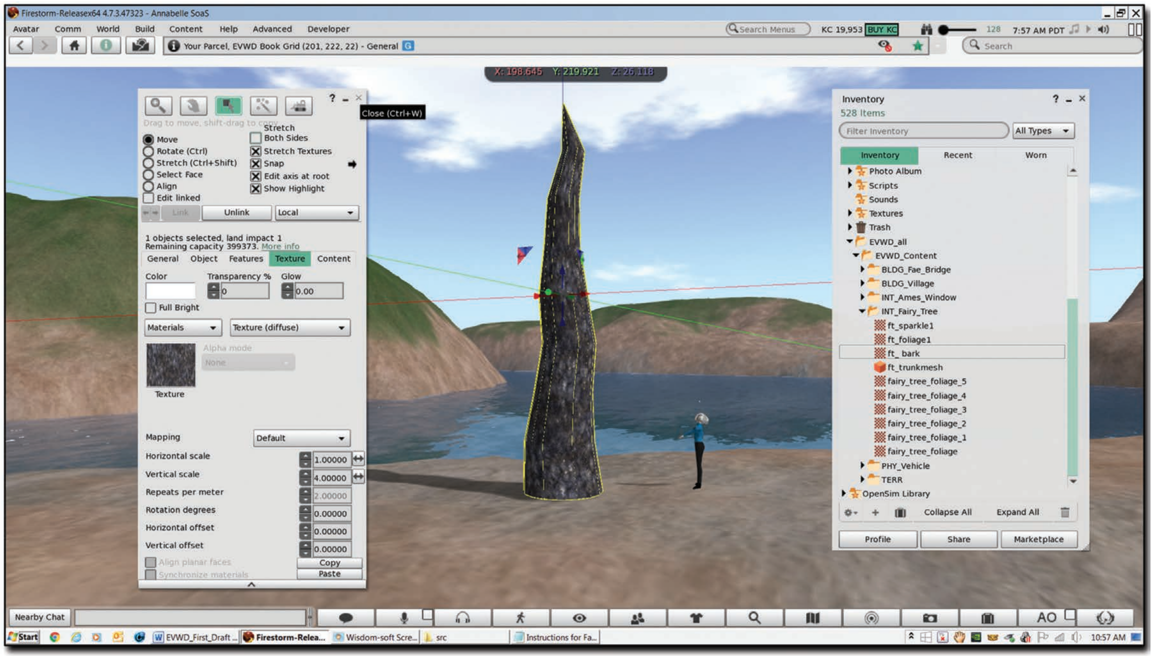
Ship it!

Generate Normals Crease Angle: 75.000

Upload fee: KC TBD Land impact: TBD Download: TBD Physics: TBD Server: TBD

Preview Spread: 0.000

FIGURE 12.2 Uploading process.



1 objects selected, land impact 1

Remaining capacity 399373.

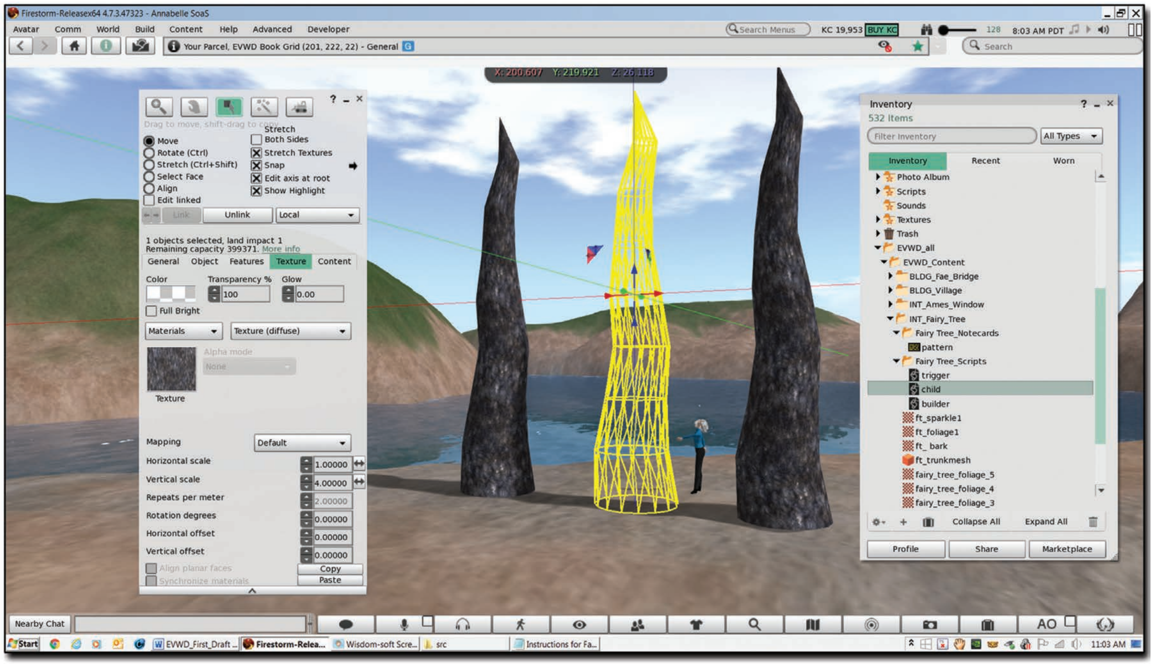
Transparency % 0 Glow 0.00

Materials Texture (diffuse)

Mapping Default

Horizontal scale	1.00000
Vertical scale	4.00000
Repeats per meter	2.00000
Rotation degrees	0.00000
Horizontal offset	0.00000
Vertical offset	0.00000

FIGURE 12.3 Working with the trunk element.


Firestorm-Releasex64 4.7.3.47323 - Annabelle SoaS
Avatar Comm World Build Content Help Advanced Developer
Your Parcel, EVWD Book Grid (201, 222, 22) - General
X: 200.607 Y: 219.921 Z: 26.118
Drag to move, shift-drag to copy
Move
Rotate (Ctrl)
Stretch (Ctrl+Shift)
Select Face
Align
Edit linked
Stretch
Both Sides
Stretch Textures
Snap
Edit axis at root
Show Highlight
Link Unlink Local
General Object Features Texture Content
Inventory
532 Items
Inventory Recent Worn
Collapse All Expand All
Profile Share Marketplace
Nearby Chat

1 objects selected, land impact 1
Remaining capacity 399371. More info

Transparency %	100
Glow	0.00
Full Bright	☐
Materials	Texture (diffuse)
Alpha mode	None
Mapping	Default
Horizontal scale	1.00000
Vertical scale	4.00000
Repeats per meter	2.00000
Rotation degrees	0.00000
Horizontal offset	0.00000
Vertical offset	0.00000

- Photo Album
- Scripts
- Sounds
- Textures
- Trash
- EVWD_all
  - EVWD_Content
    - BLDG_Fae_Bridge
    - BLDG_Village
    - INT_Ames_Window
    - INT_Fairy_Tree
      - Fairy Tree_Notecards
        - pattern
      - Fairy Tree_Scripts
        - trigger
        - child
        - builder
      - ft_sparkle1
      - ft_foliage1
      - ft_ bark
      - ft_trunkmesh
      - fairy_tree_foliage_5
      - fairy_tree_foliage_4
      - fairy_tree_foliage_3

FIGURE 12.4 Making the child object.

Step 2: Select, and shift/drag to make two more copies of this element. The second one will become the "child" element in the generation of the tree, so you should rename it "child Object" for the time being. Drag a copy of the child.lsl script into the contents of this mesh element copy, set the texture to full transparency (100%), and take it into your inventory, stashing it into your "Fairy Tree_All" folder. In Figure 12.4 is a screengrab of this step in the process.

Step 3: Add the builder.lsl script into the contents section of the ft_trunkmesh, and then add the "child object" mesh from your inventory in as well. Change the name of the "child Object" to just "Object" (no quotes). Finally, add in the "pattern" notecard you made in the previous step. Figure 12.5 shows you what the fairy tree trunk looks like fully loaded with the necessary contents.

Step 4: The next step is to add a linkset of branches for the builder script to rezz into a full tree. Use the last remaining trunk mesh, rename it branch if you like, and copy it three times. Then arrange those branches along the trunk in a pleasing pattern. Remember, the script will make smaller copies of the linkset on every main branch, so the spacing will be repeated. For a tall, "rangy" tree, space the linkset farther apart, and for a "bunchy-broccoli" tree, keep them clustered together. Once you have the branches where you want them, select the branches and link them to the tree trunk, making the trunk the key prim. In Figure 12.6 is a shot of a rangy type linkset used for the fairy tree.

12.5.3 Running the Fractal Tree Controls

Step 5: Now that you have the trunk loaded, and the tree branch linkset made, exit out of the Edit/Build mode, and with the touch hand pointer on the tree, click it to bring up the Fractal tree controls menu interface. Simple as this looks, there is quite a bit of functionality packed into it. Figure 12.7 shows the Fractal tree control menu and how it looks on the screen.

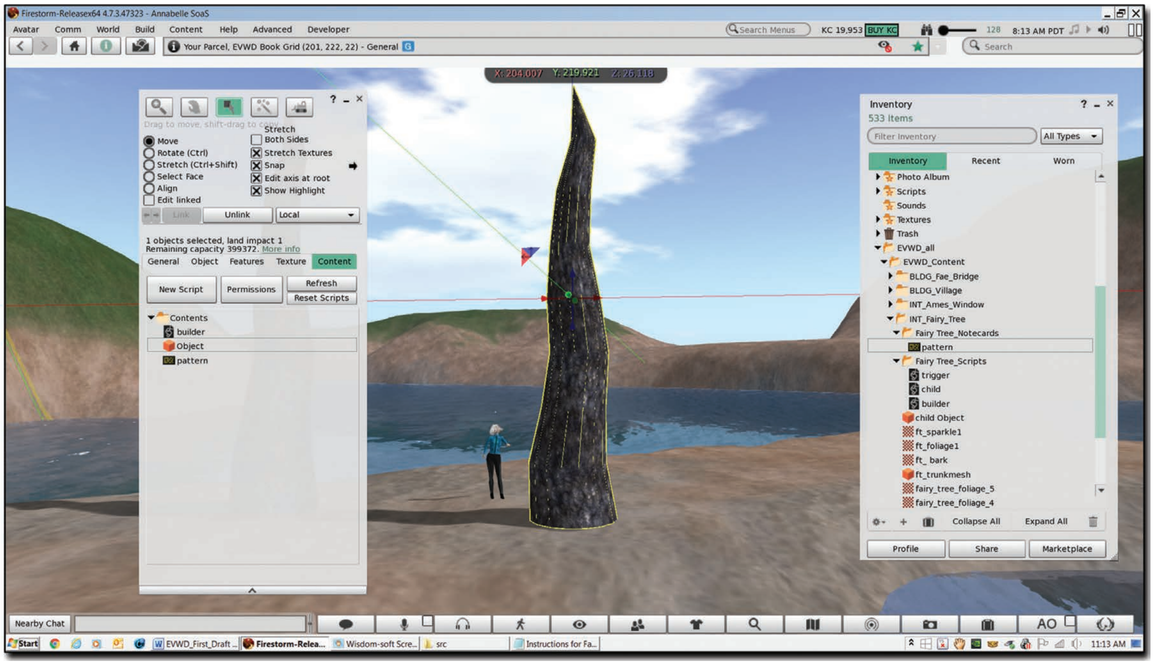
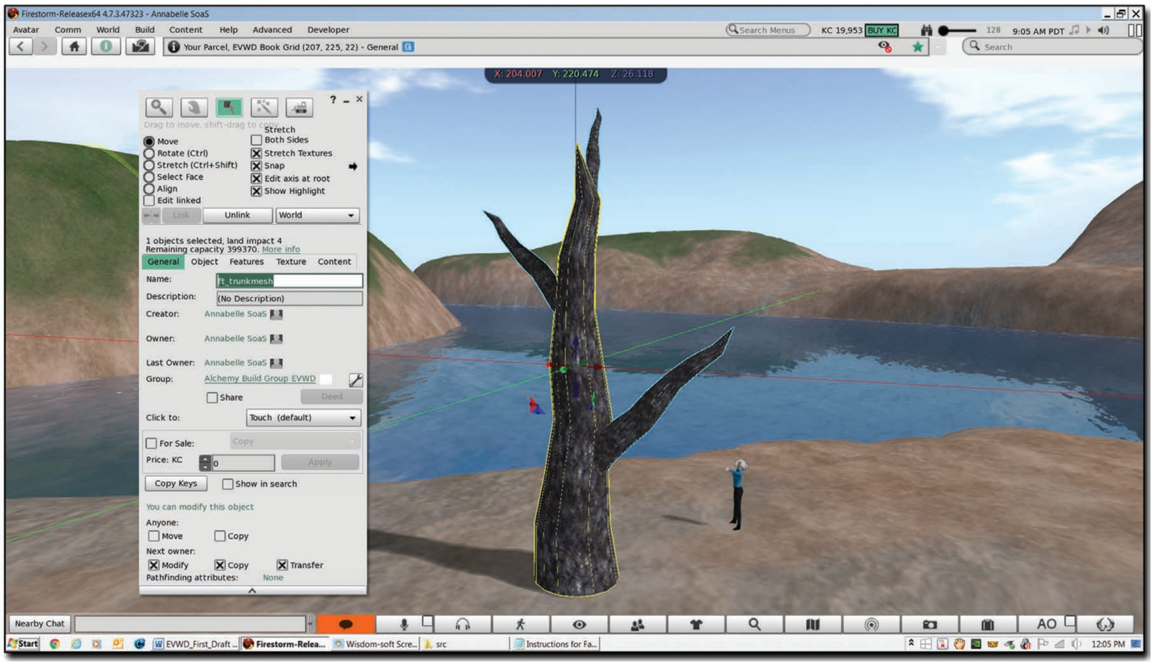


FIGURE 12.5 Fairy tree trunk fully loaded.


Firestorm-Releasex64 4.7.3.47323 - Annabelle SoaS
Avatar Comm World Build Content Help Advanced Developer
Your Parcel, EVWD Book Grid (207, 225, 22) - General
X: 204.007 Y: 220.474 Z: 26.118
Drag to move, shift-drag to copy
Move
Rotate (Ctrl)
Stretch (Ctrl+Shift)
Select Face
Align
Edit linked
Stretch
Both Sides
Stretch Textures
Snap
Edit axis at root
Show Highlight
Link Unlink World
1 objects selected, land impact 4
Remaining capacity 399370. More info
General Object Features Texture Content
Copy Keys
Show in search
Nearby Chat

Name: ft_trunkmesh
Description: (No Description)
Creator: Annabelle SoaS
Owner: Annabelle SoaS
Last Owner: Annabelle SoaS
Group: Alchemy Build Group EVWD
Share Deed
Click to: Touch (default)
For Sale: Copy
Price: KC 0 Apply
You can modify this object
Anyone: Move Copy
Next owner: [✓ Modify] [✓ Copy] [✓ Transfer]
Pathfinding attributes: None

FIGURE 12.6 Linkset in fairy tree.

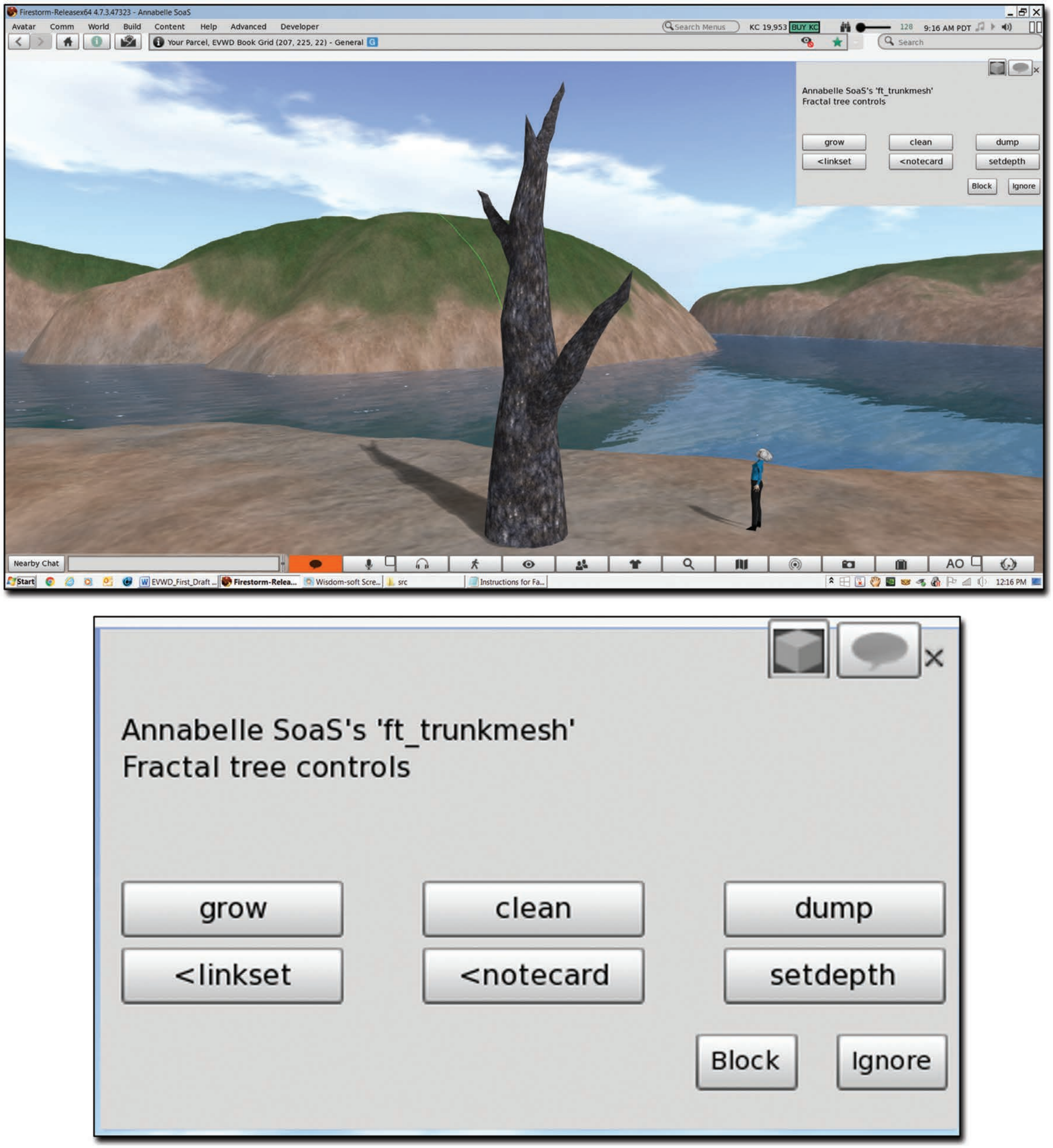


FIGURE 12.7 Fractal tree controls (close-up in bottom half).

Here is what the buttons on this control panel will do:

grow—Starts the tree's branch rezzing process.

clean—Deletes all of the child elements that the tree rezzed, leaving just its trunk, or the base linkset you have created.

dump—Will "say" the current configuration of the tree's linkset to the nearby chat channel window, so you can copy the text and paste it into a new notecard. This feature lets you save out your favorite tree configurations for future building.

<linkset—Allows you to load the positions of the tree's branches into the tree's memory, so you can modify, change the setdepth number while you work toward the kind of tree you want to create. This is the information that gets "said" to the nearby chat channel with the dump button.

<notecard—Loads the tree branch positional information from the pattern notecard. This information is overwritten each time the <linkset button is used.

setdepth—*This button opens another menu that allows you to choose how many iterations of the linkset will be rezzed, and this one should be used with caution!* For example, if you have created a tree with 1 trunk and 3 limbs, and setdepth is configured for 3, then the resulting tree will have 121 objects (branches) and a land impact of 124.

Step 6: Now you are almost ready to grow your first tree. The first thing you should do with a new tree is to click the <linkset button, so the positions of the tree limbs are recorded on the pattern notecard. This may take a few seconds, the menu will disappear, and the tree will chat back to you telling you something like "[04:27] ft_trunkmesh: Done with linkset."

Step 7: Now click the tree again to bring back the menu and click the setdepth button. *Now, carefully set the depth. Start with something low like 2 or 3, and see how that works out.* Again the tree will chat back to you saying something like "[04:29] ft_trunkmesh: Set depth to 3."

Step 8: OK, just to double check: (1) tree limbs are linked to the trunk and the trunk is the key prim, (2) <linkset is loaded, and (3) you have picked a number for setdepth. Now, you can hit the "grow" button, and watch it go. Do not get anxious if you do not see any branches appearing for 5 to 10 seconds; remember it is scripted to appear slowly. In Figure 12.8 is a screenshot of the "grown tree" showing off its particle leaves and sparkles.

Step 9: If you like the look of the tree, use the dump button to save the limb configuration to the nearby chat channel window. Then make a "New Tree_1" subfolder under the "Fairy Tree_All" main folder, and store the linkset dump to chat on a notecard named "pattern" in there. As you go along, creating trees and saving their linkset on notecards for future use, you will probably want to use a more descriptive name for the subfolder, so you will know at a glance what kind of tree the pattern will create. You may also want to include a screengrab of the grown tree in the file for visual reference.

Step 10: When you decide to create a new tree based on a linkset that you have developed, just repeat the instructions. In step 3, substitute your own notecard named "pattern" into the contents of the tree, and then continue with the instructions.

12.5.4 Further Development

Now that you have gotten the hang of working with a procedural system, you probably can see lots of potential for developing all sorts of things with this script. Twisted stone towers and magically appearing branched waterfalls are just two things that come to mind. Here are a few final notes:

- Remember to use the clean button to clear out the extra elements you do not want.
- Remember to use the dump button to preserve the settings on the linkset.

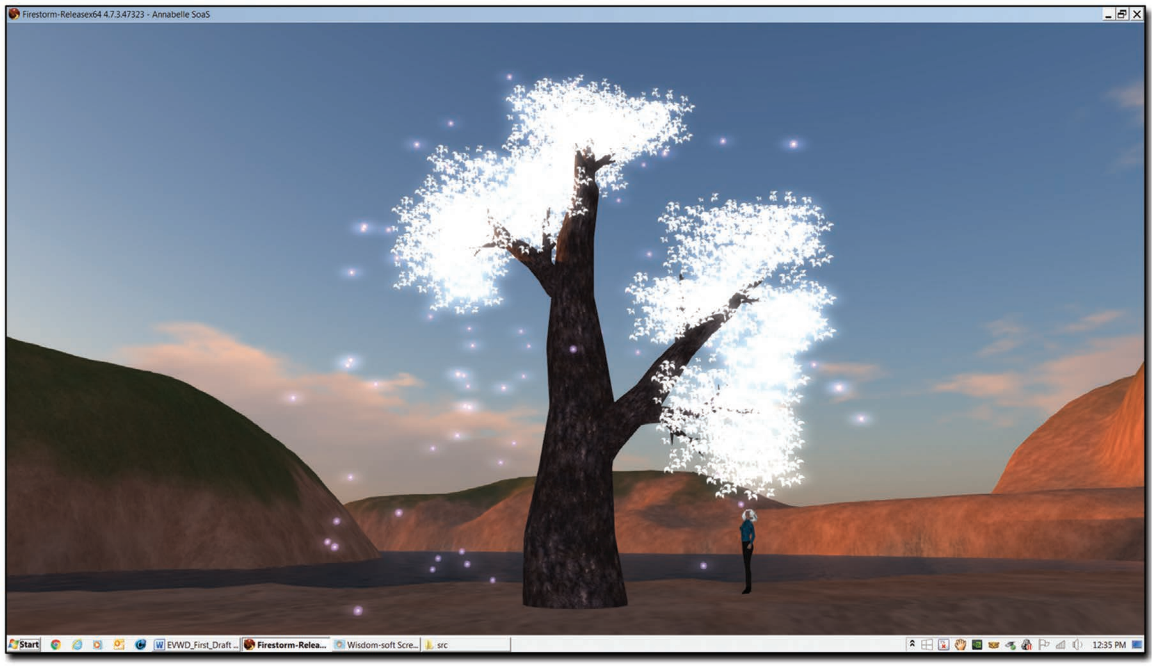


FIGURE 12.8 Screengrab of final tree after it has grown.

- You can unlink and delete the branches from the trunk, once you have made a fully functioning tree; it stores the linkset info in its notecard.
- If you want the tree to be activated by other people's touches, put the trigger.lsl script into another prim or object. This will let them touch a separate object (maybe a magic bean?) to make it grow. A second touch will make it all disappear.
- If you decide to set up a grove of these trees (check with your grid manager before you put that kind of load on the server) change the script so it loads the notecard automatically when you rezz the tree trunk.

12.6 CHAPTER SUMMARY AND FINAL THOUGHTS

This chapter has been a very wide-ranging view of theory and methods used for world building and designing with procedural techniques. The world building exercise will help you develop your world story and creation myth. The background information on the fundamental theories of procedural generation, such as shape grammars and L-systems, will help you speak in greater depth with scripters who are collaborating with you. In general, it is important to know the workflow, keep the import/export file formats in mind, strive to keep the number of tools you use on a project to a few powerful ones, and learn to use them well. Study the ways other people have used procedural techniques and systems in imaginative ways, such as the way Andrew Price uses the hair particle system to make a city, or how Christian Holzer used CityEngine, Maya, and Photoshop to make a destroyed city for the *Dark Realm* game (http://www.esri.com/software/cityengine

/industries/destroyed-city). Finally, remember to mix a little “handmade” content into your virtual environment. Procedural content generation can do the heavy lifting in world building but cannot replace the essence of human fabrication, which adds more life to the environment.

REFERENCES

1. Gary Gygax and Dave Arneson, *Dungeon Master’s Guide* (D & D Core Rulebook) (Wizards of the Coast, 2014), pt. 1, 9–68.
2. Barbara C. Sproul, *Primal Myths: Creation Myths around the World* (HarperCollins, 2013), Kindle edition, 173.
3. Paul G. Zolbrod, *Diné Bahane’: The Navajo Creation Story* (University of New Mexico Press, 1987), Kindle edition, 623–630.
4. Barbara C. Sproul, *Primal Myths: Creation Myths around the World* (HarperCollins, 2013), Kindle edition, 4791–4794.
5. Ann Cudworth, Layton Destiny, and Vicki Brandenburg, “Virtual Game-Based Sim Created for Second Life at the IBM Exhibit C region,” built in 2010 and displayed until March 2011, accessed December 28, 2014, https://www.flickr.com/photos/annabellefanshaw/sets/72157625515733846/show/.
6. Jimmy Maher, “Akalbeth,” *The Digital Antiquarian—An Ongoing History of Computer Entertainment* (blog), December 18, 2011, accessed December 27, 2014, http://www.filfre.net/2011/12/akalabeth.
7. Jimmy Maher, “Elite (or, The Universe on 32 K Per Day),” *The Digital Antiquarian-An Ongoing History of Computer Entertainment* (blog), December 26, 2013, accessed December 27, 2014, http://www.filfre.net/2013/12/elite/.
8. You can watch the rest of the interview with Sean on Game Informer’s YouTube channel, “A Behind-the-Scenes Tour of No Man’s Sky Technology,” YouTube video, 33:58, posted by Game Informer, December 5, 2014, http://youtu.be/h-kifCYToAU.
9. Jonah Weiner, “Where Do Dwarf-Eating Carp Come From?,” *New York Times*, July 21, 2011, accessed December 24, 2014, http://www.nytimes.com/2011/07/24/magazine/the-brilliance-of-dwarf-fortress.html.
10. David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steve Worley, *Texturing and Modeling: A Procedural Approach*, 3rd ed. (The Morgan Kaufmann Series in Computer Graphics) (Morgan Kaufmann, 2002), 1.
11. David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steve Worley, *Texturing and Modeling: A Procedural Approach*, 3rd ed. (The Morgan Kaufmann Series in Computer Graphics) (Morgan Kaufmann, 2002), 2.
12. *Wikipedia*, s.v. “Fractal Landscape,” accessed December 30, 2014, http://en.wikipedia.org/wiki/Fractal_landscape.
13. Ken Perlin, “Bronze Ball Made from Procedural Texture,” accessed December 30, 2014, http://mrl.nyu.edu/~perlin/homepage2006/vnt/index.html.
14. David Eppstein and Scott Drysdale, “Voronoi Diagrams: Applications from Archaology to Zoology,” *Geometry in Action*, July 19, 1993, accessed December 30, 2014. http://www.ics.uci.edu/~eppstein/gina/scot.drysdale.html (Accessed on 12/30/2014).
15. David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steve Worley, *Texturing and Modeling: A Procedural Approach*, 3rd ed. (The Morgan Kaufmann Series in Computer Graphics) (Morgan Kaufmann, 2002), 20–22.
16. David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steve Worley, *Texturing and Modeling: A Procedural Approach*, 3rd ed. (The Morgan Kaufmann Series in Computer Graphics) (Morgan Kaufmann, 2002), 558–559.
17. *Wikipedia*, s.v. “Shape Grammar,” accessed December 30, 2014, http://en.wikipedia.org/wiki/Shape_grammar.
18. David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steve Worley, *Texturing and Modeling: A Procedural Approach*, 3rd ed. (The Morgan Kaufmann Series in Computer Graphics) (Morgan Kaufmann, 2002), 307–312.
19. *Wikipedia*, s.v. “Procedural Generation,” accessed December 30, 2014, http://en.wikipedia.org/w/index.php?title=Procedural_generation&oldid=636993400.

20. *Wikipedia*, s.v. “Voronoi Diagram,” accessed July 19, 2015, https://en.wikipedia.org/w/index.php?title=Voronoi_diagram&oldid=671933416.
21. David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steve Worley, *Texturing and Modeling: A Procedural Approach*, 3rd ed. (The Morgan Kaufmann Series in Computer Graphics) (Morgan Kaufmann, 2002), 229–231.
22. David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steve Worley, *Texturing and Modeling: A Procedural Approach*, 3rd ed. (The Morgan Kaufmann Series in Computer Graphics) (Morgan Kaufmann, 2002), 557.
23. Christian Holzer, “Destroyed City,” *ESRI CityEngine*, accessed January 3, 2015, http://www.esri.com/software/cityengine/industries/destroyed-city.
24. George Stiny and James Gips, “Shape Grammars and the Generative Specification of Painting and Sculpture,” paper submitted to IFIP Congress 71 in Area 7, 1972, accessed January 3, 2015, http://www.shapegrammar.org/ifip/SGIFIPSubmitted.pdf.
25. Przemyslaw Prusinkiewicz, Aristid Lindenmayer, James S. Hanan, F. David Fracchia, Deborah Fowler, Martin J. M. de Boer, Lynn Mercer, *The Algorithmic Beauty of Plants* (Springer-Verlag, 1990), 2, accessed January 4, 2015, http://algorithmicbotany.org/papers/abop/abop.pdf.

13 Designing Virtual Environments for Head-Mounted Displays

At the core of the reality that each and every one of us experiences lies the fact that we are inference machines, not objective visitors, by which I mean that there is, presumably, a real world out there, and your brain is taking the very limited signals coming from your sensors and trying to infer what the state of that real world is based on its internal model.

Michael Abrash
Oculus chief scientist, presentation at the Facebook Developers Conference 2015

13.1 OVERVIEW OF DESIGNING VIRTUAL ENVIRONMENTS FOR HEAD-MOUNTED DISPLAYS (HMDs)

Virtual reality (VR) has been with us a lot longer than you may think. The actual concept of what virtual reality could be is described by Plato (428 to 348 BCE) in his parable called the Allegory of the Cave [1]. In movies like *Tron* (Walt Disney Productions, 1982), *The Lawnmower Man* (New Line Cinema, 1992), and *The Matrix* (Warner Brothers, 1999), virtual world metaphors have been created to provide fantasy entertainment. These movies, which often pushed the cinematic technologies to their limits, portrayed artistic interpretations of virtual spaces and what it would be like for us to visit them. They showed us the grandeur, the geometrical enchantment that a virtual environment could cast over us. It has been more than thirty years since *Tron*, and now companies such as Oculus, Valve, and Samsung are developing stereoscopic virtual reality devices that may become as ubiquitous as smartphones. Stereoscopic technology also has a long history. The first device was created by Sir Charles Wheatstone in 1838, and variations of it have been used throughout the last two centuries [2]. In 1861 another version was developed by Sir Oliver Wendell Holmes, the View-Master was introduced to the American audience at the New York World's Fair in 1939, and the Google Cardboard viewer that utilized images from a smartphone display was launched in 2014 [3]. Many of us already use devices on a daily basis that enhance or replace part of our real environments. Sometimes we are partially in the virtual world, using augmented reality (AR) to play a game like *Ingress* (2012 from Niantic Labs at Google) on our smartphones, and sometimes we are completely immersed within a virtual world using a head-mounted display (HMD). In this chapter we will consider various aspects of the stereoscopic virtual environment in an HMD and how these aspects influence the design choices that this completely immersive environment presents to us. Here are some of the key concepts we will cover in this chapter:

- The fundamentals of how a VR HMD works
- Five dimensions of design for virtual environments
- Design challenges: latency, drift, and judder

13.1.1 The Challenge of HMD Acceptance and Designing for Social Virtual Reality (VR)

"It seems truly short sighted for the industry to focus on HMDs as the primary viewing platform for the future of VR. The fact that VR headsets basically put their users into a cocoon of limited movement and isolation from real life also does not offer us a hopeful future for social interactions or experiencing real life," wrote Scott Highton in "The Future of VR – Is This All There Is?" [4]. Highton's observation, while valid, may become less significant if the new social networks dedicated to VR, such as ConVRge (http://www.convrge.co/) and AltspaceVR (http://altvr.com/), gain wide acceptance. Much of this technology is still untried in the general consumer arena, and the level of acceptance for this kind of display may be less than expected, in spite of the hype and financial backing enjoyed by the larger players like Oculus, htcVIVE, SONY, and FOVE. A nimble virtual environment designer would be wise to prepare for designing content that plays well in both VR and AR environments, creating modular content with elements that can stand alone or collectively. Elements that work in a mixed reality environment as well as an immersive virtual world would be a good achievable goal for the design team and pay dividends down the road as the content can be repurposed across the new platforms that will undoubtedly emerge. At the end of this chapter you will find a project that concerns itself with creating an optical illusion device, the Ames Window. With the 3D content provided, you will assemble, and script a "fairy compass" based on this optical illusion for your growing game-based sim.

13.2 OVERVIEW OF A VR HMD AND HOW IT WORKS

We are still in the primordial era of virtual reality environment design. As this branch of design evolves, we will have to create new user interfaces and new ways to clearly present our message within the users' sphere of perception. Since it is entirely likely you will be designing for virtual reality headsets like the Oculus Rift or the Samsung Gear VR in the near future, a good working knowledge of how this device creates and maintains the virtual environment is necessary for you to develop realistic expectations about what kind of designing is best for this system. In Figure 13.1 is a diagram of how the basic component parts of a virtual reality headset similar to the Oculus Rift DK2 work together to create the immersive optical illusion we call virtual reality [5,6].

13.2.1 HMD: Major Parts and Their Functions

As you can see in Figure 13.1, there are three main parts to virtual reality systems like this: (1) the display screen (in the Oculus Rift DK2 it is a Samsung OLED display similar to a smartphone display) is viewed through plastic lenses for the maximal field of view (100 degrees FOV), (2) a spatial tracking device that indicates the body position of the user (for the Oculus DK2 it is a specially developed infrared [IR] camera that sees the 40 IR LEDs emitting radiant energy beyond the human visible spectrum behind the front cover of the headset), and (3) the personal computer that is running the game or game-based sim environment [7]. Inside of the headset is a microelectromechanical systems (MEMS) device. This is a nanotech-scaled system that you are probably familiar with. There is a similar device in your smartphone that lets you do landscape to portrait image rotation on the phone screen, as well as dead reckoning navigation and motion detection. In the VR headset, the MEMS unit includes a six-axis gyroscope, accelerometer, and magnetometer. This device enables the HMD to acquire orientation tracking to measure linear acceleration, the angular velocity, and the 3D transformations created by your head movement. (Note: For more about these kinds of movements, see Chapter 8, Section 8.4.2.) This data is combined with the positional data the IR camera is gathering from the movements of the IR LEDs glowing from your headset. All the positional data from your body

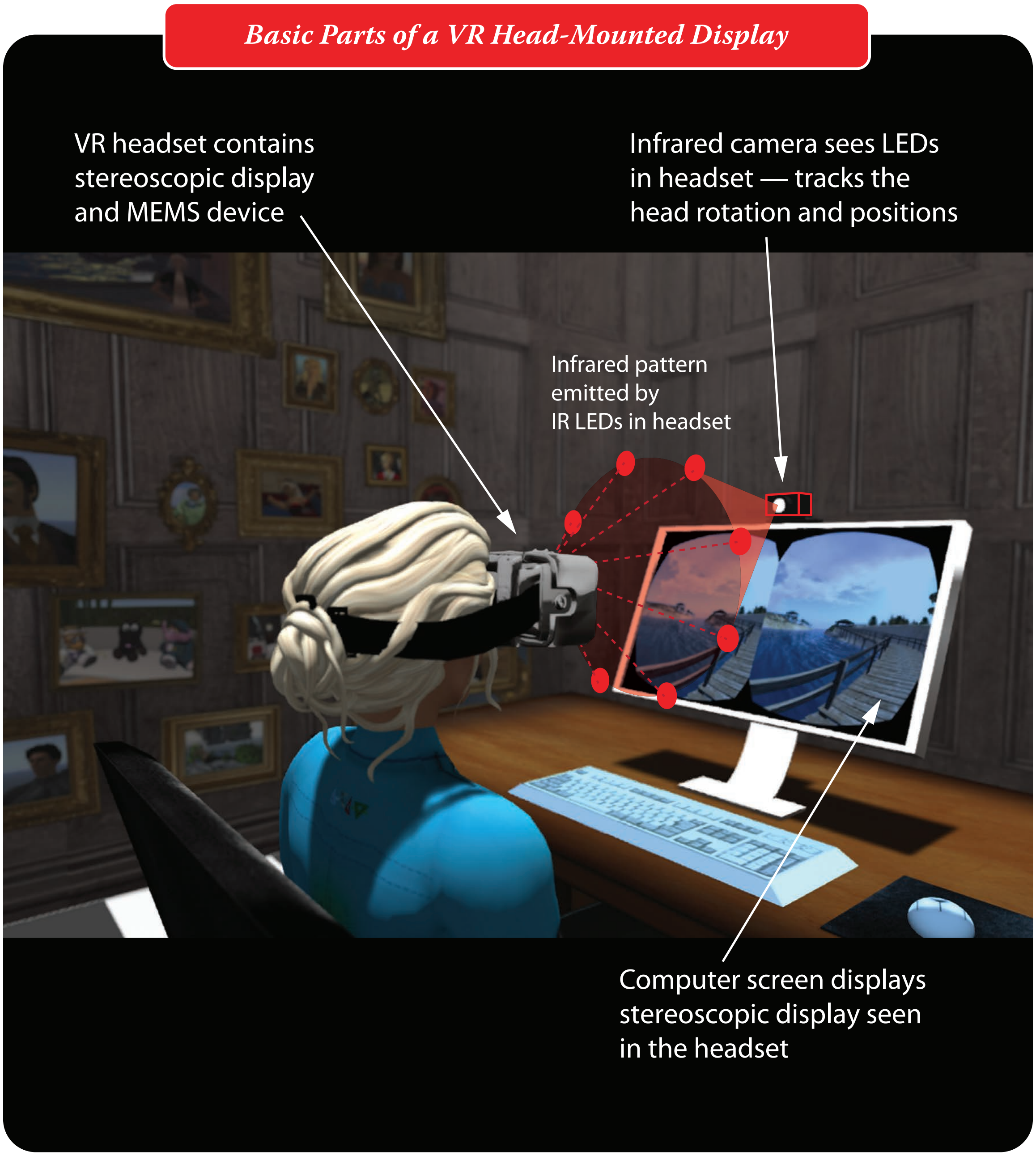

Basic Parts of a VR Head-Mounted Display
VR headset contains stereoscopic display and MEMS device
Infrared camera sees LEDs in headset — tracks the head rotation and positions
Infrared pattern emitted by IR LEDs in headset
Computer screen displays stereoscopic display seen in the headset

FIGURE 13.1 Basic component parts of a VR headset system.

and orientation data from the head position is fed into the graphics engine processors and used to change the position of your point of view (POV) in the scene you are viewing. Would you be able to identify yaw, pitch, and roll with your head movements? Here is a mnemonic device to correlate these primary movements with some basic human communication modes: (1) Shake your head to say no and you are creating a yaw-based move, (2) nod to say yes and you are making a pitch-based move, (3) tilt your head to one side in curiosity to say what and that is a roll maneuver. (See Chapter 8, Section 8.3.7.)

13.2.2 Future Developments for VR

It will be interesting to observe how the launch and subsequent popularity of a well-financed VR device will drive the development of the technology that supports it. Given the graphics processing that a VR game demands, it is no surprise that the computer required to run it falls into the “high end” category of specifications. For PC-based games at 1080p resolution at 60 Hz, a 60 frame per second (fps) display is considered acceptable. However, a VR-based game using the Oculus Rift will run at 2160 × 1200p resolution at 90 Hz and require a 90 fps display. The amount of rendering required will be approximately 3 times greater than that required by a standard PC game [8]. Stanford University and Nvidia have started to work on another kind of HMD that utilizes light field technology, a system that records the intensity and direction of light in a scene from multiple viewpoints, to provide a truer sense of depth in the view. Their first prototype uses layers of transparent LCD screens to put holographic-like images in front of the viewer’s eyes [9]. Other devices are being developed that combine HMDs and AR. Google has invested heavily in Magic Leap [10] and Microsoft has HoloLens [11] under development. At some point the distinction being made between AR and VR may be insignificant, as the 3D overlay from an AR device completely fills our real-life view of the world and the standard becomes a mixed reality.

13.3 FIVE DIMENSIONS OF DESIGN IN HMD VIRTUAL ENVIRONMENTS

As a designer you will be challenged by the “intermediary” nature of virtual environments presented via an HMD. This virtual space is at the intersection of the real world with all its noisy, chaotic data, and the virtual world that only has limited means to cope with all of this information. As computer users we have become accustomed to interacting with a flattened virtual space in a 2D window, our computer screen. Now we will be completely surrounded by the virtual environment, and visual and audible information can be perceived from any direction we can turn our head and body toward. We have traded out a flat screen for a sphere of perception. You may recall our exploration of Peter Sloterdijk’s theories regarding spheres and existential space in Chapter 5, Section 5.1.1. He believes that we experience and create the space or sphere in our minds simultaneously. If this is the truth, then how will a designer work within this conceptual framework? There are numerous possible workflows and design concepts to consider. Let us focus on five of the most important and compelling aspects or “dimensions of design” for virtual environments, the ideas that appear again and again whenever the content and structure of virtual world design is a topic of discussion. In the following sections are brief discussions of these five dimensions of design: (1) agency, (2) presence, (3) affordance, (4) story, and (5) safety.

13.3.1 Agency

We will take the sociological meaning of agency, in this case the capacity of an individual to act and interact (move, gesture, communicate) within the virtual environment. How can the design of a VR environment support agency? This is a question with a thousand answers. Agency in a virtual environment is the combined result of visual understanding of the virtual content displayed and the feedback of data or reaction from the individual visitor within the virtual space. For instance, if the roadway/pathway/route in the virtual environment is clearly defined, visually obvious, then there is a strong possibility that the visitor will travel along it to explore the environment. If the road is hidden, or broken off, or branches in several directions, then the visitor is presented with the cognitive problems of finding the road, making a bridge, or choosing their way. When designers decide to put in these obstacles to the visitor's agency, then they must offer a way for the visitor to regain agency either through interactivity or response feedback to the visitor's choices. For instance, designers can offer a way to make the hidden road appear, provide elements that will enable the visitor to build a bridge, or offer an informational source (a sign, a talkative hermit, etc.) for the visitor, so that the visitor can gain insight as to the possibilities each choice offers. Agency in a virtual environment has very strong ties to the interfaces being used by a visitor. When wearing a VR device on the head that only tracks head and body movement, there is little if any way the visitor can utilize the keyboard and mouse to create agency in the virtual environment. The addition of hand-tracking devices opens this level of agency to the visitor, but the inclusion of objects that need to be clicked or touched in the HMD-viewed virtual environment becomes a new design challenge. How big should they be, where should they be placed in the field of view, and how will they function are all questions that can only be answered by testing the practical application of such devices. Only a small part of your eye's visual field is used to read text, typically the fovea (see Chapter 7, Section 7.3.1), so placement and size of the signage is crucial for communicating in your virtual environment. As John Carmack speculated on Twitter (June 1, 2014): "I wonder if there would be utility in dynamic sizing of UI/text with accurate gaze tracking. Constant size text useless outside fovea." The problems abound for the HMD–VR environment designer as they seek to deliver agency to the visitor, but many companies such as Softkinetic (http://www.softkinetic.com/Products/UserExperiences) and Razer (http://www.razerzone.com/vrpromo/) are working on the problem of bringing "touch and click" into the HMD–VR environment. Having the capacity to see their hands in a virtual space in front of them does allow for visitors to click and touch virtual objects, but it also causes body fatigue as they are required to hold up their hands within sensors range or move their handheld devices around their bodies.

Returning to the visitor on a road scenario, you will want to consider how the visitor will travel. In his address to the Game Developers Conference in 2015, John Carmack mentioned these design considerations when working with a mobile HMD–VR environment [12]:

- Avoid lots of bounding, jumping, parabolic motions
- Do not build narrow tunnels, because of the large amount of parallax that the computer will have to process
- Skip near-field objects because position tracking isn't there yet
- Do not have easing into acceleration, just go

In his keynote address to the 2014 OpenSim Community Conference, Dr. Steve Lavalle mentions that teleportation as an option to give the visitor some agency without creating uncomfortable levels of vestibular mismatch, the discomfort caused by movement in the virtual environment that is not matched by what your body is sensing in the real world [13].

13.3.2 Presence

The quest for realism seems to be the quixotic goal of virtual environments. Realism requires detail and detail requires large amounts of bandwidth and rendering horsepower. Consider this question for a moment. Do we need to have realism in the virtual environment to create a sense of presence that will allow the visitor to become immersed in the world? Most environments that are in virtual worlds fall far short of realism and opt instead for a stylistic and/or minimalistic representation. After all, most of us have active imaginations that are willing to fill in the details of the HMD–VR environment. Is it quantifiable, is there a minimum level of realism that will capture our imagination, transport us, and give us a sense of presence in the virtual environment? Early in 2015, Oculus VR formed Story Studio (https://storystudio.oculus.com/en-us/) to explore what storytelling can become in virtual environments. Max Planck (technical director), Saschka Unseld (lead creative director), and Ramiro Lopez Dau (animation supervisor), all former employees of Pixar, have completed two projects so far: *Lost*, a 4-minute short about a misplaced robotic hand, and *Henry*, the story of a physically unavailable hedgehog (because of his spiky exterior) who longs for emotional connections. Neither of these environments or the characters within them are realistic, however they do engage our feelings, and much is going on "under the hood" to ease us into the story and create a sense of being there, of presence. These two narrative virtual environments create a place for your presence in different ways. The virtual space in *Lost* is designed to ramp you slowly up into the story, giving you time to adjust to the VR experience. The environment is a moonlit forest, and there to meet you is your firefly guide. There is an artificial intelligence (AI) program running under the visual experience that waits for you to become comfortable in the surroundings and ready to receive the titles that appear. Sound cues have also been included to guide you around the forest, helping you to discover the story within it. "It's like an immersive theatre. It's a stage," Max Planck says of the *Lost* viewer experience [14].

13.3.2.1 Presence, Artificial Intelligence, and Henry the Hedgehog

The second project completed by Oculus Story Studios was *Henry*, the story of a lonely hedgehog having his birthday and looking for friends who can attend the celebration in spite of his quills. Henry's coding allows him to make eye contact and respond to the visitor in his woodland home. This creates a powerful bond with the visitor and increases the empathy they feel for Henry, but may also usurp the decisions and control the director has over the story [15,16]. This is one of the many interesting challenges that HMD–VR is bringing to storytelling and creative production. It is possible that many of the old tropes and systems for making a story work will evolve into something different. In the near future, we may see coded AI entities like a "director/storyteller AI" that partners with us to set the parameters of the story, or possibly an "empathetic AI" that senses our mood and picks a new path in the storytelling to complement it. All design decisions impact the creation of presence in the virtual environment, so we need to pay attention to what supports presence and helps us connect the visitor to the environment and the story within it.

13.3.3 Affordance

Affordance, or the actual or perceived qualities of a design or object that implies to the observer how something should be used, is a crucial aspect of HMD–VR environments. Just as in the real world, we need to know what virtual doorknobs look like and how they work. Never mind that a virtual doorknob may open a door that sends you plummeting into Wonderland to have tea with the March Hare. The design concept of affordance is a biggie in HMD–VR, because currently there are several unnatural aspects to head-mounted displays that may delay or derail a user's capacity to negotiate the virtual environment. Deepen your understanding of interface affordance through books such as *Information Visualization: Perception for Design* by Colin Ware and *Designing with the Mind in Mind: A Simple Guide to Understanding User Interface Design Guidelines* by Jeff Johnson to optimize the affordance of the elements in your virtual environment. For example, Ware notes that "when using a head-mounted display to read text, make the width of the text area no more than 18 degrees of (the) visual angle" [17]. Think about how this simple rule will affect the interactive elements you are designing for that space plane cockpit or entry signage for a virtual park. Focus on the *visual hierarchy* in your design, suggests

Johnson, to clarify and deepen the understanding of the affordances surrounding the visitor in your virtual environment [18]. As designers, we should strive toward making the medium transparent and try to find a way to make every aspect of the HMD–VR environment supportive and not in the way of our message or storytelling efforts. By making the affordances obvious, we lower the cognitive load on the visitor as they move around the HMD–VR environment and increase the capacity for storytelling. Some of this may be accomplished by realistic rendering of real-scale models and devices within the HMD–VR environment, and some may be achieved with iconographic objects such as what we already see on our 2D screens. Some notable examples of these universally understandable visual elements are the file tabs and cascading menus on the top line of most software interfaces, the crossed arrows used for selecting and moving an object, and the intersecting circles used to help us rotate an object in 3D space. Eventually, these will all be ubiquitous in virtual environments, and whatever hand-tracking device or verbal command we use will seem natural and comfortable. There are so many interface tools to design for HMD–VR environments that it may seem to be a daunting task at first, but eventually we will crack the design problems and create intuitive interfaces that we use almost subconsciously. As Janet Murray says in her book *Hamlet on the Holodeck*:

> As the virtual world takes on increasing expressiveness, we will slowly get used to living in a fantasy environment that now strikes us as frighteningly real. But at some point we will find ourselves looking through the medium instead of at it. Then we will no longer be interested in whether the characters we are interacting with are scripted actors, fellow improvisers, or computer-based chatterbots, nor will we continue to think about whether the place we are occupying exists as a photograph of a theatrical set or as a computer-generated graphic, or about whether it is delivered to us by radio waves or telephone wires. At that point, when the medium itself melts away in to transparency, we will be lost in the make-believe and care only about the story. We will not notice it when it happens, but at that moment—even without the matter replicators—we will find ourselves at home on the holodeck. [19]

Welcome to the land of absolute affordance.

13.3.4 Story

The story in a HMD–VR environment is the primary focus, and everything in that environment should exist to serve it. For example, let us suppose you decided to tell the story of Romeo and Juliet, the star-crossed lovers of Verona, Italy, in your virtual environment. Here is how Shakespeare's words open the story in the play:

> Two households, both alike in dignity,
> In fair Verona, where we lay our scene,
> From ancient grudge break to new mutiny,
> Where civil blood makes civil hands unclean.
> From forth the fatal loins of these two foes
> A pair of star-cross'd lovers take their life;
> Whole misadventured piteous overthrows
> Do with their death bury their parents' strife.
>
> *Romeo and Juliet*, 1.1

Within eight lines, Shakespeare has told us (1) where this story will take place and what kind of place it is, that is, "fair Verona"; (2) that there is an ancient blood feud between two families who live there; (3) that their children will fall in love and die together; and (4) that the event will end the feud between these two families. Every single line in this prose tells us a key piece of information about the story and what will happen in it, and it leaves just enough out so that we immediately want to hear the rest. Now imagine how you would tell about Romeo and Juliet from the multiple points of view and visual angles, and you will have begun to create the film version of this story. For HMD–VR, the design question branches in multiple directions:

- How do you design a 3D environment that allows the actions of this story to happen, that is, the houses of the Montagues and the Capulets, the streets of Verona, and the tomb that Romeo and Juliet have their final scene in?
- How do you design visual and audible access at any point along the storyline, so the visitor can connect with the story, even if they start exploring in the final tomb?
- What if the story was told by non-player characters standing in the space like actors on a bare stage, just using Shakespeare's words to create the environment in our imagination?
- What if the non-player characters are encoded to procedural actions and the visitor must play the part of a character in order to experience the story?

The design possibilities are numerous, and keeping the design simple in the face of such a complex problem is tough. Remember to keep the story front and center, because you too are serving it. As a designer, you need to be interested and invested in the story structure. After they finished production on *Lost*, the Oculus Story Studio team posted a list of five lessons learned about HMD–VR storytelling [20], and they are

- Do not rush the pacing.
- Respect the ritual of "settling in and setting the scene."
- Let go of forcing the viewer to look somewhere.
- Be aware of spatial story density; there should be things happening everywhere.
- Simplify scope.

These observations are great big glowing clues and a gift for the HMD–VR environment designer. In Table 13.1 is a comparative study of these lessons and some suggestions about how the design can support them.

TABLE 13.1
Lessons Learned and Aspects of a Virtual Environment That Will Support Them

Lesson Learned	Aspects of a Virtual Environment That Will Support This Lesson
Do not rush the pacing.	Provide pathways, obstacles, overlapping walls, changing illumination, add elements that slowly appear, let each thing have a "moment."
Respect the ritual of "settling in and setting the scene."	Provide some place for the visitor to admire a "wide angle establishing shot" that feels safe and secure. Set up the "in," a character or device that leads the viewer into the scene slowly.
Let go of forcing the viewer to look somewhere.	This is "spherical" design, and that means much more design to develop in the HMD–VR environment. Consider the inside and underneath surfaces of what you are designing; they will be seen. These places may be useful for increasing spatial story density. Build your environment for exploration as well as storytelling.
Be aware of spatial story density; there should be things happening everywhere.	In addition to considering the inside and underneath surfaces of your environmental elements, you should think about how these elements affect each other. What is the effect of rain or wind on each one? What about the force of gravity and the balance on each thing in the environment? What visual and physical "side stories" are going on, and how do they support the main story?
Simplify scope.	Be aware of the limitations on rendering, and design your virtual environment for those parameters. Create only what you need to support the story you want to tell, and the message you want to send.

Now, let us consider the differences in the narrative structures of a stage play, a cinematic film, and a virtual/game-based environment. In Figure 13.2 notice where these forms of narrative structures overlap and where they diverge. There are design opportunities in these areas, and you can discover them by asking these questions:

- What kind of narrative structure (linear or nonlinear) is needed by this project?
- Is it fundamentally an experiential or narrative event?
- What sort of environment best supports the story I am going to tell?

These questions are undoubtedly on the minds of filmmakers who are interested in working with virtual reality. On December 4, 2014, the Sundance Institute announced the "films and installations" to be shown at the Sundance Film Festival in early 2015 [21]. There were 13 VR "New Frontier Installations" on the list, including *Perspective; Chapter 1: The Party, Way to Go,* and *Wild*. In Figure 13.3 are screen images from the first two films on this list.

13.3.4.1 Realistic Environment/Changing Point of View in *Perspective; Chapter 1: The Party*

In *Perspective*, Directors/Lead Artists Rose Troche and Morris May tell the story of date rape from two points of view: the young college woman who is the victim and the college man who rapes her. The environment is real, the actors are in a large house decorated for a college party, and the color pallet is subdued, almost monochromatic. This is photo real HMD–VR; you are right inside the actors' field of view, experiencing the same visual perspective that they do. A viewer's reaction to this kind of realistic scenario showing such a grim subject is typically serious; this is not intended as light-hearted entertainment. The production design of this kind of project is challenging, as Jamie Thalman describes in the Spotlight section later (Section 13.3.6).

13.3.4.2 Experiential-Narrative Hybrid Filmmaking in *Way to Go*

Way to Go, directed by Vincent Morisset; with music and sounds by Philippe Lambert; creative coding and technology by Édouard Lanctôt-Benoit; and animations, drawings, macro videos, and costumes by Caroline Robert, can best be described as a walk in the woods whilst dreaming of reality. The images of woodlands surround you, in black and white initially, as you guide your box-headed puppet down the forest path. With mouse clicks and keystrokes you can make him fly, jump, and tilt your view in all directions. The whole scene changes color and flattens into bill-boarding character generator (CG) trees at times, and into tinted photorealism at others. Periodically you are joined by a human being dressed like another box-head, all in black, with cartoon expressions drawn on his face area. A chase-dance ensues, and ultimately you reach the end of the trail and the experience is over.

13.3.4.3 Artificial Intelligence-Based Character Interaction in a *Wild* HMD–VR Spinoff

In 2014, Reese Witherspoon and Laura Dern starred in *Wild*, produced by Fox Searchlight Pictures and Pacific Standard. Later that year Felix Lajeunesse and Paul Raphaël (Felix and Paul Studios) created *Wild: The Experience* a new interactive HMD/VR app/movie to go with the Reese Witherspoon film. This "experience" (hopefully the industry will generate a better name) can be accessed via the selection available on the Gear VR applications, as well as nonvirtually on several YouTube pages. Essentially, this story starts with Reese Witherspoon entering a clearing wearing her hiking gear and taking a seat on a nearby rock. She starts to have a conversation with her dead mother, and when you look around the clearing to see who she is talking to: Laura Dern has appeared, seated on another rock. The experience is coded to make Laura Dern appear when you are looking in the other direction; you can never actually see her appear or disappear. This subtle effect is a harbinger of the whole new world of possibilities involving code driven/edited effects that can play with the viewer's area of focus in a HMD–VR environment.

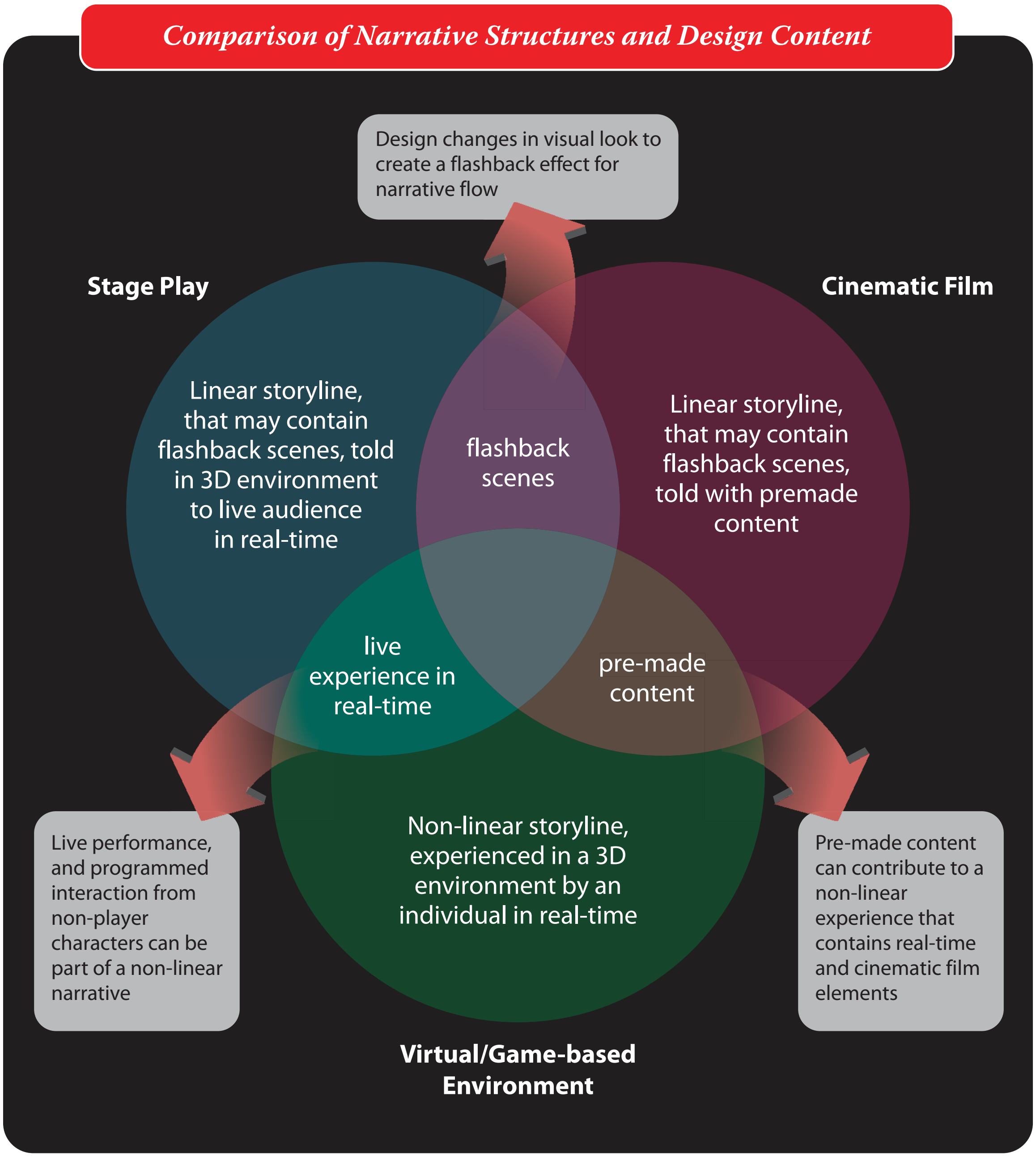

Comparison of Narrative Structures and Design Content
Design changes in visual look to create a flashback effect for narrative flow
Stage Play
Cinematic Film
Linear storyline, that may contain flashback scenes, told in 3D environment to live audience in real-time
flashback scenes
Linear storyline, that may contain flashback scenes, told with premade content
live experience in real-time
pre-made content
Non-linear storyline, experienced in a 3D environment by an individual in real-time
Live performance, and programmed interaction from non-player characters can be part of a non-linear narrative
Pre-made content can contribute to a non-linear experience that contains real-time and cinematic film elements
Virtual/Game-based Environment

FIGURE 13.2 Comparing the narrative structures of a stage play, a cinematic film, and a virtual/game-based environment.

Design Examples from Virtual Reality Film Projects

"Perspective; Chapter 1: The Party"
Director, Rose Troche; Tech Lead, Morris May; Production Designer, Jamie Thalman

Way to go © France Télévisions / National Film Board of Canada. All rights reserved

FIGURE 13.3 Images from VR films *Perception* and *Way to Go.*

13.3.5 Safety

Along with all the wonderful new aspects that the HMD–VR environment brings to design are some new safety, health, and accessibility issues for the virtual environment designer. Cybersickness, underdeveloped user interfaces that lack affordance, and inefficient 3D modeling all contribute to the potential for user discomfort and lessened safety. There are many challenges ahead for virtual reality display. Our eyes and brain work in a sophisticated collaboration between convergence (the overlapping images created by our two eyes on an object) and accommodation (the focusing of our eyes on an object) to locate and identify objects around us, and that can be disrupted by viewing a 3D environment that has been translated into 2D and projected on a screen close to our eyes [22].

13.3.5.1 Read the Manual and Be Aware of the Safety Guidelines

Many of the safety issues are described in the health and safety documents that come with VR HMDs. For instance, the online developer's guide issued by Oculus includes the following recommendations: "We recommend seeing a doctor before using the headset if you are pregnant, elderly, have pre-existing binocular vision abnormalities or psychiatric disorders, or suffer from a heart condition or other serious medical condition." Its other warnings include

> Some people (about 1 in 4000) may have severe dizziness, seizures, epileptic seizures or blackouts triggered by light flashes or patterns, and this may occur while they are watching TV, playing video games or experiencing virtual reality, even if they have never had a seizure or blackout before or have no history of seizures or epilepsy. Such seizures are more common in children and young people under the age of 20. Anyone who has had a seizure, loss of awareness, or other symptom linked to an epileptic condition should see a doctor before using the headset.
>
> This product should not be used by children under the age of 13. Adults should monitor children (age 13 and older) who are using or have used the Headset for any of the symptoms described below, and should limit the time children spend using the Headset and ensure they take breaks during use. Prolonged use should be avoided, as this could negatively impact hand-eye coordination, balance, and multi-tasking ability. Adults should monitor children closely during and after use of the headset for any decrease in these abilities.

Furthermore, Oculus recommends that these devices should only be used in a safe environment:

> The headset produces an immersive virtual reality experience that distracts you from and completely blocks your view of your actual surroundings. Always be aware of your surroundings when using the headset and remain seated at all times. Take special care to ensure that you are not near other people, objects, stairs, balconies, windows, furniture, or other items that you can bump into or knock down when using—or immediately after using—the headset. Do not handle sharp or otherwise dangerous objects while using the headset. Never wear the headset in situations that require attention, such as walking, bicycling, or driving. [23]

With serious warnings such as these, a responsible virtual environment designer needs to think about what they can add to increase the safety and enjoyment of the virtual experience.

13.3.5.2 Designing for Safety and Comfort: Some Beginning Steps

Current research has turned up some interesting results and these can inform your choices for design in a virtual environment. Here are three interesting aspects to design and how a human being uses the virtual environment.

- Using ramps instead of stairs within the virtual environment may reduce cybersickness and/or motion sickness because it lowers the amount of vection or the false sensation of movement that a visitor perceives when a large portion of their visual field moves around them [24].

- The design of cursors must accommodate how an object is perceived in the virtual environment, and 3D cursors that can encircle as well as point at an object are the most useful [25].
- Direct selection in a virtual environment is affected more by the visual perception of the object than by the motor movement direction of the person selecting the object. Some ways that may ameliorate this difficulty is to (1) bring items that must be selected right up into the main field of view, and (2) expand the size of the cursor into an ellipsoid shape with a larger selection field [26].

13.3.6 Spotlight on Jamie Thalman: A Q&A about Production Design for VR Films

Expanding the creative process is central to Jamie Thalman's design approach. As a production designer/editor he calls on his global perspective (BA in economics, diplomacy and world affairs from Occidental College, 2006–2010) and combines that insight with his training in spatial design and architecture (Making + Meaning Program at Southern California Institute of Architecture 2012) to create narrative digital and film content. He is a partner at Rustic Media (http://www.rustic.la) where he makes original and branded documentary projects, and a CAP (Creative Activist Program) Mentor for Creative Visions (http://www.creativevisions.org/). The production design for *Perspective; Chapter 1: The Party* was his first experience in designing an environment for a VR film. Let us see what he has to say about it.

Question: What were your chief considerations when designing the environments for *Perspective*?

Jamie Thalman: Perspective was my first time working with VR. My biggest consideration was about understanding the set of constraints and possibilities around the visual language of a story told exclusively from characters' points of view (POV). My first feature film project in 2010 followed a team of Sherpa as they climbed Mount Everest to remove dead bodies and rubbish from the top of the mountain. The GoPro Hero 2 had just become available and was a good fit for that project for a bunch of reasons. A majority of their story on the mountain was captured on head-mounted GoPros. While the Mount Everest project was very different than *Perspective*, I became familiar with live-action first-person POV as a visual language and storytelling framework. *Perspective* pushed my understanding of POV storytelling and helped us understand the challenges that might be unique to VR live-action filmmaking.

For the *Perspective* specifically, my biggest environmental design task was hitting a realistic three-hundred-and-sixty-degree world. *Perspective* provides viewers with an uncomfortable narrative about sexual assault that is all too real for too many people today. Instead of stylizing for a certain look or cinematic quality, Rose Troche (director) wanted to make sure it felt like a real experience for viewers. When Morris May (lead artist) cued up the test footage on the Oculus Rift, it became clear the cameras were capturing something like 270 degrees at any one time. With the handheld POV camera action, it turns into 360 degrees pretty quickly. We were shooting on-location in a house and had to dress for 360 degrees across the board. It ratchets up the design demands.

We were locked into a wide focal length, something wider than the human-eye equivalent to roughly 22 mm, which puts a few additional demands on production design. Alison Kelly (director of photography) didn't have the ability to light many of the scenes in ways she would for a conventional film project. Alison couldn't hide any cinema lights or stands from the wide angle field of view. We were in smaller spaces on-location in a house and had to light almost exclusively with practical lights (light sources that are endemic to the scene). We learned that practical light sources are more crucial than ever not just as light source motivations, but also primary light sources. Alison also lost the ability to backlight the characters, which helps create visual separation between them and the background. Being so wide all of the time, we didn't have the shallow depth of field of a longer focal length that helps create a cinematic look with background separation. It became even more crucial to create visual depth and separation through production design. While reflections are always a concern, they were especially hard to avoid because we shot wide the entire time. Also because blocking and character placement determined where the camera was, Alison couldn't cheat the camera position as much to avoid reflections. We also didn't have VTR (video tape record) or a monitor to reference the camera's frame. The cameras captured a distorted image that Morris May later wrapped for each eye of the Oculus. Based on blocking rehearsals, I would put myself in the character's POV; what that character saw was the general frame with which I had to work.

VR also changes how we approached design within each frame. We had to say goodbye to the 16:9 or anamorphic frame. I'm still trying to understand how you work to build a composition when the audience sets the frame with the orientation of their headset. My job became less about how to work within Alison's frame and more about how to work with the world in the character's path. Rose and Alison set the cameras on a course that followed a particular character's path, pacing, directionality, and vitality based on where they were in the story.

Q: How do you see VR-based films changing the job of the film production designer?

Thalman: Perspective helped me understand how design can be used to make better VR films. While the audience is charged with setting their own viewing frame, design can influence the frame they choose with environmental features, color cues, levels, horizon lines, lights, etc. I think it's an opportunity to explore choice architecture to help shape the viewer's experience and supplement the narrative. I was really interested in the sound capture for the *Perspective*; it was four-direction binaural recording at times. This has some interesting implications for production design. We can introduce elements into the world that motivate the soundscape and give the viewer a sense of directionality and spatial awareness. As POV's change in the story, changes in the directionality can impact the viewer's sense of place. The potential for creating a disorienting environment through the use of audio is also compelling.

Q: What would be your dream project in VR?

Thalman: I hope that when viewers experience different perspectives through VR it might help them create an emotional connection with the subject and build a greater capacity for compassion. My dream project would maybe be a live-action story told from perspectives we usually have a hard time experiencing. If I could tell the story of a family of elephants from the point of say a calf being cared for by her mother, a young adult being hunted by poachers or in captivity, and other moments that define the experiences of elephants.

13.4 EXPERIENCING THE VIRTUAL ENVIRONMENT WITH A REAL BODY

In the 1999 movie *The Matrix* (distributed by Warner Bros.), Morpheus explains virtual reality to Neo. Morpheus says, "What is real? How do you define 'real'? If you're talking about what you can feel, what you can smell, what you can taste and see, then 'real' is simply electrical signals interpreted by your brain."

Defining what we experience in a virtual environment is the first step toward designing effective virtual environments that enhance visitors' abilities to interact and understand the messages we wish to convey to them. After their initial encounter with a virtual space, having turned their heads for a good look around, many people begin to move about. They step side to side, or forward and back, making their first steps in a fantasy world that almost seems to exist around them. This is a critical moment. The believability of the virtual environment has much to do with our familiarity with real-world objects, and how we perceive scale, motion, and perspective in the real world. If the design of the virtual environment does not support this framework of perception, then the experience becomes invalid and the environment appears false to the visitor. This is not to say that a completely photorealistic environment is required to engage a visitor; many environments with stylized 3D have proven to be compelling experiences. What is necessary is to make it seem familiar to visitors, so that they can understand and anticipate how things function. The objects in the environment should be scaled to the proper size, exhibit the correct kinds of parallax (the overlapping movement of objects in our visual field) when we go past them, and move according to the physical laws of our real world. The environment needs to match up to the "internal model" we all know.

13.4.1 Design Challenges Specific to HMDs

Technologically speaking, HMD–VR exists at the intersection of video display, positional tracking parameters, and the rendering capacity of your computer's graphics system. Positional tracking on the HMD–VR headset is recorded and sent to the graphics engine. However, that data can be distorted by real-world environmental factors that interfere with the magnetometer and the infrared sensors. Further down the pipeline, the graphics rendering output is dependent on the graphics engine and graphics card you are using, as well as the viewer or game software you are trying to run. With so many sources of information to coordinate, render, and display on the screen in front of your eyes, it's no wonder that sometimes the smooth flow of a moving vision is interrupted by latency, drift, or judder. Why should you, the designer, care about latency, drift, and judder? Because the more efficiently and effectively you tailor your virtual environments to a HMD–VR display, the more these effects can be mitigated and a deeper sense of immersion can be achieved.

13.4.2 The Problem of Latency

Latency occurs when what we see in the HMD–VR environment does not match up to our human senses. As Michael Abrash said, "When it comes to VR and AR, latency is fundamental—if you don't have low enough latency, it's impossible to deliver good experiences, by which I mean virtual objects that your eyes and brain accept as real. By 'real,' I don't mean that you can't tell they're virtual by looking at them, but rather that your perception of them as part of the world as you move your eyes, head, and body is indistinguishable from your perception of real objects" [27]. If the latency of the VR system is too high, it can cause "simulator sickness," nausea, and headaches if not corrected. Two strategies for dealing with latency are (1) predictive tracking and (2) time warping (image shifting and image warping). In his keynote presentation to the Open Simulator Community Conference in 2014, "Virtual Reality: How Real Should It Be?" Steve LaValle, the head of research and development for Oculus VR discusses the surprising results of these tracking strategies [13]. He noted that if a bit of predictive tracking and time warping image modification is applied to the images sent out to the right and left eyes of the display screen it will smooth the appearance of movement. (There is more information about Timewarping [28] in the end references.) LaValle also mentioned that smoothing out the tracking prediction positions introduces latency into the results while really accurate prediction introduces jitter into the image. He said that a hybrid solution, called "perceptually tuned filtering," is being applied to VR systems. This filtering blends predictive tracking along a greater time span of movement when the viewer's head moves were fast and smoothed the tracking data when head moves were slow, adding a transitional effect between them.

13.4.3 When Objects and Settings Start to Drift

Drift occurs when the VR scene you are viewing starts to be displaced in its orientation to your head position, and each successive head move increases the mismatch. As LaValle informs us, the problem of drift comes from misbehaving sensors. He notes that you cannot use just numerical gyroscope data or the virtual world starts to tilt, and if you use the accelerometer data to correct for this problem you have to time it carefully.

If the correction is made too quickly, the viewer will see it and start to get cybersick, and if the correction is made too slowly the display will start to drift. He said that the faster you turn your head, the faster the errors will accumulate from MEMS sensors, and the faster the drift rates will grow. He and his team at Oculus have been working on algorithms to compensate for these errors [13].

13.4.4 The Judder Problem

Judder happens when the frame rate being generated by the HMD–VR system is not matching up to our eye movements. This undesirable effect can appear as double vision, vibration, and stuttering in the display image. As Dr. Michael Abrash defines it, "eye movement relative to a head-mounted display can produce a perceptual effect called judder, a mix of smearing and strobing that can significantly reduce visual quality. The straightforward way to reduce judder is to make displays more like reality, and the obvious way to do that is to increase frame rate" [29]. The quickest way increase frame rate and to reduce judder, is to design and create the most optimized and geometrically efficient scene possible.

13.5 PROJECT: DEVELOPING A HMD–VR NARRATIVE ENVIRONMENT FOR STORYTELLING

In this project, you will utilize one part of the virtual environment you have been constructing throughout this book to set up a narrative environment for telling a story visually and audibly. This environment will need to be designed especially to accommodate the limited keyboard and mouse accessibility that using a HMD–VR currently requires of your visitors. Some potential spaces on this game-based sim that you can utilize for this project are

- Castle spaces, such as the dungeon, main hall, second-floor rooms (book room and caretaker's room), and Rooftop
- Village spaces such as the Mousehole Inn, and the "Mayor's House" (on the northeast end of the street), the buttercross, and the quay
- The fairy environs, such as the fairy island in the middle of the lake, the fairy (fae) bridge, and the standing stones

13.5.1 Step 1: Finding a Concept to "Hang Your Design Hat On"

Find a short poem that inspires you and can be used to evoke a story idea for one of the spaces contained within the virtual environment you have created with the projects of this book. For example:

When the coal
Gave out, we began
Burning the books, one by one;
First the set
Of Bulwer-Lytton
And then the Walter Scott.
They gave a lot of warmth.
Toward the end, in
February, flames
Consumed the Greek

Tragedians and Baudelaire,
Proust, Robert Burton
And the Po-Chu-i. Ice
Thickened on the sills.
More for the sake of the cat,
We said, than for ourselves,
Who huddled, shivering,
Against the stove
All winter long.

Weldon Kees, “The End of the Library”

13.5.2 Step 2: Choosing a Location for Your Storytelling

This step and the previous one can be done simultaneously. If you have a strong feeling about the kind of story you would like to tell, then choose a place in the game-based sim and find a poem to encapsulate the mood and ambiance of the story you have in mind. You might be inspired by the locations we have provided, or by some of the props included. You may also decide to take an empty space and decorate it for telling the story you have in mind. For this example project, the book room on the second floor of the castle and the “The End of the Library” poem seemed to be a good fit in style and tone.

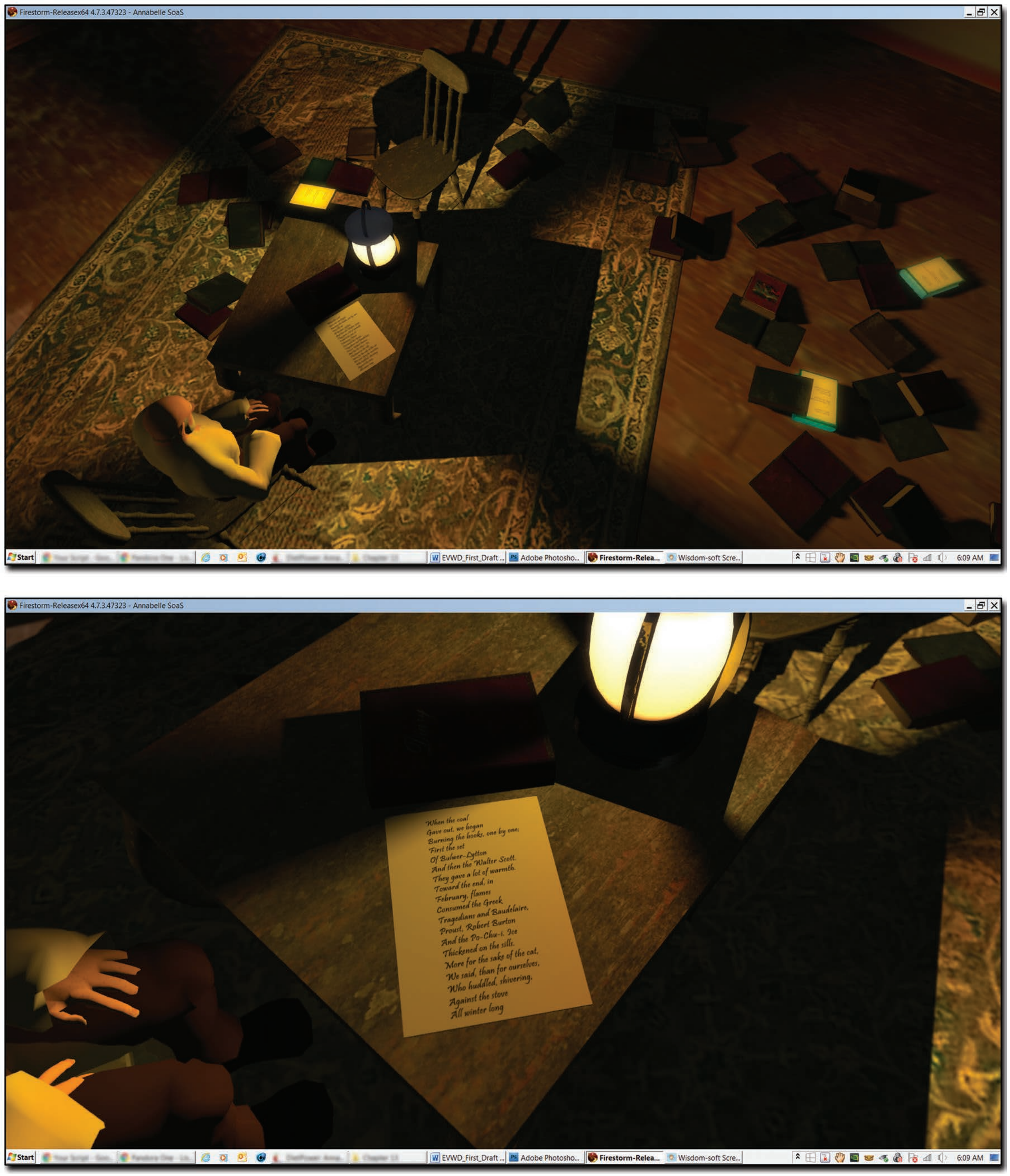
13.5.3 Step 3: Adding in the Five Dimensions of Design

Remember the “five dimensions of design” that were introduced in Section 13.3? You are going to apply them in this project. Using the following list, take some time to write down your own ideas about how you will use each of these elements.

- Agency—Define how you will use this quality/element.
- Presence—Define how you will use this quality/element.
- Affordance—Define how you will design for this capacity.
- Story—Write a one-page story that includes the ambience of the poem you have chosen for your project. Bring in other themes that relate to events and characters you can imagine for the rest of the sim. For example, in this project the ambience of desperation and dejection being experienced by the caretaker in the book room is counterbalanced by the books themselves showing images and text referring to a fantasy environment that surrounds the sim.
- Safety—Define how you will include safety/accessibility in your narrative environment.

13.5.4 Step 4: Adding Agency with a “Character” Avatar

Create another avatar, either a non-player character (created by using an NPC creating script) or simply a redress of your own avatar. This will be the element that provides agency for the visitor, allowing them to understand the nature of the room and “enter” the narrative. For example, we created an avatar named “Caretaker,” and set him up at the table in the book room. The caretaker avatar is wearing an invisible, proximity-sensing scripted object that plays a sound clip containing a recitation of the poem we chose in Section 13.5.1, “The End of the Library” by Weldon Kees. This poem is also displayed as a written page on the table in front of the caretaker, creating a supportive, visible version of the audible poetry clip. If you desire, this visible page object can be scripted to magically appear on the table when a visitor approaches. You can see this setup in Figure 13.4.



When the coal
Gave out, we began
Burning the books, one by one;
First the set
Of Bulwer-Lytton
And then the Walter Scott.
They gave a lot of warmth.
Toward the end, in
February, flames
Consumed the Greek
Tragedians and Baudelaire,
Proust, Robert Burton
And the Po-Chu-i. Ice
Thickened on the sills.
More for the sake of the cat,
We said, than for ourselves,
Who huddled, shivering,
Against the stove
All winter long

FIGURE 13.4 Agency is added to the environment by surrounding the Caretaker avatar with visible and audible story components.

13.5.5 Step 5: Adding Presence—The Sound of the Environment

Now you can embellish the presence of the environment by adding scripted sound-generating objects. (Note: You could also feed the sounds in via a Media Texture if you want a longer sound track.) Imagine that you are sculpting the sound of the space to create another dimension of experience for the visitor. You might hear the sounds of the wind scratching around the casements, the mutterings of the caretaker, the crackles and pops of the fireplace, ethereal music, and other sonic effects. Create five to ten sources for these sounds and put them into the environment, in locations that add to the sense of presence.

13.5.6 Step 6: Adding Affordance—Storytelling Elements That Are Easy to Use

In this particular environment, there are many objects that provide obvious clues as to their affordance. Most of us use books for information, so they should be an obvious source for clues as to what story this environment wants to tell us. The books lying all around the floor, besides providing fuel to warm the caretaker, can also provide the visitor with words and images from the story you want to tell. There are many ways to make books "talk" to a visitor. The most direct way is to have them simply display a new graphic/texture on their surfaces or rezz an image when the visitor approaches, as demonstrated in Figure 13.5.

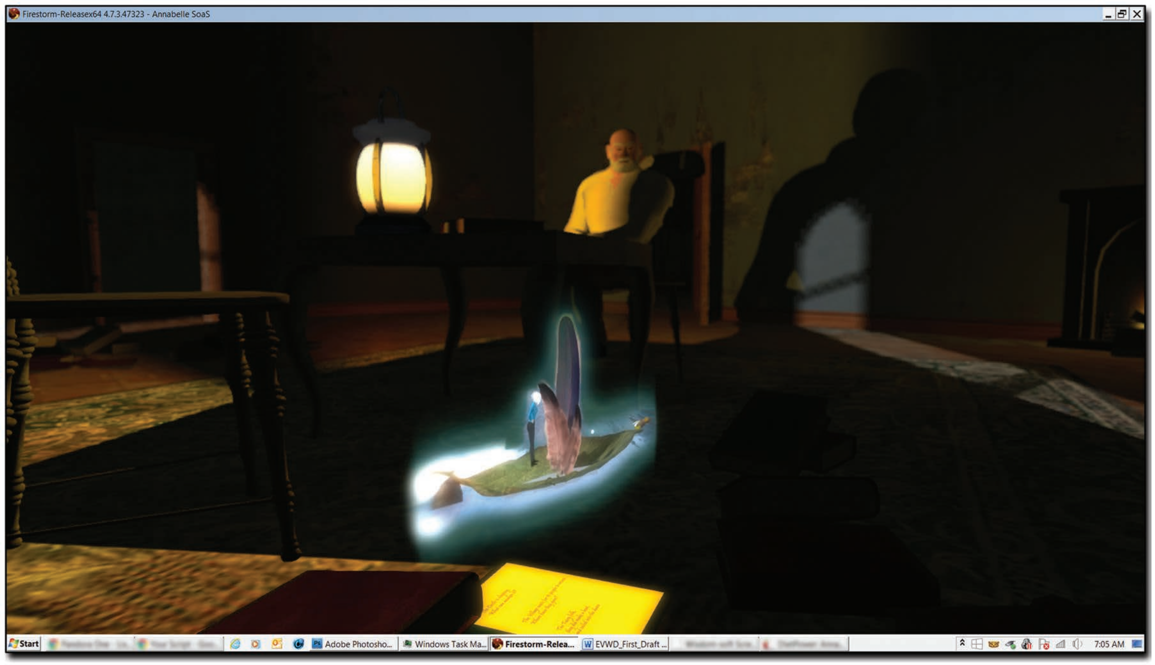


FIGURE 13.5 Showing affordance on interactive objects.

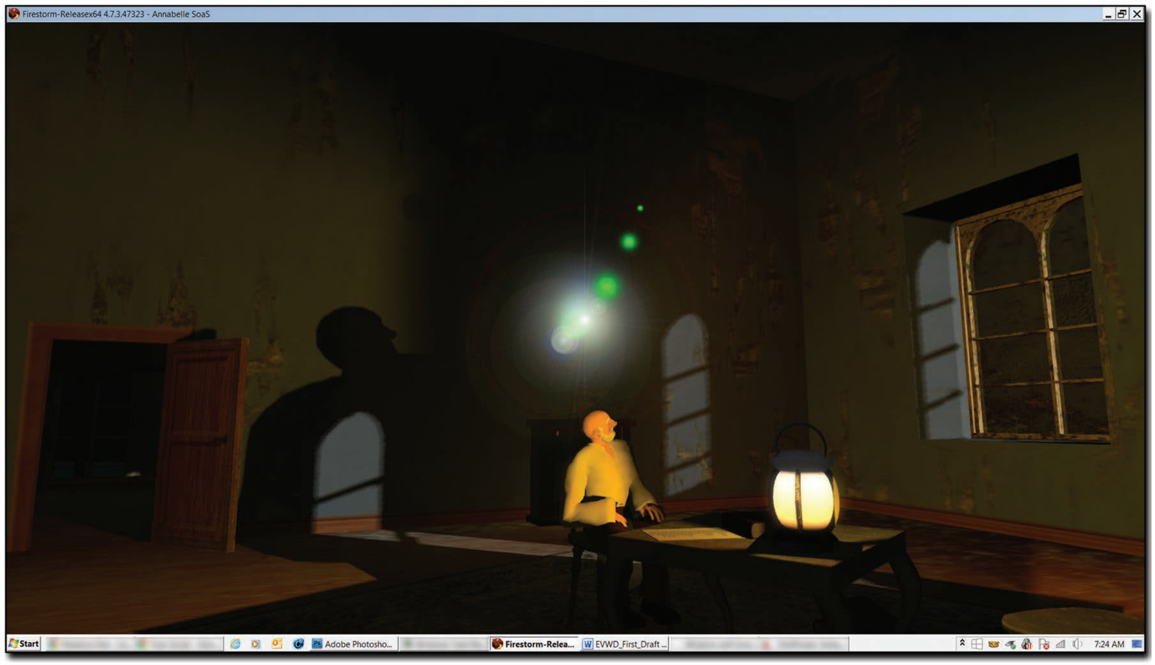


FIGURE 13.6 Caretaker's will-o'-the-wisp pet follower.

13.5.7 Step 7: Adding Story with Additional Characters

Now you can deepen the story of the environment by adding in subcharacters to your environment. Perhaps some of the other books can talk with audio clips. For instance, when the visitor is in proximity to them, they might say something like "Please don't burn me!" and "I'm a good read!" Perhaps the caretaker has a "familiar," a pet, or spiritual guide. This can be made with many sorts of scripts, such as "pet follower" or "NPC character," and are available online in script libraries such as http://www.outworldz.com/. A "sensorless pet follower" script was utilized to create a will-o'-the-wisp character that hovers above the caretaker, as seen in Figure 13.6. This subcharacter could be scripted to emit sound bite comments in a tiny whisper.

13.5.8 Step 8: Adding Safety and Accessibility to VR Narrative Environment

It is good practice to make every interactive object in your storytelling environment use visible and audible components. By using avatar proximity sensing in your interactive scripts, you will take the onus of using the mouse and keyboard off the visitor and leave them free to focus on moving around the room and observing what they experience. Also make sure that any pitfalls, tricky corners, and other physically challenging parts of your environment are made safe for all avatars to navigate. You can utilize invisible prims to keep the visitor within "safe" areas.

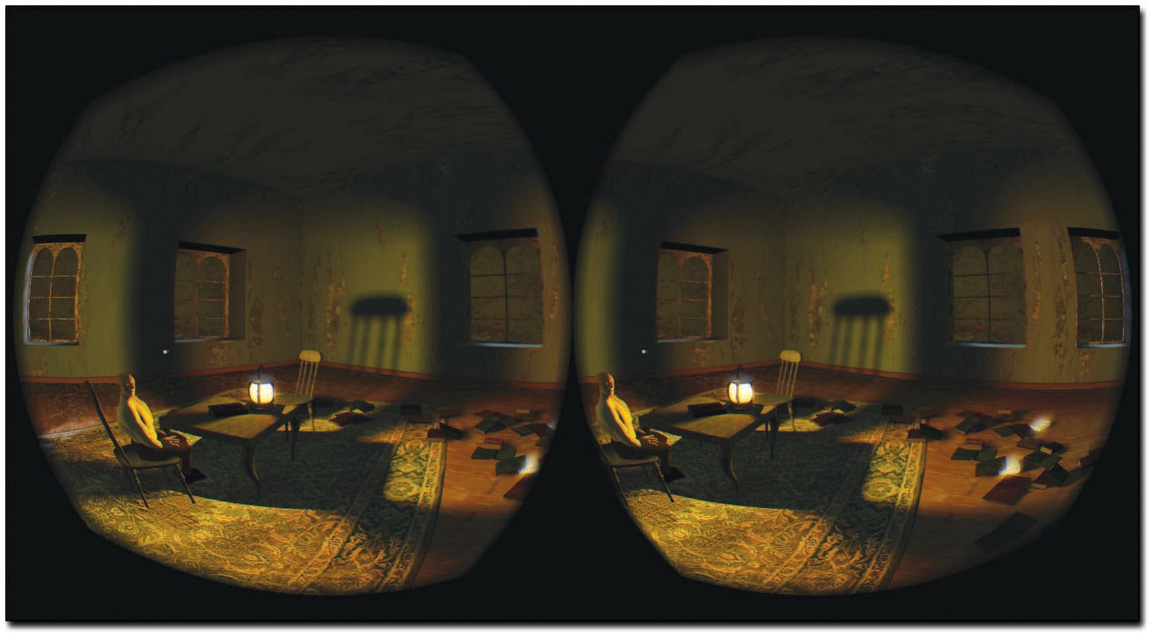


FIGURE 13.7 Narrative environment shown in HMD (Oculus Rift, Dev Kit 1).

13.5.9 Step 9: Keep Adding Depth to the Experience

We have just started to understand how virtual environments can help us tell stories, and the platforms available will continue to improve in quality and ease of use. As you work your way through this book, come back to this project from time to time and add more details to the environment, embellishing its overall capacity to support the story you would like it to tell the visitor. As you can see in Figure 13.7, the storytelling experience is very different when viewed through the Oculus Rift interface.

13.6 CONCLUSION: BUILDING THE LAST PLATFORM?

There are many big picture questions for virtual environment designers to consider when they design for HMDs. Here are a few of them:

- Is it possible for an HMD–VR experience to include a narrative that results in the personal transformation or enlightenment of a visitor who experiences it?
- Would an HMD–VR environment be able to contain the spirit of a magnum opus, a great work that defines the genre?
- Will it be a new art form, providing us with a new way of reframing mental and emotional experiences?

In his address to Carnegie Mellon University's Robotics Institute, Michael Abrash, Oculus chief scientist, calls this "the last platform we will build" [30]. Perhaps he is right.

REFERENCES

1. Wikipedia, s.v. “Allegory of the Cave,” accessed April 20, 2015, http://en.wikipedia.org/w/index.php?title=Allegory_of_the_Cave&oldid=656282727.
2. Charles Wheatstone, “Contributions to the Physiology of Vision.—Part the First. On some remarkable, and hitherto unobserved, Phenomena of Binocular Vision,” June 21, 1838, accessed August 12, 2015, http://www.stereoscopy.com/library/wheatstone-paper1838.html.
3. *Wikipedia*, s.v. “Stereoscope,” accessed August 13, 2015, https://en.wikipedia.org/w/index.php?title=Stereoscope&oldid=655042748.
4. Scott Highton, “The Future of VR—Is This All There Is?,” *Virtual Reality Photography*, July 2015, accessed August 1, 2015, http://www.vrphotography.com/futureofvr/.
5. “Documentation,” *Developers, Oculus VR*, accessed April 14, 2015, https://developer.oculus.com/documentation/.
6. “Documentation Archives” *Developers, Oculus VR*, accessed April 14, 2015, https://developer.oculus.com/documentation/archive/.
7. “Oculus Rift Development Kit 2 Teardown,” *IFixit*, August 2014, accessed April 15, 2015, https://www.ifixit.com/Teardown/Oculus+Rift+Development+Kit+2+Teardown/27613.
8. Atman Binstock, “Powering the Rift,” *Oculus Blog*, May 15, 2015, accessed July 23, 2015, https://www.oculus.com/en-us/blog/powering-the-rift/.
9. Aaron Tilley, “Stanford and Nvidia Team Up on More Realistic ‘Light Field’ VR Headset,” *Forbes*, August 6, 2015, accessed August 6, 2015, http://www.forbes.com/sites/aarontilley/2015/08/06/stanford-and-nvidia-team-up-on-more-realistic-light-field-vr-headset/.
10. David Gelles and Michael J. De La Merced, “Google Invests Heavily in Magic Leap’s Effort to Blend Illusion and Reality,” *New York Times*, October 21, 2014, accessed August 17, 2015, http://nyti.ms/1yVQKzQ.
11. Jessi Hemple, “Project Hololens: Our Exclusive Hands-on With Microsoft’s Holographic Goggles,” *Wired*, January 21, 2015, accessed August 17, 2015, http://www.wired.com/2015/01/microsoft-hands-on/.
12. John Carmack, “GDC Talk,” YouTube video, 1:32:47, posted by Ruthalas Menovich, March 4, 2015, accessed August 1, 2015, https://www.youtube.com/watch?v=UNAmAxT7-qs.
13. Steve LaValle, “OSCC14—Keynote Speaker: Steve LaValle—Virtual Reality: How Real Should It Be?,” YouTube video, 1:28:27, posted by AvaCon Inc., December 5, 2014, accessed April 12, 2015, https://www.youtube.com/watch?v=L5JKsQ_w_0A.
14. John Volpe, “Oculus Story Studio Is the Pixar of Virtual Reality,” *Engadget*, January 27, 2015, accessed August 1, 2015, http://www.engadget.com/2015/01/27/oculus-story-studio-is-the-pixar-of-virtual-reality/.
15. John Volpe, “‘Henry’ is Oculus’ First Emotional Step to Making AI Characters,” *Engadget*, September 29, 2015, accessed August 12, 2015, http://www.engadget.com/2015/07/29/henry-oculus-story-studio-vr/.
16. “Henry’s Premiere,” Vimeo video, 2:38, posted by Story Studio, July 28, 2015, accessed August 19, 2015, https://vimeo.com/134754691.
17. Colin Ware, *Information Visualization: Perception for Design* (Morgan Kaufman, 2013), 47.
18. Jeff Johnson, *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Guidelines* (Morgan Kaufman, 2014), 34.
19. Janet H. Murray, *Hamlet on the Holodeck* (MIT Press, 1997), 217–272.
20. Saschka Unseld, “5 Lessons Learned While Making Lost,” *Oculus Story Studio Blog*, July 15, 2015, accessed August 1, 2015, https://storystudio.oculus.com/en-us/blog/5-lessons-learned-while-making-lost/.
21. “Sundance Institute Announces Spotlight, Park City at Midnight, New Frontier for 2015 Sundance Film Festival,” *Sundance Institute* (press release), December 4, 2014, accessed August 12, 2015, http://www.sundance.org/blogs/news/spotlight-midnight-and-new-frontier-films-announced-for-2015-festival.
22. Sarah Zhang, “The Obscure Neuroscience Problem That’s Plaguing VR,” *Wired*, August 11, 2015, accessed August 12, 2015, http://www.wired.com/2015/08/obscure-neuroscience-problem-thats-plaguing-vr/.
23. “Oculus: Health and Safety,” accessed August 13, 2015, http://static.oculus.com/documents/health-and-safety-warnings.pdf.
24. Jose L. Dorado and Pablo A. Figueroa, “Ramps are Better than Stairs to Reduce Cybersickness in Applications Based on an HMD and a Gamepad” (paper presented at the IEEE Symposium on 3D User Interfaces 2014, Minneapolis, Minnesota, March 29–30, 2014).

25. Leïla Schemali and Elmar Eisemann, "Design and Evaluation of Mouse Cursors in a Stereoscopic Desktop Environment" (paper presented at the IEEE Symposium on 3D User Interfaces 2014, Minneapolis, Minnesota, March 29–30, 2014).
26. Paul Lubos, Gerd Bruder, and Frank Steinicke, "Analysis of Direct Selection in Head-Mounted Display" (paper presented at the IEEE Symposium on 3D User Interfaces 2014, Minneapolis, Minnesota, March 29–30, 2014).
27. Michael Abrash, "Latency—the sine qua non of AR and VR," *Ramblings in Valve Time* (blog), December 29, 2012, accessed April 18, 2015, http://blogs.valvesoftware.com/abrash/latency-the-sine-qua-non-of-ar-and-vr/.
28. Ross.admin, "What is Timewarp? Should I Use It?," *Oculus*, December 22, 2014, accessed August 4, 2015, https://answers.oculus.com/questions/32/what-is-timewarp-should-i-use-it.html.
29. Michael Abrash, "Down the VR Rabbit Hole: Fixing Judder," *Ramblings in Valve Time* (blog), July 26, 2013, accessed April 17, 2015, http://blogs.valvesoftware.com/abrash/down-the-vr-rabbit-hole-fixing-judder/.
30. Michael Abrash and Dov Katz, "Why Virtual Reality Isn't (Just) the Next Big Platform: Michael Abrash & Dov Katz of Oculus VR," YouTube video, 58:13, posted by cmurobotics, May 2, 2014, accessed on July 23, 2015, https://youtube/dxbh-TM5yNc.

This page intentionally left blank

14 The Multiple Dimensions of Game-Based Virtual Environments

To me games have an extremely great and still unrealized potential to influence man. I want to bring joy and excitement to people's lives in my games, while at the same time communicate aspects of this journey of life we are all going through. Games have a larger potential for this than linear movies or any other form of media.

Philip Price

14.1 OVERVIEW ON MULTIPLE DIMENSIONS OF GAME-BASED VIRTUAL ENVIRONMENTS

The future of virtual reality (VR) and virtual worlds looks bright. Head-mounted displays, 3D scanning software, and new haptic technologies are developing rapidly for use in web-based virtual worlds and virtual gaming environments. As the capacity to present us with full-blown immersive 3D increases with these devices, the design of a virtual environment is even more important. A designer of virtual environments frames the focus and creates the scaffolding for interaction and presence-based immersion. The design also provides the visitor with a means for understanding the underlying message in these spaces. The key word is "message." A 3D virtual space without a message or story to tell is just another 3D tour. Perhaps the visuals are stunning, and it may take you to places you cannot go otherwise like deep space, but at the end of the tour, what have you learned? What emotions did you feel? How did the virtual reality experience add an understanding of yourself or your reality? Obviously, these are tough questions to answer. Jaron Lanier, a noted computer scientist and a pioneer in the development of virtual reality said,

> Now of course, Virtual Reality only gives us a temporary limitlessness. We still live in our physical bodies and we're still mortal. It might highlight our mortality to a degree that will make it harder to ignore than it is currently. People imagine Virtual Reality as being an escapist thing where people will be ever more removed from the real world and ever more insensitive. I think it's exactly the opposite; it will make us intensely aware of what it is to be human in the physical world, which we take for granted now because we're so immersed in it. [1]

With the words from Lanier in mind, ask yourself this question. How can I, the designer of virtual worlds, leverage the effects of virtual reality to impact our real-world perspective? For instance, suppose you wanted to create a virtual environment to teach students about the relative physical scale of life on earth. Now as you design, work backward from the emotional and intellectual state you would like the students to be in when they exit the virtual environment. If you want them to experience a state of heightened awareness regarding their human scale as it compares to an ant or whale, your virtual environment needs to demonstrate this. You may choose to just show them an ant and a blue whale in actual 1:1 scale, but that may not make any more impact than taking them to see ant and whale displays in a museum. Let us suppose you choose to demonstrate the scale of these animals by changing the scale of a familiar object, something the students use every day, their desks. Imagine that the students enter the virtual world, and see that their avatars are sitting at a desk that looks just like the real one they are sitting in for the demonstration. Then slowly, the desk begins to enlarge in scale, so that their avatars' bodies now are the same size as an ant standing on the seat of the desk. They can look at each other, maybe some of them grew antennas or developed ant-like mandibles, there may even be some real ants roaming around with them across the vast plane of the desk chair seat. At some point in the demonstration, the environment begins to fill with water and their avatar bodies begin to morph into aquatic forms. The familiar desk chair shrinks as they begin to experience what it is like to be an enormous marine mammal. The class has become a pod of blue whales. All of them are so large it is difficult to see the tails on some of them, and way down below is a tiny student desk resting on the bottom. After a recreational play period with their new aquatic forms, the students once more morph back into humans, and emerge from the virtual environment sitting at their real desks. I believe that it is unlikely that many of those students will ever look at an ant or a whale in quite the same way, or accept the scale of their human bodies as the standard measure for building all environments. As you may recall from Chapter 3, Section 3.4, semiotics and the use of symbols in design is a significant part of our virtual world experience. What kinds of text, graphics, verbal commentary, visual media, and other sorts of signs and symbols could be used in the ant–whale demonstration to create purpose, meaning, and mood? The term *game-based* has been used throughout this book to describe a virtual environment that presents the visitor with the opportunity for playing a game or interacting with other visitors within the context of a set of guidelines or rules. How can the ant–whale environment be developed into a game-based sim? In this chapter we shall discuss some aspects of how game-based virtual environments might be developed. Here are some of the key concepts:

- A virtual environment design for a game-based sim is more than storytelling; it creates a visual context, an interactive space, and a sandbox for the imagination.
- Game-based virtual environments can evolve from real-world-based games and events like game jams and gaming competitions.
- Virtual worlds can be used to prototype many kinds of real-world games, and real-world games can inspire game-based virtual environments.

14.2 DEVELOPING GAME-BASED ENVIRONMENTS FOR THE FUTURE PLAYER

To design engaging game-based virtual environments for future players, you must design for the intertwining relationship of technology, society, and games. Every technological advance has provided opportunities for new kinds of games to enter into our social experience. However, we have not abandoned our ancient games; chess or backgammon games are still widely popular activities, although now you can play against artificial intelligence (AI) in your phone when you lack a human opponent. To develop the design of a compelling game-based virtual space, you will need to understand the foundations of game development. Let us take a brief look at three of the most important factors: virtual reality technology, society-based influences, and our human need to play games.

14.2.1 Virtual Reality Technology

We have been developing virtual reality technology since the late 1950s. Now in the second decade of the 21st century, we are witnessing the inexorable convergence of virtual-reality-based hardware and software technologies with the social Metaverse. Like spectators at a prolonged stakes race, we watch as companies like Oculus Rift (owned by Facebook), the Valve Corporation (creators of the SteamVR games platform), and Samsung Electronics (creators of the Gear VR) compete to create the most popular consumer grade virtual reality interface. It is interesting to note that each of these companies has developed their virtual reality systems with a key component of publically accessible virtual reality: Oculus Rift began with head-mounted display hardware, Valve created a strong gaming platform, and Samsung is deeply tied into mobile phone technology. Who will win the race and develop the standard platform? That question may not be answered until we decide what to use virtual reality for. Many game players would like to have the "holodeck" experience, but what does that actually mean? In the television series *Star Trek: The Next Generation* (1987–1991 on CBS), the holodeck was used as a screenplay plot device, and in *Star Trek: Voyager* (1995–2001 on CBS) holographic characters like the Doctor added the aspect of AI and virtual characters into the human cast. As much as we love to tell stories, and hear them told to us, we do not live in a television series; our storylines do not resolve in an hour. We do not need to hold controllers in our hands to manipulate things in our real environment, unless we are using a tool. As you design virtual environments, these are but two aspects of the virtual space that will need special consideration. How would you make the environment contain an endless story? How would you design content to be manipulated by our hands or our voices?

14.2.2 Society

Let us move over a level and think about the social aspects of future virtual reality design. Dr. Johan Huizinga said, "Now in myth and ritual the great instinctive forces of civilized life have their origin: law and order, commerce and profit, craft and art, poetry, wisdom and science. All are rooted in the primeval soil of play" [2]. Society is the crucible within which we test the strength of our ideas. Our social media culture reflects our world culture in its diversity, and so should our virtual environments. Look at myths and rituals and let them inform the virtual world design of your next project. In his book *How to Do Things with Videogames*, Ian Bogost asks, "What if we allowed that videogames have many possible goals and purposes, each of which couples with many possible aesthetics and designs to create many possible player experiences, none of which bears any necessary relationship to the commercial videogame industry as we currently know it. The more things games can do, the more the general public will become accepting of, and interested in, the medium in general" [3].

The same follows for virtual environments and their potential. It is best to design for the diversity of society. However, it is also wise to realize that society needs to keep developing the legal systems of our virtual spaces. As Dr. Edward Castronova says in "The Right to Play":

> With synthetic worlds, society seems to have begun an exploration of the dimension of significance that may be attributed to a game. For every player who is content to view the synthetic world as a game, there is another who gleefully buys and sells the game's wands, armor, and gold pieces for U.S. currency on eBay. For every player who does not care if the synthetic world is hacked and accounts are robbed, there is another who views the breach as a computer crime of the highest order. For every player who sleeps soundly after being banished from a guild, there is another who thinks about committing suicide. This broad spectrum of significance and the ensuing emotional reaction that people manifest in synthetic worlds provide an incentive for the state to regulate and prosecute virtual crimes. [4]

14.2.3 The Basic Human Need to Play

We share our understanding of play and the need to do it with a wide variety of creatures in the world. Our pets bring us toys and entreat us to play a game with them, inviting us with physical gestures, poses, and vocalizations. Even animals of different species will play together, the evidence posted daily on YouTube. Play behavior is universal and a vital factor in our lives. In fact, Dr. Stuart Brown, who has made a lifelong study of the importance of play says, "The truth is that play seems to be one of the most advanced methods nature has invented to allow a complex brain to create itself" [5]. If you agree with the preceding statement, then you are a designer who understands that virtual world design comes with great responsibility. You can design an environment that helps people, young and old, learn new, creative cognitive skills through play, and that is a goal worth striving for, especially if designing these environments is "play" for you. In the next section we will survey various kinds of games found in the real world that can be adapted to virtual environments.

14.2.4 Finding Inspiration for Virtual Games

The world is full of games, and many of them can be translated into interesting virtual environments. For instance, when the Alchemy Sims game design team members wanted to playtest their real-world board game called *Tribe*, they set up a huge virtual game board on the sim and played with their avatars as positional markers. Instead of reaching down with their hands to move a plastic piece across a cardboard surface, they induced their avatar to walk to the proper position in turn. There is an overview of their setup in Figure 14.1, as well as a sampling of preliminary designs that led to the version that was playtested in a virtual world.

In Table 14.1 is a comparative list of real-world game types with their components and how they might be reimagined in a virtual environment. Notice how in some cases, the chess and tic-tac-toe games for instance, that there is almost a direct equivalent in the virtual space. However, in other examples, such as the rock-paper-scissors game, creating a game like that gives you the opportunity to be virtually literal in your interpretation of how these items are presented. While we would use hand gestures in the real world to play this game, in a virtual environment you can bring out rock boulders that can squash scissors or a sheet of virtual paper that explodes into thousands of particles when touched by the scissors. Develop a mindset that takes advantage of the surreal and comic effects of the virtual environment as you develop real-world scenarios into virtual game-based design.

The most gratifying experiences I have while in a virtual world happen when emergent play develops with my build group. Sometimes while we are hard at work constructing the contents for a sim, we discover a quirk or anomaly in some 3D geometry or LSL script that amuses us. This is the only catalyst we need. From these anomalies, we have invented games like *Prim Tennis*, a contest to see how much we could distort the shape of a simple primitive as we pass an object back and forth. Points are awarded for aesthetically pleasing forms and interesting scripted modifications. A special favorite of mine involves prim explosions, which is always entertaining. This game utilizes the special attributes of inworld objects and the physics engine. The process involves building a large structure with many primitives tightly packed together. When completed, the entire structure is selected and set to be a physical object. As the simulator's physics engine sorts out the positions and gravitational effects on each prim, the entire structure will disassemble in a spectacular fashion. Points are given for the most entertaining "explosion." Of course, our avatars have often become part of the games we made. At one point we had a giant slingshot that would launch an avatar like a physics ragdoll for some great distance out into the sea. Points were given for the closest landing to our floating targets.

14.2.5 Skill-Based Games and Gambling

If you are going to devote your game-based sim building skills to creating games in Second Life, then you should know about Linden Lab's policy toward skill-based gaming and gambling. Linden Lab has provided an overview of its policies at http://wiki.secondlife.com/wiki/Linden_Lab_Official:Second_Life_Skill_Gaming_Approved_Participants#Program_Overview.

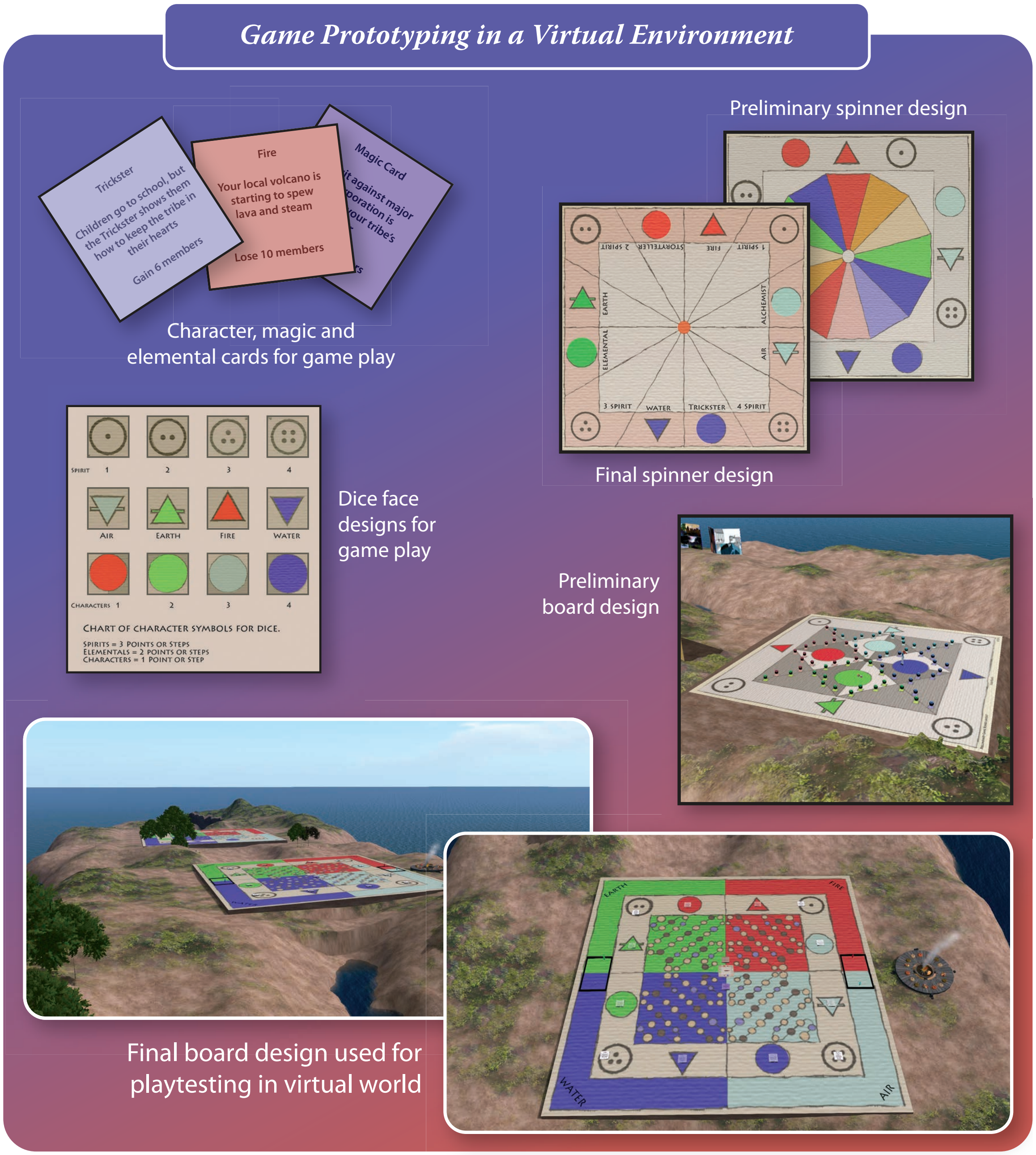

Game Prototyping in a Virtual Environment
Trickster
Children go to school, but the Trickster shows them how to keep the tribe in their hearts
Gain 6 members
Fire
Your local volcano is starting to spew lava and steam
Lose 10 members
Magic Card
Character, magic and elemental cards for game play
Preliminary spinner design
Final spinner design
Spirit 1 2 3 4
Air Earth Fire Water
Characters 1 2 3 4
Chart of character symbols for dice.
Spirits = 3 points or steps
Elementals = 2 points or steps
Characters = 1 point or step
Dice face designs for game play
Preliminary board design
Final board design used for playtesting in virtual world
Earth
Fire
Water
Air

FIGURE 14.1 Developmental components used for prototyping and playtesting the board game *Tribe*.

TABLE 14.1
Real-World Games and Their Translation into a Virtual Environment

Real-World Games and Their Components		Virtual Environment Version	
Scavenger hunt	List of items to find and surrounding territory to find it in	Virtual hunt	List of items to collect and clues to the locations in a virtual world
Puzzle hunt	Collection of puzzles to solve and territory to solve them in	Virtual puzzle hunt	Collection of puzzles and clues to their solutions in a virtual world
Building challenges (speed building or inventive approach)	Theme for building something, collection of everyday objects and a timer	Virtual building challenge	Theme for building, standard primitive objects, and a timer
Chess	Chess set	Virtual chess	Virtual chess set
Tic-tac-toe	Game board and markers	Virtual tic-tac-toe	Virtual game board and markers
Rock-paper-scissors	All you need is your hands	Virtual rock-paper-scissors	Virtual representations of a rock, a sheet of paper, and a pair of scissors that can be rezzed from the avatars inventory onto the ground in front of them while they play
Economic chance games like *Monopoly* or *Life*	Game board, markers, cards, dice and game pieces	Virtual version	Virtual 3D representations of the game board and game pieces, scripted dice, and notecard givers
Dice-based board games like backgammon	Game board, game pieces, and dice	Virtual version	Virtual 3D representations of the game board, game pieces, and scripted dice
Information quiz games like *Trivial Pursuit*	Game cards with question-and-answer book, scoring system	Virtual version	Virtual notecard giver, answer book, and scoring system

14.3 SPOTLIGHT ON JACQUELYN FORD MORIE—ART, VIRTUAL REALITY, AND EDUCATION

In this spotlight, we are talking with Jacquelyn Ford Morie, also known in Second Life as ChingALing Bling (and China Bling, her ersatz SL twin). Morie is founder and chief technology officer of All These Worlds, LLC, whose clients include NASA and the Army Medical Command. She was an early innovator in virtual reality and has been using virtual worlds, the social form of VR, since 2004. From 1999 to 2013 she was a senior research scientist at the University of Southern California Institute for Creative Technologies (ICT), which she helped found. She also developed artistic and technical training programs for talent at Walt Disney Feature Animation (1994–1997), as well as at VIFX/Blue Sky and Rhythm & Hues Studios (1997–1999).

Question: Jacki, your training and experience with the arts and virtual environments is deep and wide. From the early years of graphic arts and medical illustration leading to an MFA, onto a computer science MS degree in computer graphics, and finally a PhD in computer information sciences and printmaking, your education and career has paralleled the development of virtual reality and the arts. Given that experience, how do you think art will evolve in the next generation of virtual reality environments? Will we see new forms of art, such as virtual sculpture, or virtual environmental experiences being accepted into collections like the Met or MOCA or even the Louvre?

Jacquelyn Ford Morie: I believe they already are, especially if we look at institutions such as Ars Electronica, which has paved the way for such art to be collected and displayed. More traditional art museums have also shown virtual reality artwork. In 2001, the San Francisco Museum of Modern Art presented both of the amazing, fully immersive VR works by Char Davies, *Osmose* and *Ephémère*, as part of their "010101 Art in Technological Times" exhibit. *Beyond Manzanar*, created by Tamiko Thiel in collaboration with the Iranian-American writer Zara Houshmand, is in the permanent collection of the San Jose Museum of Art. Now notably, these museums are in Silicon Valley, but others will undoubtedly follow. The issue for museums is how to keep such complicated pieces running in the face of the rapid change technologies undergo. Many of the best virtual reality artworks no longer run. I have heard that some groups are looking for funding that would bring these works back to life in a museum setting. I hope they are successful.

Question: Jacki, I am thinking of your experience with the educational departments at various places like VIFX/Blue Sky Studios, Rhythm & Hues Studios, and Walt Disney Studios. What kind of fundamental training do you think someone should have before they enter a career in computer graphics and animation? (I used to tell people who wanted to be set designers, that they should learn how to draw and draft with a pencil first, so that they would learn how to understand what they see, and how to explain what they want, but this may not be as valid as it was 5 years ago, especially if they develop smart design programs that you can speak to, think Tony Stark telling his robots what he wants to make in his lab.) What are your thoughts on that?

Morie: While I was at Disney Feature Animation we spent a huge amount of time and energy setting standards for school curricula that would best prepare students to come into the computer animation field. We wanted them to be well-rounded, not just experts in one type of animation software. We looked at their drawing portfolios first and foremost. Drawing, especially life drawing, trains for essential ways in which a person is able to view and interpret the world around them, a key skill for being an artist or animator. If a person was accepted into the animation training program they were given equal amounts of artistic and technical classes and exercises. They not only had to hand-animate a flour sack going through emotional changes, they had to learn to program RenderMan shaders! I still believe, even in this day of smart everything, that there is no substitute for doing it yourself, the traditional ways. That includes drawing, painting, color theory, as well as the particulars of design software. If you let the software make the decisions, then things are going to start looking pretty homogeneous. Creativity requires breaking out of established modes of thinking and most software embeds established modes. When I taught game design at UCLA, the students came in with a huge diversity of majors and talents. I had filmmakers, engineering students, business and even language majors! What was great about that was the teams they put together were eclectic, which led to a broader range of games being made than I believe we would have seen from single-discipline teams. As the academic year was divided into quarters of 10 weeks (too short for developing a game from scratch), we used the virtual world platform Second Life as our "game engine" because it was easy for the students to learn. We ended up with RPGs [role-playing games], games about fine art, puzzle games, and of course, team sporting events (UCLA vs. USC).

[Note: There are some screengrabs from these games in Figure 14.2.]

Question: Given the close connection between our bodies and our minds, and how this can be loosened when we are in a virtual environment, do you see that as an advantage? Do we want to become bodiless in a virtual environment, creatures of pure thought? Or do we need to have a connection to the physical to fully realize the potential of a virtual environment, first-person shooter games aside?

Morie: I think there is room for a continuum of embodiment styles in virtual environments. It depends on the experience being accessed. Often it is enough just to be seeing through disembodied eyes.

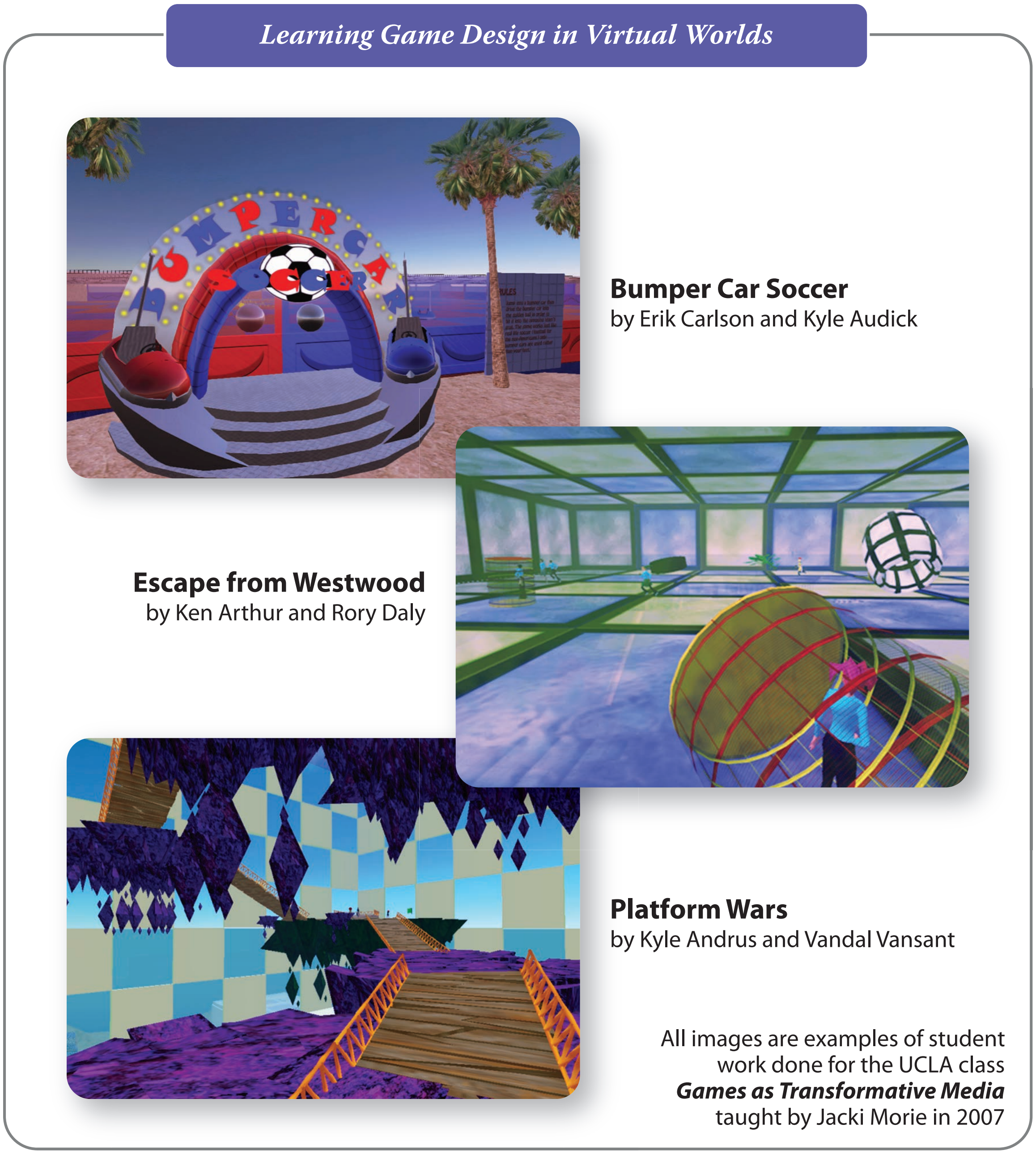

Learning Game Design in Virtual Worlds
BUMPER CAR SOCCER
Bumper Car Soccer
by Erik Carlson and Kyle Audick
Escape from Westwood
by Ken Arthur and Rory Daly
Platform Wars
by Kyle Andrus and Vandal Vansant
All images are examples of student work done for the UCLA class *Games as Transformative Media* taught by Jacki Morie in 2007

FIGURE 14.2 Images of student work from the "Games as Transformative Media" class taught by Jacquelyn Ford Morie at UCLA in 2007.

But if it is a place (say, a virtual world) into which one often goes to spend time and access many experiences, then an avatar is a continuity vehicle for those events. What is wonderful about virtual environments is the freedom that digital representation has for us—we can fly, go deep into the ocean, or become a particle in a particle accelerator! We are not bound by our normal body's physicality.

14.4 CHAPTER SUMMARY: THE MULTIPLE DIMENSIONS OF GAME-BASED VIRTUAL ENVIRONMENTS

In this chapter we looked very briefly at the major aspects of game making and how it relates to virtual reality technology, social media, and human play psychology. The design of virtual reality game-based environments is still nascent, and our technology has just barely started to provide us with the kind of experiences that our brains could infer as a "reality." These current limitations require that you empower your imagination to create game-based environments that engage us and immerse us. Here are some questions to ask as you set about to design a game-based sim:

1. Does this design encourage the visitor to create their own story as well as follow yours?
2. Will your design move the visitor's mindset past the storytelling space and into an exploratory space?
3. What is the metaphysical push–pull of your game-based environment?

The answers to the first 2 questions will lie in your specific design layout and the feedback you get from playtesters who try out your designs. For question 3, let us consider the "reality anchor" concepts of Dr. Michael Heim, in his book *The Metaphysics of Virtual Reality*. He defines the three hooks of our reality anchor as (1) our mortality; (2) our temporality, or the timeline within which we live; and (3) our fragility, the capacity to become damaged in the real world [6]. These three things combine to form our reality anchor, and frame our real-world existence, influencing our cultural and personal behavior. He says, "Should synthetic worlds, then contain no death, no pain, no fretful concerns? To banish finite constraints might disqualify virtuality from having any degree of reality whatsoever. Yet to incorporate constraints fully, as some fiction does, is to produce an empty mirror over and above the real world, a mere reflection of the world in which we are anchored."

Here is the push–pull mentioned in question 3. As a designer, you have the tools to create the mechanics in your game-based sim that will influence the metaphysical push–pull experienced by your visitors. This kind of designing may help people prepare for an incurable illness, as Christian Donlan discovered from playing catastrophe games [7]. Facing virtual death and injury may in fact prepare us for the serious aspects of real life. Christos Reid, a game developer, makes games based on his own human experience. "Game development is therapeutic," he says. "If you suffer from specific anxieties, it's nice to develop a game—it's a mathematical thing. You can put shapes together and everything works. You can construct a world you feel safe in" [8].

As a designer of game-based virtual environments, think about the potential here. We can design places that support play-based behavior to find answers to our real-world problems. We would be using a virtual reality that is anchored in our real-world experience, while also providing a view on solutions that only virtual reality can help us imagine.

REFERENCES

1. Jaron Lanier, “A Vintage Virtual Reality Interview,” accessed March 21, 2015, http://www.jaronlanier.com/vrint.html. (Interview by Adam Heilbrun, originally published in *Whole Earth Review*, 1988)
2. Johan Huizinga, *Homo Ludens: A Study of the Play-Element in Culture* (Routledge & Kegan, 1949), accessed March 24, 2015, http://art.yale.edu/file_columns/0000/1474/homo_ludens_johan_huizinga_routledge_1949_.pdf.
3. Ian Bogost, *How to Do Things with Videogames* (University of Minnesota Press, 2011), Kindle edition, 2403–2404.
4. Edward Castronova, “The Right to Play,” *New York Law School Law Review* 49, no. 1 (2004), 185–210, accessed March 29, 2015, http://ssrn.com/abstract=733486.
5. Stuart Brown and Christopher Vaughan, *Play: How it Shapes the Brain, Opens the Imagination, and Invigorates the Soul* (Penguin Publishing Group, 2009), Kindle edition, 40.
6. Michael Heim, *The Metaphysics of Virtual Reality* (Oxford University Press, 1993), 136.
7. Christian Donlan, “Catastrophe Games: How Spelunky and XCOM Helped Prepare Me for an Incurable Illness,” *Eurogamer.net*, November 26, 2014, accessed March 22, 2015, http://www.eurogamer.net/articles/2014-11-26-catastrophe-games-how-spelunky-and-xcom-prepared-me-for-an-incurable-illness.
8. Keith Stuart, “GameCity Interviews—Christos Reid on Depression and Biography,” *The Guardian.com*, October 31, 2014, accessed March 22, 2015, http://www.theguardian.com/lifeandstyle/2014/oct/31/gamecity-interviews-christos-reid-on-depression-and-biography-video.

Glossary

affordance: or “perceived affordance” is the implied usage of an object through its form and design to the potential user of the object.

agent: a term used by Second Life to indicate the avatar in the scene or the player’s representative.

algorithm: a set of sequential mathematical rules or operations that tells the computer how to do its work.

alpha blending: a progressive rendering process used to transition between two levels of detail by shifting the alpha transparency of the objects, fading one in while the other fades out.

angular deflection: the displaced movement of a virtual physical vehicle along its preferred axis that is a result of forces applied to surfaces, e.g. the wings and rudder of an airplane causing that vehicle to move in head forward direction when the engines are on.

AR: abbreviation of the term “augmented reality,” which refers to the augmentation of the observer’s surroundings to include computer-generated 3D forms or objects, typically using a head-mounted display or computer screen display.

atmospheric perspective (aerial perspective): a sense of distance that is created by the atmosphere in the environment, causing more distant objects to have a lower level of contrast and appear less distinct.

bas relief: a description of sculptural form meaning “low relief” such as you would see on a plaque or coin face.

Boolean: this term in reference to LSL script writing indicates information in a script indicated as TRUE or FALSE.

chim: a Second Life slang term referring to the capacity for a dance heads-up display (HUD) to animate additional avatars alongside the original user.

coalesced object: multiple items of content in a virtual world that are selected, stored into the inventory of the avatar user without linking, and represented in the inventory list with a “pile of blocks” icon.

COLLADA: a file format signified as “content.dae” that enables the upload process for 3D content into a virtual world from numerous modeling programs.

cybersickness: motion sickness experienced from wearing a head-mounted display.

emergent play: the spontaneous occurrence of newly invented games and related play created within a virtual environment by the visitors.

event: a visual or audible experience that is planned for the visitor by the designer of a virtual environment.

fat pack: content in a virtual marketplace that includes one item in many color options.

Fibonacci numbers: a list of numbers that are created from the sum of the previous two numbers; for instance, 2, 1, 3, 3, 6, 9, etc. is a string of Fibonacci numbers.

Foley artist: a sound effects specialist that creates sound for film and television productions.

forced perspective: the creation of an illusory depth and distance effect by shrinking the elements of the scene as they increase in distance from visitor’s viewpoint.

first-person shooter (FPS): one of the original video game forms that displays the scene from the player’s point of view, as they hunt and shoot their opponent.

fractal: a mathematically calculated display of shapes and forms that can be viewed at an infinite number of scales.

game: an activity that includes play behavior guided by a set of rules and a goal, encouraged by a reward, and fosters social interaction.

game mechanics: the mechanisms or rules by which a game functions, such as the taking of turns, the route of travel, and scoring.

game-based environment: a virtual environment that contains elements that encourage and support emergent gameplay.

gameplay: the strategic actions of individuals involved in a game together, often in response to the game mechanics, especially in video games.

genetic algorithm: an algorithm written to function like the natural selection process of the real world and used for such purposes as automated design.

gray goo: malicious content that continuously copies itself and can overload the simulator.

imposter: flat avatar shape substituted in the virtual environment when that avatar is seen at a distance.

Indra: a Linden Lab internal name for the software project concerning the Second Life servers (including the Userserver, Spaceserver, Dataserver, Simulator, and Backbone) and the Viewer (the client).

Lindenmayer systems (L-systems): a mathematical rewriting system originally created by Aristid Lindenmayer to visualize plant growth, and is now used in fractal creating systems to generate random branching and architectural structure.

linear deflection: the movement of a physical vehicle along the line where it is free to roll or slide as it is steered over the terrain.

linkset: a collection of objects or primitives that are joined (linked) thereby creating one “key” or “root” that controls the rest via scripting and its transformational axis.

level of detail (LOD): the amount of detail and graphics/texture resolution that is visible on the client viewer at a given distance and graphics card capacity.

mesh: a 3D object composed of vertices and the planar faces they create.

mind’s eye: the parts of our brain that can provide a perceptual experience without direct visual input.

MMORPG: an acronym that stands for massively multiplayer online role play game.

neoclassical: a modern art or architecture style that refers back to Ancient Greek and Roman periods.

non-player characters (NPC): avatars that are created to “populate” a sim, and are animated by scripts and an artificial intelligence base system like ALICE.

Ogg Vorbis: a free, open source, professional-grade audio encoding and streaming technology that is used in virtual worlds and game environments where sounds are stored as “soundname.ogg” files in the asset system.

organic light-emitting diode (OLED): electroluminescent layers created from electrified organic compounds used to make displays for computer screens, mobile phones, and head-mounted displays.

operator: in the LSL scripting language an operator is a mathematical symbol that indicates the type of math operation that should be performed, such as addition (+) and subtraction (–).

Perlin noise: a procedurally created gradient noise pattern invented by Ken Perlin to enhance the natural look of computer-generated graphics for the movie *Tron* (Disney, 1982).

Phong shading or interpolation: invented by Bui Tuong Phong in 1973, this shading method is used in many rendering applications to create a smooth gradient across the surface of an object.

photogrammetry: the process of making measurements and constructing 3D models from photographs.

postmodern: art and architectural styles started in the late-20th century that deconstructed and destructuralized the forms and ideas of Modern art.

pre-Raphaelite: arts movement in mid-19th century that reintroduced the Renaissance art styles of Raphael (1438–1520) and Michelangelo (1475–1564) back into British painting.

sculpty (or sculpted prim): a primitive object that takes on a special shape by utilizing a sculpt or displacement map that transforms the x,y,z position of its vertices.

six-point perspective: a six-point method for drawing the environment around you, created in two hemispheres such as you would see if you were tracing the world from the inside of a transparent sphere.

string: in LSL scripts a string is a sequence of text data enclosed in quotes such as `"Hello, my name is John"`.

transrealism design: a design that blends aspects of realistic design with fantasy elements in a nonlinear way.

Universal Access (All Access or Design for All): designing and building a virtual-game-based environment that is usable by all types of players, including those who are other-abled.

variable: in LSL scripts a variable is a place to store information; a string is one kind of variable.

visual hierarchy: arrangement of elements in the visual field to clarify and enhance their importance and rank through the use of color, contrast, and scale.

vizome: a portmanteau of "virtual" and "rhizome," as a term to denote the interconnecting networks of social media and the virtual experience.

Voronoi diagram: a visual diagram that shows the closest surrounding areas to each of a given set of random points; a device famously used by Dr. John Snow to show how the Soho cholera epidemic (London 1854) spread from the infected Broad Street pump and not any other water sources in the area.

voxel: a value that exists on a regular grid and can be used to represent volumetric data such as caves and tunnels in a virtual terrain.

Important Links and Resources

CONFERENCES FOR VIRTUAL WORLDS AND VR/AR

OpenSim Community Conference, http://conference.opensimulator.org/
Virtual Worlds Best Practices in Education, http://vwbpe.org/
Silicon Valley Virtual Reality, http://svvr.com/

FREEWARE FOR MODELING AND GRAPHICS/TEXTURE CREATION

The Art of Illusion, http://www.artofillusion.org/index
Blender, http://www.blender.org/download/
GIMP, http://www.gimp.org/
Snappy Tree, http://www.snappytree.com
Can Tree (online tree generator), http://arnaud.ile.nc/cantree/generator.php

Useful Converters to Know about

MeshLab, http://meshlab.sourceforge.net/
MilkShape 3D, http://www.milkshape3d.com/

FREEWARE FOR TERRAIN, FRACTALS, AND PROCEDURAL GENERATION

Mandelbulber, http://mandelbulber.com/
Terragen, http://www.planetside.co.uk/
L3DT, http://www.bundysoft.com/L3DT/
Height Map Editor, http://hme.sourceforge.net/
Nvidia texture tools for Adobe, https://developer.nvidia.com/nvidia-texture-tools-adobe-photoshop
Graphic-generating plugin for GIMP, https://code.google.com/p/gimp-normalmap/

GAME CENTERS/EDUCATIONAL FACILITIES/ONLINE LEARNING

MIT, http://gamelab.mit.edu/
NYU, http://gamecenter.nyu.edu/
Princeton Review, top schools for game design, http://www.princetonreview.com/top-graduate-schools-for-video-game-design.aspx
Game Tutor (online learning), http://www.gametutor.com/live/home-live/

GAME SITES

Gamasutra, http://www.gamasutra.com/
Massively, http://massively.joystiq.com/

INFORMATION ABOUT WORLD BUILDING

Dungeons & Dragons Online, http://www.ddo.com/en

OTHER USEFUL SITES

Make Use Of, http://www.makeuseof.com/
Instructables, http://www.instructables.com/
Don Norman Design, http://jnd.org/dn.mss/affordances_and_design.html

PAYWARE FOR MODELING AND GRAPHICS/TEXTURE CREATION

Adobe Creative Cloud applications, https://www.adobe.com/
AutoDesk subscription applications, http://www.autodesk.com/
Plugins for 3ds Max from AutoDesk, http://www.scriptspot.com

PAYWARE FOR TERRAIN, FRACTALS, AND PROCEDURAL GENERATION

Pro Fantasy, https://secure.profantasy.com/products/ft.asp
Voxel Farm, http://voxelfarm.com/
Filter Forge, https://www.filterforge.com

PHYSICS ENGINE INFORMATION

Bullet Physics Library, Physics Simulation Forum: Physics engine list, books, PhD thesis, online resources, http://www.bulletphysics.org/Bullet/phpBB3/viewtopic.php?p=&f=6&t=63

SCRIPTERS AND SCRIPTING

Script Library Outworldz, http://www.outworldz.com/

RANDOM NAME GENERATORS

RanGen, http://www.rangen.co.uk/
Seventh Sanctum, http://www.seventhsanctum.com/index.php
Chaotic Shiny, http://www.chaoticshiny.com/index.php

VIRTUAL WORLD TYPES/ALIEN WORLDS

Exoplanets, http://exoplanets.org/
NASA Exoplanet Science Institute (NExScI), http://nexsci.caltech.edu/
Stellar Classification, *Wikipedia*, http://en.wikipedia.org/w/index.php?title=Stellar_classification&oldid=644629179

WIKI INFO

Plugin for WordPress, https://premium.wpmudev.org/project/wordpress-wiki/

GOOGLE SITES TEMPLATE

https://sites.google.com/site/projectwikitemplate_en/home

Bibliography

Bartle, Richard A. *Designing Virtual Worlds*. New Riders Publishing, 2004.

Baur, Wolfgang, Jeff Grubb, Michael Stackpole, Chris Pramas, Keith Baker, Steven Winter, and Jonathan Roberts. *Kobold Guide to Worldbuilding* (Kobold Guides to Game Design). Open Design LLC, 2012.

Bogost, Ian. *How to Do Things with Videogames* (Electronic Mediations). University of Minnesota Press, 2011.

Brown, Stuart, and Christopher Vaughan. *Play: How It Shapes the Brain, Opens the Imagination, and Invigorates the Soul*. Penguin Publishing, 2009. Avery reprint, 2010.

Deleuze, Gilles. *The Fold: Leibniz and the Baroque*. Translated by Tom Conley. University of Minnesota Press, 1993.

Deleuze, Gilles, and Félix Guattari. *A Thousand Plateaus: Capitalism and Schizophrenia*. Translated by Brian Massumi. University of Minnesota Press, 1987.

Ebert, David S., F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steve Worley. *Texturing and Modeling: A Procedural Approach*. 3rd ed. Morgan Kaufmann, 2002.

Gygax, Gary, and Dave Arneson. *Dungeon Master's Guide* (D&D Core Rulebook). Wizards of the Coast, 2014.

Heim, Michael. *The Metaphysics of Virtual Reality*. Oxford University Press, 1994.

Huizinga, Johan. *Homo Ludens: A Study of the Play-Element in Culture*. Routledge & Kegan, 1949.

Johnson, Jeff. *Designing with the Mind in Mind: Simple Guide to Understanding User Interface Design Guidelines*. Morgan Kaufman, 2014.

Krueger, Alice, Ann Ludwig, and David Ludwig. "Universal Design: Including Everyone in Virtual World Design." *Journal of Virtual Worlds Research* 2, no. 3 (2009).

Kunstler, James Howard. *Home from Nowhere: Remaking Our Everyday World for the 21st Century*. Simon & Schuster, 1998.

Lévy, Pierre. *Collective Intelligence*. Translated by Robert Bononno. Helix Books/Perseus Books, 1999.

Luebke, David, Martin Reddy, Jonathan D. Cohen, Amitabh Varshne, Benjamin Watson, and Robert Huebner. *Level of Detail for 3D Graphics* (The Morgan Kaufmann Series in Computer Graphics). Morgan Kaufmann, 2002.

Miller, Geoffrey. *Spent: Sex, Evolution, and Consumer Behavior*. Penguin Group, 2009.

Moore, Dana, Michael Thome, and Karen Zita Haigh. *Scripting Your World*. Wiley Books, 2008.

Murray, Janet H. *Hamlet on the Holodeck: The Future of Narrative in Cyberspace*. The MIT Press, 2000.

Pease, Allan, and Barbara Pease. *The Definitive Book of Body Language*. Bantam Books, 2006.

Schell, Jesse. *The Art of Game Design: A Book of Lenses*. Elsevier, 2008.

Sproul, Barbara C. *Primal Myths: Creation Myths around the World.* Harper One, 1979.

Stephenson, Neal. *Snow Crash*. Spectra. Random House, 2000.

Weinschenk, Susan. *100 Things Every Designer Needs to Know about People* (Voices That Matter). New Riders Publishing, 2011.

Zolbrod, Paul G. *Diné Bahane': The Navajo Creation Story*. University of New Mexico Press, 1987.

Appendix A: Game-Based Sim Design Document

Note: This document has been included to nurture creative game-based design thinking. All characters, incidents, and locations you describe should be fictitious, and any resemblance to real-life characters living or dead, events, and location should be entirely coincidental.

A.1 GAME DESIGN DOCUMENT

A.1.1 Summary of Overall Concept and Backstory for Game-Based Sim

Write a 200-word summary of the overall concept for the game-based sim.

A.1.2 Characters

All throughout the game environment, aspects of these characters will be present in clues and other sorts of content scattered about. They can also be played by other people's avatars, in the MMORPG (massively multiplayer online role-playing game) mode, if desired.

- Human Characters: Create a minimum of 3
  - Human Character 1: name, age, economic and social status
  - Human Character 2: name, age, economic and social status
  - Human Character 3: name, age, economic and social status
- Other or Nonhuman Characters: Create a minimum of 3
  - Other Character 1: name, age, special powers, social status
  - Other Character 2: name, age, special powers, social status
  - Other Character 3: name, age, special powers, social status
- Magical Characters: Create a minimum of 1
  - Magical Character 1: name, age, special powers, social status

A.1.3 Places in the Game

In this section describe the terrain, the places, and buildings on the terrain, as well as any archeological and magical significance of areas in the terrain. Also note if any of these areas have a specific bearing or influence on the games to be played on the terrain.

A.1.3.1 Human-Based Places

List of human-based places, for example, the village.

A.1.3.2 Fairy (or Fae)-Based Places

List of fairy-based (or magical places), for example, circle of standing stones.

A.1.3.3 Props Important to the Puzzle Hunt

List the objects used to create interactive elements in the game-based sim.

A.1.3.4 Brief Summary of Backstory

Write a two- to three-page summary of the backstory in the game. Describe the relationships between the characters and define their motivations to interact with each other. Include a method for discovering this backstory for the visitors to the game-based sim, i.e., a diary or old newspaper that describes events that happened.

A.1.4 Ambiance, Attitude, and Mindset of Environment

In this section you should describe the ambiance (visual and audible) of the game-based sim, how that ambiance underscores the attitude of the fictitious characters, and why this creates an emergent play-based mindset for the visitor.

A.1.5 The Goals

In this section you define the goals and scoring system of the game. You may also touch on the mechanics of the game if it helps to define how the goals are achieved.

A.2 TECHNICAL ASPECTS OF GAME-BASED SIM

In this section you define the technical aspects of the game design, the use of screens, controls, and mechanics.

A.2.1 Screens and HUDs to be Used (Inventory Item Prefixes: SCR and HUD)

List and define the screens and heads-up displays (HUDs) to be used in your game-based sim.

A.2.2 Control Devices to be Used (Inventory Item Prefix: CTRL)

List and describe the controls for gameplay that will be provided by the scripted objects in the virtual environment, for example, a sign that gives out a notecard with instructions at the start of the game.

A.2.3 Mechanics to be Used in the Gameplay

Describe the game mechanics for your game; define the mechanisms that will allow for someone to play the game and keep score.

A.3 TERRAIN DESIGN (INVENTORY ITEM PREFIX: TERR)

Define the terrain elements and how these will connect to the gameplay levels, for example, the second level is entered through the stone circle in the forest and the right-hand stone gives clues for getting up to Level 3 if the player says the magic words.

A.4 LEVELS OF GAMEPLAY (INVENTORY ITEM PREFIX: LVL_NUMBER AND NAME)

In the following level (LVL) sections, you will provide the following items:

1. Description of level
2. Location
3. Scripted item used for the game-based play
4. Prize for completion of the level
   - 4.1 Level Entry (LVL_Entry)
   - 4.2 Level 1 (LVL_1)
   - 4.3 Level 2 (LVL_2)
   - 4.4 Level 3 (LVL_3)
   - 4.5 Level 4 (LVL_4)
   - 4.6 Level 5 (LVL_5)
   - 4.7 Level 6 (LVL_6)
   - 4.8 Level 7 (LVL_7)
   - 4.9 Level Meta (LVL_Meta)

A.5 GAME FLOW

In this section, create a step-by-step description of the game flow on the game-based sim, as you see it. For example:

Step 1—Log into the sim and find greeter object (or character; possibly Morwenna in her cat form) that provides information about the nature of the Meta puzzle hunt and some initial clues to Riddle 1.
Step 2—With clues from the initial greeter's message, solve Riddle 1 and get to Level 1.
Step 3—Look for special object in that location (Level 1) to get the next riddle (Riddle 2) and the first part of the Meta puzzle (Meta puzzle 1), which is "worn" as an HUD on the player's screen.
Step 4—With the image clue on the Meta puzzle 1 and the other clues provided in Riddle 2, the player solves the puzzle and finds the location of the Level 2.
And so forth.

A.6 GAME COMPONENTS

In this section you will list all of the game components by category. This will provide you with a build list as well as an organizational structure for your virtual world inventory. Feel free to add or delete categories as you need to.

A.6.1 Interactive Scripted Objects (Inventory Prefix: INT)

List the objects such as doors, windows, props, and so on that interact with the players.

A.6.2 Physics (Collision-Based) Objects (Inventory Prefix: PHY)

List the objects that are physics based, such as vehicles, in this section.

A.6.3 Avatar Components (Inventory Prefix: AVA)

List the avatar-based player characters you will use if you are going to do an MMORPG.

A.6.4 Non-Player Characters (Inventory Prefix: NPC)

List the NPCs you intend to create.

A.6.5 Building and Structural Objects in Game Environment (Inventory Prefix: BLDG)

List the buildings and structural objects you intend to create for the game-based environment.

A.6.6 Attachments or Wearable Objects for Avatars (Inventory Prefix: ATT)

Create a list of all the wearable objects for the avatar players and MMORPG player characters in this section.

A.6.7 Animation (Inventory Prefix: ANIM)

Some of the props and vehicles that you may use will put the avatar into an animated sit or stand position, so create an organized list for these.

A.6.7.1 Animations Needed for Props

A.6.7.2 Animations Needed for Physics-Based Vehicle Objects

A.7 GRAPHICS

In this section you will be defining and organizing the graphics/textures that will be made for the game-based sim.

A.7.1 Style of Graphics

Describe the overall look of the graphics and textures made for the environment. Define the style, for example, photorealistic, and progression of the color palette across the game-based sim.

A.7.2 Graphics Needed for HUD (Inventory Prefix: GFX-HUD)

Describe and list the images needed for the creation of the HUD you are planning to use in the game.

A.7.3 Graphics Needed for Characters (Inventory Prefix: GFX-NPC)

Describe and name the graphics needed for the creation of the NPCs and avatar-based (MMORPG) characters in the game.

A.7.4 Graphics Needed for Buildings (Inventory Prefix: GFX-BLDG)

Define the style and describe the graphics needed for creating the buildings.

A.7.5 Graphics Needed for Landscaping (Inventory Prefix: GFX-LAND)

List and define the graphics needed for the landscaping.

A.7.6 Graphics Needed for Ambiance/Particles (Inventory Prefix: GFX-PART)

List and define the graphics/textures needed for the particle scripts.

A.7.7 Graphics Needed for Weapons and Other Wearable Content (Inventory Prefix: GFX-ATT_type)

Describe and list the graphics/textures needed for weapons and other wearables.

A.7.8 Graphics Needed for Physics-Based Objects (Inventory Prefix: GFX-PHY)

Describe and list the graphics/textures needed for the vehicles and other physics-based objects.

A.7.9 Graphics for Interactive Objects (Inventory Prefix: GFX-INT)

These should be based on the research pictures, with photorealistic images.

A.8 SOUNDS AND MUSIC

In this section you will describe the overall ambiance for your game-based sim, what the "soundscape" will be in your environment.

A.8.1 Sound for Interactive Objects (Inventory Prefix: SND-INT_object name)

List and define the special sounds made by each interactive object in the game-based sim, for example, the diary will make page turning noises when the avatar touches it.

A.8.2 Sound for Physics (Collision-Based) Objects (Inventory Prefix: SND-PHY_object name)

List and describe the sounds made for the physical objects.

A.8.3 Sound for Non-Player Characters (Inventory Prefix: SND-NPC_character name)

List and describe the sounds for the NPCs.

A.8.4 Sound for Buildings (Local Ambiance) (Inventory Prefix: SND-BLDG_location name)

List and describe sounds specific to the buildings, such as interior fireplaces, wind through casements, and water dripping, that will be added to building interiors.

A.8.5 Sounds for Landscape (General Ambiance) (Inventory Prefix: SND-LAND_area name)

List and define sounds specific to the landscape areas around the major level locations on the landscape.

A.8.6 Sounds for Particle Effects (General Ambiance) (Inventory Prefix: SND-PART_particle effect name)

List and define the sound for any particle effects.

A.8.7 Music Needed

Various parts of the game may have associated music. Describe and define what this is, and talk about how it supports the ambiance and emergent gameplay.

A.8.8 In-Game Exploring Music-General Ambient Theme (Inventory Prefix: MUS-AMB_theme name)

List and define the looping music that you will use throughout the game-based sim.

A.8.9 Location Specific Music (Inventory Prefix: MUS-LVL_location name)

List and define the subloops from the general ambient theme, which are location-specific to the game and events that occur in those locations. See Levels section for these locations.

Appendix B: Exhibit C—The Journal of Dr. Aubrey Wynn

(A backstory for "The Search for the Sy," a game-based narrative environment.*)

Every man carries within himself a world made up of all that he has seen and loved; and it is to this world that he returns incessantly, though he may pass through and seem to inhabit a world quite foreign to it.

Chateaubriand

Virtual reality was embedded in our lives by the second half of the twenty-first century. Much of the world's population stayed at home, while their avatars worked and played in asynchronous synthetic worlds. As they lived day to day in virtual reality, people grew accustomed to a virtual skin, grooming their avatars' appearance to define personal identity. Those data, so plastic and malleable, became dominant in their psychological profile, and they became less familiar with their real appearances. Mirror sales dropped. Digital skin textures sold like hotcakes.

Then, one day, something weird happened. The resident population of a virtual environment, known as a "sim," vanished. Their avatars collapsed and flared and disappeared like a piece of flash paper in a magician's hand. They heard *shuuup*, the sound of vacuum sucking up some air, and their avatar was gone, leaving nothing but a particle dust cloud fading away in silence. For the 45 real people who were logged on to the sim called Exhibit C, the psychological result of watching their avatars destroyed caused an intense physical reaction. People vomited all over their keyboards, passed out, and collapsed on their desks. One person even had a heart attack and almost died on the way to hospital. Like victims of the terrorist bombing, these people fell into a mental and physical state of shock. Post-reality stress disorder locked them into a looping cycle of virtual paralysis. Later it was reported on Socialvision, the hybrid media, that these victims saw visions of bright, glowing landscapes. Some of them said that they saw the pits of hell. None of them wanted to log back into a virtual world, not ever again.

1/20/2065—AUBREY WYNN'S RESIDENCE, 2:49 AM EST

As the resident virtuo-anthropologist at Kalen Labs, Dr. Aubrey Wynn didn't hear the phone ring at this hour very often. She dragged her awareness up from the fuzzy darkness of sleep and sat up wondering why she had left the phone turned on last night. The scattered clothing and somnolent form beside her brought the answer to mind. Ah yes, there were less orderly things on her mind when they got home, blissed out from a good dinner and a robust Cabernet. Dr. Kalen snored, rolled over, and continued to sleep. Smiling, Aubrey got up to look for the phone and answer it.

Six hours ago, she was unaware of the events on Exhibit C, and now in the silence of the night, at the behest of Metaversal United, she was going to be updated for assignment.

"Dr. Aubrey Wynn?" inquired the potent voice.

"Yes, speaking," she said, settling onto a kitchen stool.

* This backstory is a work of fiction. The names of characters and places, as well as the incidents described, are fictional or used with fictitious meaning. Any resemblance to persons living or dead, events, or places in the real world is coincidental.

"I am Ang Fulton from Metaversal United. We need help with the problem in one of our simulations." His accent had the crispness of dry crackers, upper middle class, well-educated, and a bit tight in the jaw. Not someone who would enjoy a practical joke.

"Well, Mr. Fulton, I'm an anthropologist," she said. "I don't see how my expertise will help your simulation problem."

Ang cut her off. "Sorry to be abrupt, Dr. Wynn. What we need is someone who can explain an alien culture to us. There is one growing on the sim known as Exhibit C and we think it started when the avatars disappeared 6 hours ago."

"Oh," Dr. Wynn said, in a stunned whisper. "I'll get in touch with Dr. Kalen, and we will set up a team for the field survey as soon as possible."

"Good, I'll organize the upload for you and arrange for the highest performance we can get out of the servers," Ang replied, relief relaxing his voice. "Thanks," he said as the connection ended.

"No problem," she said, to the silent phone line. "Dan Kalen loves getting up in the middle of the night for alien cultural hoedowns."

"Vingh, get Dr. Dana and log in to the Metaversal United staging area," Dan Kalen rasped 20 minutes later. His jagged voice was just a bit louder than the mechanical rattle of his vapor cig. "Dr. Aubrey Wynn will meet with you both soon." Dan looked across at Aubrey, ogling her as she suited up for the V-rig. She rolled her eyes, and kept dressing.

Seated in his office overlooking the harbor of Mumbai, Vingh sighed. Kalen didn't call him when everything was fine; Kalen called when it all went "tits up" and needed to know why.

"Yeah, good morning to you too, Dan," Vingh replied, pushing back a mass of silver hair to look at the clock, "I'll be in contact as we set up for teleport to the incident site."

The trip to Exhibit C was the digital equivalent to climbing Mt. Denali. An overworked server, staggering like a climber succumbing to hypoxia, held them motionless for long moments as their avatar data trickled into its cache.

While Aubrey stood there, waiting and resisting the impulse to hit more controls, she could not decide what caused her more irritation, the frozen molasses downloads or the neon pink onesie Dr. Jessellyn Dana chose to wear. Squinting, Aubrey thought, God's teeth, was there a name for that color, or was it a primal sound?

"Be careful, Aubrey," said Dan, his voice a soft growl in her ear. "We don't know what this is yet." Aubrey nodded and yawned, trying to shake off the rags of sleep as she booted up her V-rig.

Dr. Nao Vingh, their ancient and cranky expert on Geo-meshology and Astro-electrical fields, was not a believer in fashion forward expedition wear. Vingh snorted rudely over the com when Jessellyn Dana's avatar rezzed. Aubrey's avatar smirked at Vingh's materializing form, and Jessellyn ignored them both.

"Jumpin June bugs!" Jessellyn twanged with her Tennessee mountain accent. "What's happening here? Look at this place, it's sure one hot mess!"

They stood on a temporary teleport pad, overlooking the region from low cliffs on the western edge. Near the center of the sim, rising above their eye level, was a massive building, a huge pile of carved stone and marble. A virtual Grand Central Station, any old city would landmark and give tours in. Flooded, with several inches of water covering the main concourse floor. Surrounding the exterior and covering the rest of the sim was a mess of overgrowth, fractal vegetation resembling digital glitches of real plants. These plants had strange primitive shapes. Spindly stocks with enormous leaves and mangrove knee-like roots. There were copses of oddly tall palms, spiked like Joshua trees. Along the shore, 10 meter tubers rooted in the muddy banks grew scaly green bulbs. The air stank of rot and wet stone. Small ripples on the water surface hinted at the surreptitious movement of submerged creatures. As the scientists climbed down from the cliff, Aubrey looked at her on-screen stats. From the readings, the server was on the verge of vomiting the whole thing back into reset. Their feet splashed through the shallow water and weeds covering the terminal floor.

"Duck!" Shouted Vingh, as a dragonfly, the size of a military drone, buzzed over their heads.

"Jesse," Aubrey whispered, "are those things carnivorous?"

"Not likely," Jesse exhaled, "but keep your hands down. No way of knowin' how those critters got programmed."

1/27/2065

The group of scientists established a makeshift camp on the hills above the terminal building overlooking the sea, and set their avatars on permanent rezz. The whole team covered 20 real-life time zones, so at least one of them was awake and logged in at any given time. Dr. Jessellyn Dana, a talented virtuo-biologist despite her theoretical fashion sense, formulated a rough hypothesis to explain the odd flora and fauna.

"Aubrey, this sim is de-evolving," she said, "like the Earth would if human beings became extinct. Look at the primitive structure of these plants, and these dragonflies are like something from the fossils of Earth's Carboniferous age."

Aubrey nodded as Jesse talked, remembering the deadly sound of those insect wings. Those insects could be the virtual equivalent of a prehistoric Meganeura dragonfly. Modern dragonflies were stone-cold killers who could drop with deadly accuracy onto their prey.

In the distance, as Vingh wandered around, intrigued by the strange electrical readings emitted by the landscape on Exhibit C, he noticed that the land textures etiolated and the terrain stripped down to its wire frame. Odd phenomenon, he thought, sputtering into his recorder about mesh fractures and seismic anomalies. The evidence of cultural remainders puzzled Aubrey. As a virtuo-anthropologist, Aubrey studied the cultural life cycles of native avatar populations. Aside from flooding on the terminal floor, a result more likely to occur from a data shift in the heightmaps than tragedy, there was no evidence of a virus epidemic or war. The improbable concept of avatars vanishing through a split in the face of their world seemed possible here. What remained look like daily life, abandoned in midsentence and covered with ash. This was a place held in a frozen moment. A virtual Pompeii with no bodies.

2/2/2065 AUBREY WYNN'S JOURNAL

I know about creation myths, of course. The Native Americans have one about the Coyote, a trickster spirit who spilled a bucket of stars and made the Milky Way appear. Dan Kalen told me that there were odd, fantastical reports from earlier visitors. There are stories that they tell of bright stars walking among the trees. There are a couple of blurry photos, showing flashes of light in the darkness, no proof of anything anomalous. I cannot shake the feeling that we are watched. There is another presence here, a much older nonhuman, machinelike presence.

2/3/2065

This morning, more strange readings appeared on Vingh's instruments. Assuming a malfunction, Vingh recalibrated the system and set it to record again. Later, when his analytical mind connected the strange readings to memories of his early graduate studies of space anomalies, he got up from his dinner and went back to review the data. "I still don't believe this kind of thing can happen in a virtual world," Vingh commented when he returned to the camp. "The sim seems to function with abnormalities and screenshots indicate weak spots or transparencies in the meshes, located below the ruins of the station and its support piers. I have never seen anything like this before."

"And what about the unusual underwater plant life here?" Jessellyn said. "The silicon-based diatom life in the water is much more robust, their numbers much higher than usual. Vingh, do you think that phenomenon correlates with your findings?"

"Keep looking," Aubrey said, poking the fire. "Something is happening. We need the right lens to see it."

2/23 2065 AUBREY WYNN'S JOURNAL

We cannibalized my equipment to help Vingh construct a new sensor, something he calls a "crystal oscillator." Vingh's theory about creating a resonant wave with the crystal oscillator could provide us with more information about these mysterious anomalies. I'm monitoring the radio frequency spectrum for messages from the previous inhabitants. Last night, Vingh's said that he keeps thinking of an old Zulu myth, about how the stars are the eyes of our ancestors looking down on us. After he said that, we all looked up.

3/5/2065

The 23rd attempt to make an oscillator that resonated with the correct frequency was successful. Vingh found red crystal shards on the lake bed under the station and incorporated them into the device. At once a low subsonic resonance filled the space, and the mesh terrain surrounding them seemed to sparkle.

3/8/2065 AUBREY WYNN'S JOURNAL

I have fallen under the spell of this place. Somehow it seems to heal me. Even my traumatic memories and darkest fears have become like old photographs bleaching and curling up and the sun.

3/9/2065

"I've got something!" Vingh shouted. "It's on the highest frequency. Not words, but numbers in groups of three, almost like the coordinates of the sim. The numbers start, and play for 10 minutes, then pause and repeat almost like a beacon."

"Interesting," said Aubrey. "I'll try to connect these numbers to some sort of language; but the progress may be slow." I truly have no idea what I'm looking at, she thought.

3/12/2065

"OK, I think I have a translation," Aubrey said. "The message seems to be saying this: 'We are Sy, the walking stars. Carbons, the Sy adapt. Gather with us.'"

Jessellyn shivered with excitement at the possibility of discovering intelligent life. Aubrey thought of walking stars, the words reminded her of the old images she had, the testimonies of the surviving victims.

"Where are they?" Vingh said, on the hunt, with his oscillator.

Where indeed, thought Aubrey.

3/20/2065 AUBREY WYNN'S JOURNAL

Vingh has disappeared. His avatar missing when I logged in for the day. Jesse found the crystal oscillator lying on the ground, its red crystal melted and shattered. All Jesse and I have left from him is this chat log that says, "Aubrey, Jesse, I know what's happening. Our virtual presence is taking the Sy world apart. Our two species must find a way to live together or be torn into atoms. Help them! Help me, Aubrey!"

GAMES

Written as the successor to *Virtual World Design: Creating Immersive Virtual Environments*, this book carries the ideas brought forward in its predecessor to new levels of virtual world design exploration and experimentation. Written by an Emmy award–winning designer with 22 years of experience creating virtual environments for television and online communities, **Extending Virtual Worlds: Advanced Design for Virtual Environments** explores advanced topics such as multiregional design, game-based sims, and narrative structure for environments.

The book provides bedrock knowledge and practical examples of how to leverage design concepts within the intertwined structures of physics engines, level of detail (LOD) systems, and advanced material editors. It also shows designers new ways to influence the experience of virtual world visitors through immersive narrative and storytelling. With over 150 illustrations and 10 step-by-step projects that include the necessary 3D models and modular components, it delivers hours of stimulating creative challenges for people working in public virtual worlds or on private grids.

By using this book, novices and advanced users will deepen their understanding of game design and how it can be applied to creating game-based virtual environments. It also serves as a foundational text for class work in distance learning, simulation, and other learning technologies that use virtual environments.

K24293

CRC Press
Taylor & Francis Group
an **informa** business
www.crcpress.com

6000 Broken Sound Parkway, NW
Suite 300, Boca Raton, FL 33487
711 Third Avenue
New York, NY 10017
2 Park Square, Milton Park
Abingdon, Oxon OX14 4RN, UK

ISBN-13: 978-1-4822-6116-5
90000
9 781482 261165