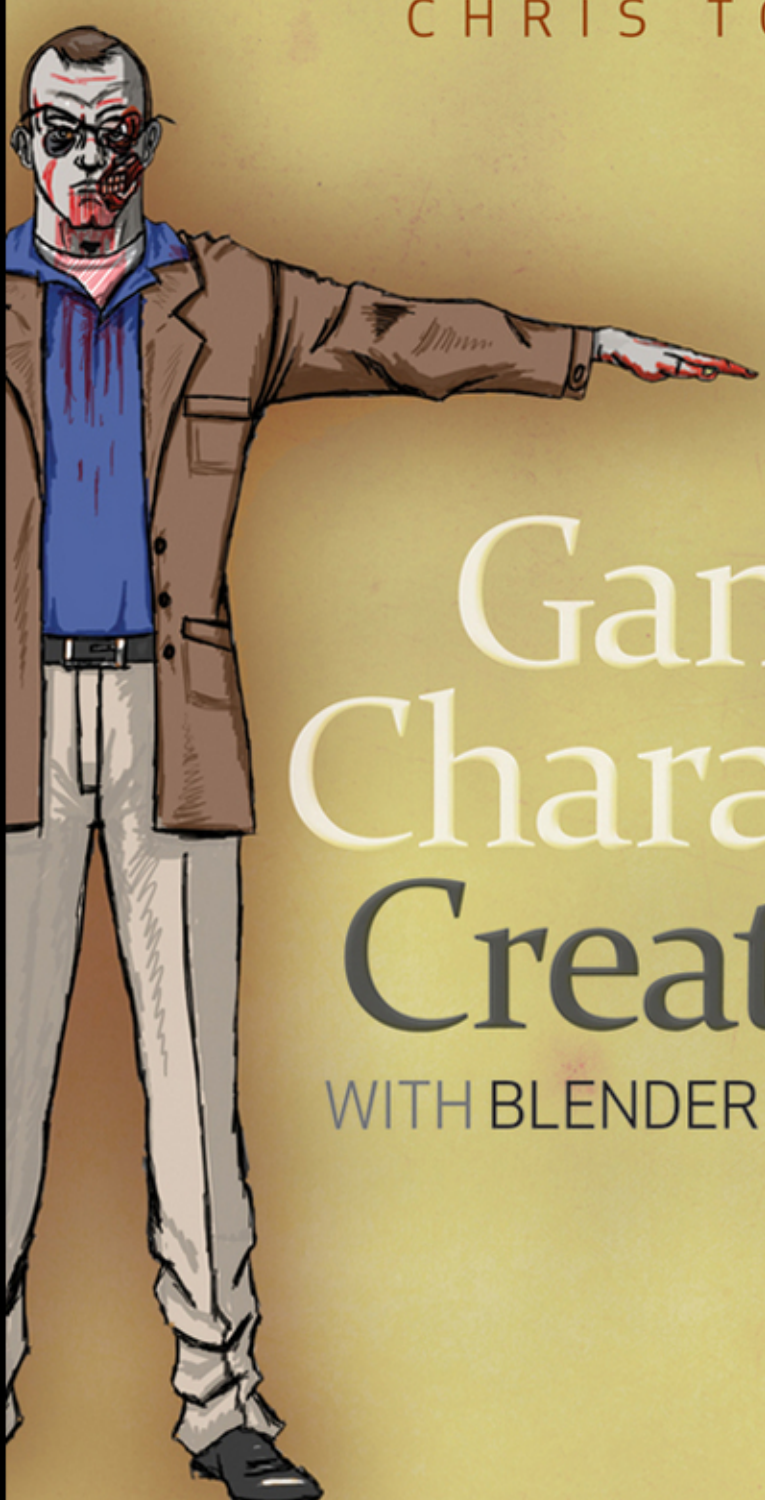
CHRIS TOTTEN



Game Character Creation

WITH BLENDER AND UNITY

SYBEX | SERIOUS SKILLS.

Game Character Creation with Blender and Unity

Game Character Creation with Blender and Unity

CHRIS TOTTEN

John Wiley & Sons, Inc.

Acquisitions Editor: Mariann Barsolo
Development Editor: Laurene Sorensen
Technical Editor: Terry Wallwork
Production Editor: Dassi Zeidel
Copy Editor: Liz Welch
Editorial Manager: Pete Gaughan
Production Manager: Tim Tate
Vice President and Executive Group Publisher: Richard Swadley
Vice President and Publisher: Neil Edde
Book Designer: Caryl Gorska
Compositor: Chris Gillespie and Kate Kaminski, Happenstance Type-O-Rama
Proofreader: Amy Schneider
Indexer: Jack Lewis
Project Coordinator, Cover: Katherine Crocker
Cover Designer: Ryan Sneed
Cover Image: Chris Totten

Published simultaneously in Canada

ISBN: 978-1-118-17272-8
ISBN: 978-1-118-22690-2 (ebk.)
ISBN: 978-1-118-23771-7 (ebk.)
ISBN: 978-1-118-26457-7 (ebk.)

Library of Congress Control Number: 2012937911

10 9 8 7 6 5 4 3 2 1

Dear Reader,

Thank you for choosing *Game Character Creation with Blender and Unity.* This book is part of a family of premium-quality Sybex books, all of which are written by outstanding authors who combine practical experience with a gift for teaching.

Sybex was founded in 1976. More than 30 years later, we're still committed to producing consistently exceptional books. With each of our titles, we're working hard to set a new standard for the industry. From the paper we print on, to the authors we work with, our goal is to bring you the best books available.

I hope you see all that reflected in these pages. I'd be very interested to hear your comments and get your feedback on how we're doing. Feel free to let me know what you think about this or any other Sybex book by sending me an email at `nedde@wiley.com`. If you think you've found a technical error in this book, please visit `http://sybex.custhelp.com`. Customer feedback is critical to our efforts at Sybex.

Best regards,

Neil Edde
Vice President and Publisher
Sybex, an Imprint of Wiley

To Clara, without whom none of this would have been possible

Acknowledgments

Books like this are the Olympian effort of not just the author but a great many people, who all deserve recognition for their efforts. I would of course like to thank the fine folks who make Blender and Unity and distribute them for the masses to use. You are doing a great thing for independent artists and game developers. A special thanks goes out to Ton Roosendaal for getting me in touch with Wiley in the first place to make this book happen. ■ I'd like to thank the teachers who introduced me to 3D art and animation back in college and pushed me to pursue a career making and studying games. I'd also like to thank the D.C. chapter of the International Game Developers' Association for being so encouraging and including me in many of your great game projects. ■ I'd also like to thank the people who helped me put this book together, including Mariann Barsolo, Laurene Sorensen, Pete Gaughan, Connor O'Brien, Jenni Housh, Liz Welch, and Dassi Zeidel. You guys not only fixed many of my grammatical, technical, and formatting errors but also taught me a great deal about the publishing process. I also know more about Blender than I did going into this process because of you. ■ I would also like to thank the administrators of `www.lovetextures.com` for their generous permission to use some of their texture files in the downloadable materials. They do a great job and everyone reading this should check them out. ■ Lastly, I'd like to thank my family for their love and support as I worked on this project, especially my mom and dad. Without you guys I wouldn't have been capable of writing this. Thanks also to my fiancée, Clara, for pushing me to pursue this project and putting up with my long workdays creating art for the book.

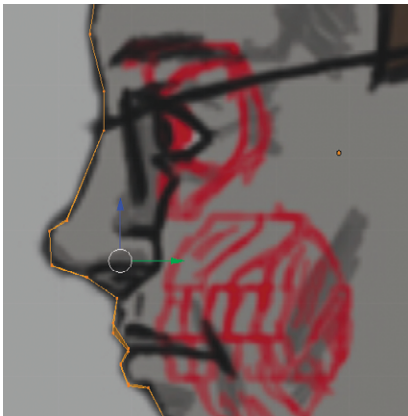
About the Author

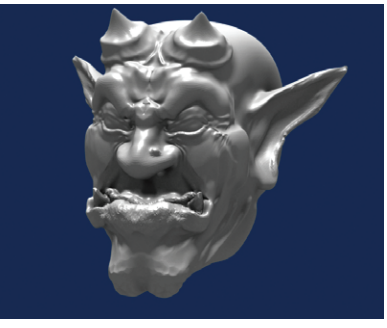
Christopher Totten is a Washington, D.C.–based professor of game design and 3D animation. He has participated in several independent game projects as an artist, animator, and project manager. Chris has written articles featured on both Gamasutra and Video Game Writers. He has been a guest speaker at Dakota State University's Workshop on Integrated Design and at GDC China.

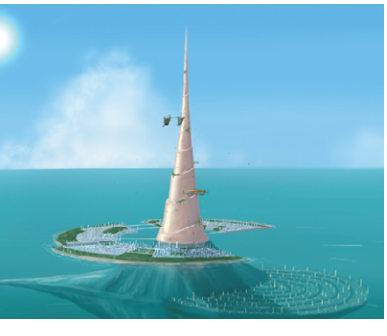
He has a master's degree in architecture with a concentration in digital media from the Catholic University of America in Washington, D.C. Chris wants to help shape a new generation of game designers who look deeper into their designs. He works with students and other designers to challenge gaming conventions through cross-disciplinary research.

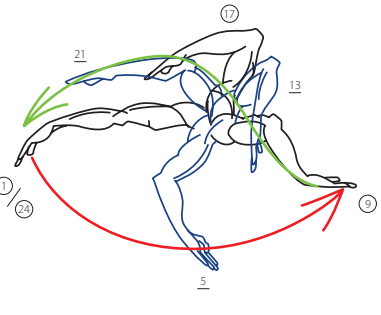
CONTENTS AT A GLANCE

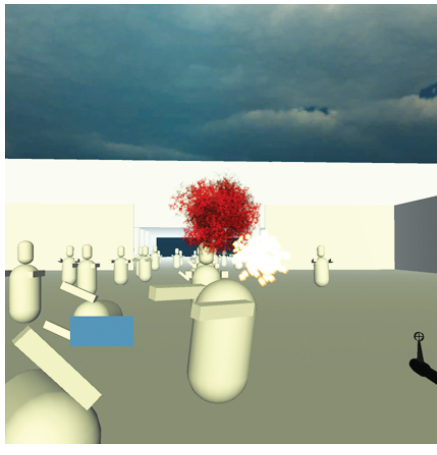
Contents











Introduction

When video games first entered the consumer market scene in the 1970s, they were incredibly simple, abstract affairs when compared to today's multi-million-dollar projects. An Internet search for "Pong source code" shows that the game can be made with about 200 lines of C++ code, whereas a game like Crysis runs on over one million. Back in the days of Pong and other early games, it was within the reach of enthusiasts, albeit those with programming skills, to make their own games on home computers. The founders of CodeMasters, Philip and Andrew Oliver, used to write games on their home PC and distribute them by having the source code published in magazines for other children to type.

Years later gaming computers and consoles became more powerful, and new worlds of artistic and narrative possibilities opened up. Commercial game production was no longer accomplished by a single talented programmer but rather by an entire team of programmers, writers, sound technicians, artists, and other creative professionals. As games became more complex, however, the vision of making a "do it yourself" game that was as good as, if not better than, what was in stores grew increasingly further away from gamers.

That is…until now…

You hold in your hand a guidebook that will put you on your way to making your own commercial-quality games. You may be holding it because you are interested in making your own games or because you want to learn about how 3D art works together with game engines. Whatever your reasons, the important part is that with this book the power is back in the hands of you, the independent game maker…part of it anyway.

Game production in this new do-it-yourself world is still largely the domain of several talented individuals working on one project. Even Jonathan Blow, the rockstar indie developer behind *Braid,* had help from artists and musicians.

Game design theorist Jesse Schell, in his book *The Art of Game Design* (Morgan Kaufmann, 2008), said that a well-rounded game designer needs to have working knowledge of animation, anthropology, architecture, brainstorming, business, cinematography, communication, creative writing, economics, engineering, history, management, mathematics, music, psychology, public speaking, sound design, technical writing, and visual arts—a body of knowledge that no one book can impart. However, this book will show you how to complete one of the more complex parts of game content creation: character creation.

One of the most rewarding and challenging portions of game development is creating good characters. It is rewarding because you get to make enemies or even the character on the box. It is challenging because 3D content creation programs like 3ds Max, Maya, ZBrush, and commercial game engines are both expensive and incredibly difficult to learn. However, by mastering the concepts in this book, you can apply your newfound uber-talents in game art to much simpler (yet still incredibly important) visual elements of games, like environmental modeling and props.

As previously stated, the tools used by the "big guys" to accomplish the tasks that lie before you are expensive, and each tool is specialized in only one or two parts of the entire process, requiring lots of time spent importing and exporting. You, as part of the new DIY movement, are more resourceful and economical than that, however, and will streamline your process with two powerful and financially accessible tools: Blender and the Unity game engine.

What Is Blender?

Many Blender books begin with a sort of sales pitch for the program. This comes from the era and mind-set where Blender was an underdog in the animation and games industry. Although it's not at the point of being a so-called industry standard yet, it is far from the up-and-comer that it was even a few years ago. As it stands, Blender is an incredibly full-featured and powerful program that has found its way into the productions of *Spider-Man 2*, Oscar nominee *The Secret of the Kells*, Sega's *Virtua Tennis 4*, and scads of other movie and mobile game projects. Blender is demonstrating the decline of reliance on one or two "industry standard" programs and instead a workflow of experimentation that results in studios having their own "office standards." So what is Blender?

The short answer to this question is that Blender is the world's foremost free and open-source 3D graphics application. That's right, free, as in free software: required balance = $0.00. The long answer takes us back to the 1990s when Blender was developed as an in-house application for the Dutch animation studio NeoGeo. The program was authored primarily by Ton Roosendaal, who eventually started Not a Number Technologies (NaN) to distribute the program. It was released as shareware until 2002 when NaN went bankrupt and Blender was released under the GNU General Public License (GPL). Blender's licensing is important to note, as it is the crux of how flexible the program is in terms of fitting into the game designer's workflow.

The GPL is a license that stipulates that the source code for programs under the GPL is freely available for anyone to download, use, copy, change, and distribute as they see fit.

Blender itself, and the source code that runs it, is available to anyone who wants it. Those looking to modify the code also have the freedom to do so as they wish, as long as they explicitly identify the changes they have made and release the code as they got it: open source and free of charge. What this means is that Blender is absolutely free, "as in free speech" as GPL author Richard Stallman would say. For artists this means that 3D modeling and animation is no longer the domain of corporations or the very rich but whoever believes that 3D is the proper medium for their next design production or work of art.

For many studios, however, the low price tag is a moot point. Many studios use Blender because they can freely modify the program and develop their own in-house versions of the software. This serves as a huge boon to Blender itself, as many of the features created by these production houses find their way back into the versions released by the Blender Foundation. These changes also occur incredibly fast: where other programs release a new version once a year, Blender is constantly in development both by the Blender Foundation and independent users. This collaborative development model means that the Blender Foundation runs with only four employees, one of whom is Roosendaal himself.

Why Make Game Assets in Blender?

What Blender's open-source outlook and community focus means to you, our DIY game auteur/intrepid hero, is that Blender is an ideal tool not only for its price tag but also for its flexibility. Even if you are not programming literate, you can access the massive Blender community for help with using the program or directing you to a build of Blender that will best serve your needs.

Additionally, Blender features a variety of modes that are analogous to other industry-level packages like ZBrush and Maya. While the workflow of someone making characters in other programs may have them exporting, importing, and re-exporting their models, all of the 3D content creation can be accomplished within Blender.

Beyond cost and time concerns with using proprietary software packages, Blender also excels in compatibility. The program is cross-platform, meaning that it works not only in Windows but also in Mac OS (both Power PC and Intel models) and in Linux, the operating system on which Blender originated. Best of all, files from one version of Blender are compatible with others, even older versions (minus some of the animations created in post-2.5 versions).

When I go off into one of my unintentional sales pitches for Blender, people often ask me, "If it's so great, then why doesn't every studio just use it?" To be honest, programs like Maya and 3ds Max are still the so-called industry standards and due to the mighty

funding power of Autodesk will likely remain that way for a long time. The other answer is user preferences. A lot of current game studio employees began their 3D careers in proprietary programs, and jumping ship for a new content creation environment in the middle of development is a nightmare, no matter how amazing that new environment is. That being said, Blender is still an amazing tool to have at your disposal, whether you are a professional looking into open-source software options or a student looking to beef up your portfolio.

What Is the Unity Game Engine?

Unity is a game engine created by Unity Technologies for Windows and Mac OS. For those who are unfamiliar with game engines, a *game engine* is a program within which someone can create a video game. These programs include a rendering engine for displaying game art assets, physics engines for controlling object interaction, sounds, scripting, animations, and many other functions we associate with the modern video game. Unity itself can produce 3D games for Windows and Mac OS. With special licenses it can also make games for the Nintendo Wii, Xbox 360, PlayStation 3, iOS platforms such as the iPhone and iPad, and Android devices. One of Unity's unique features is that it can also deliver 3D game content through web browsers by way of the Unity Player plug-in, which works with Internet Explorer, Firefox, Safari, Chrome, Opera, Mozilla, Netscape, and Camino. It can also deliver content to Adobe's Flash Web Player. Unity is notable for receiving the *Wall Street Journal*'s 2010 Technology Innovation Award in the software category. Unity Technologies was also named one of Gamasutra's "top 5 game companies of 2009."

Awards and features aside, Unity is one of the up-and-comers in the game industry both for its ability to reach so many users on multiple platforms and for its ease of development. Unlike other game engines that are built as esoteric, expensive, studio-specific tools that are later released to the public, Unity has developed more as a consumer product. Where this makes a difference is in the user interface: compared to the other game engines out there, Unity is user friendly and financially accessible. This combination of flexibility and economy has allowed it to be a favorite of both independent studios and larger companies for games such as EA's *Tiger Woods PGA Tour Online*, *Marvel Super Hero Squad Online*, *Cartoon Network Fusion Fall*, and the *Battlestar Galactica* MMO.

Unity comes in two basic forms: basic (free) Unity and Unity Pro. Unity gets much of its user base with the basic version, which is free of charge. This free version gives anyone making games access to a fully featured game development environment. It lacks some of the graphic and deployment features that are integrated into the Pro version, and users

of the free version must publish their games with a preinstalled Unity splash screen at the beginning of their games. These issues aside, the free version of the Unity engine is excellent for tackling game projects of all scales.

The Pro version of Unity is priced at $1,500 for one *seat* (or copy of the program). If you are part of a studio that makes over $100,000 a year, this is also the version you must use if you are going to sell a game made in Unity. Even if you are one of these big guys, however, Unity Pro comes with no yearly subscription fee or royalty fee for delivering game content, unlike other so-called freely available engines like the Unreal Development Kit. This book will use (and was created with) the free version, as it is assumed that most of you will be using it.

The engine makes itself more user-friendly with several features. One of them is Unity's WYSIWYG environment. WYSIWYG stands for "what you see is what you get," as in the image on the screen is what will appear in your game. This makes it different from other game engines that rely mainly on coding for game development. The most important features, however, are its drag-and-drop interface and its asset manager. Unity manages assets—3D models, textures, sound files, scripting files, and so forth—by creating a project folder on a user-specified location on the hard drive. This project folder is viewable directly in the Unity interface, and any assets saved into it are automatically added to the project, eliminating the need for troublesome importing of potentially incompatible files. These assets can then be dragged and dropped into place in Unity itself, whether they are a 3D model that needs to be somewhere specific in a level or a behavior script that is added to that model as a "component" of it. Once these relationships are established, the resulting objects can even be saved as *prefabs*, prefabricated collections of objects that can be dragged and dropped again and again into the game scene.

Where Do I Get Blender and Unity?

Blender and Unity are available for download directly from their developers' websites as shockingly small download files. For example, 3ds Max and Maya can take up several gigabytes of hard drive space, whereas Unity's install file is under 700 MB. Blender takes the prize for smallest install file at 21 MB. The websites are:

- Blender: `www.blender.org`
- Unity: `www.unity3d.com`

These sites also update their versions regularly, so you are always sure to have the latest version of the software if you check often.

For Blender, there is also a separate website called GraphicAll where you can download the latest user-generated or development versions of the software. While these are not the Foundation's stable releases, they feature a ton of useful plug-ins that could potentially benefit your workflow. These can be found at `www.graphicall.org`.

Who Should Buy This Book

This book will teach you how to use Blender and Unity to create characters in a game development workflow. If you have not yet bought this book and are perhaps reading it in your bookstore's computer section, then perhaps you may want to see if you fit into one of these groups:

You're a motivated person who is new to 3D modeling and animation and wants to make games. Are you a general artist who says to yourself, "Gee, I'd really like to make my own video game," but you don't know where to start? You may be afraid to step into the potentially overwhelming world of other 3D modeling packages and the variety of software needed to make a game due to expense or perceived level of difficulty. This book will not only give you an overview of Blender and Unity but will also help you on your way to making the next big indie hit.

You know how to use Blender's modeling and animation tools, but you're not sure how to get them to work for a game workflow. Do you already have all the basic Blender skills you need but feel unsure about how to use them to create game characters or import them into a game engine? Maybe you are an animator who is used to working with things like the subsurface modifier to create smooth Pixar-like characters but can never quite get a handle on how to make a low-poly character look good. If so, you may want to skip the basic stuff in the book and move ahead to the chapters on modeling and other techniques.

You are familiar with game art creation in other software packages. Do you already know how to model low-poly characters to create game assets but want to know how to do the same in Blender? There are many reasons to know multiple software packages: to look good in job interviews, to get work done when you're not at your office computer, or even to make Blender part of the workflow of your game studio. You may skip around this book, because you already know the workflow required for low-poly game assets and instead can use the book to get an idea of how Blender can streamline your workflow. You'll see how Blender is a useful all-in-one environment and eliminates the need to export models to and from other programs.

You know all that 3D stuff and want to get your hands dirty in an engine. While this book will mostly focus on Blender game character creation, there will be a fair amount of information on how to get the model up and running in Unity. Chapter 9, "Unity Engine Basics," and Chapter 10, "Implementing Your Zombie in a Unity Game," explore Unity's interface and importing workflow. These chapters also show you how to implement the character into an arcade-style shooter.

Overall, if you have an interest in games and 3D modeling or animation, this book is probably for you.

Whether you are a hobbyist game developer or an industry insider looking for new tools, this book will help you diversify your skill set and potentially reach new audiences with Unity's wide-ranging delivery options. With the rise of the casual gaming market, platforms where games are played are beginning to be as diverse as the people who play them. Blender and Unity can help you create your projects with efficiency of time and money and give you the ability to reach a broad user base.

What You Will Learn

So let's get into it, O intrepid DIY gaming warrior, and talk about what you'll be doing with this book. As stated earlier, through the exercises in this book you will be creating a game character and adding it to a game program in the Unity engine. This is a much more difficult task than many people realize, so we will attack it step by step. The book gives you a basic lesson in game art, and then introduces you to the Blender interface. From there you will model, texture, sculpt, rig, and animate your character before moving into the Unity engine, where you will import the character and use scripting to give it behaviors. At the end of our quest, you will have your very own game enemy to fight.

For these exercises, you will create a common enemy found in many games. Though it is not the greatest warrior around, this character can be particularly terrifying as it tries to eat your brains with hordes of its friends.

While many people think that zombies are a bit cliché in today's gaming landscape, they provide a good basis for learning how to create an animated character model and give them simple behaviors. They are also a lot of fun to fight (Figure I.1).

Figure I.1

This book will lead you through the process of creating and implementing a zombie character.

How to Use This Book

The chapters in this book go step by step and you are free to take them on either in a linear fashion or take your pick and skip around. Each chapter demonstrates part of the 3D character creation process and brings you closer to the final showdown at the end of the book with your horde of zombies.

Hardware and Software Requirements

To do the exercises in this book, you will need a copy of both Blender and Unity. Specifically, make sure you have Unity version 3.5 or above. They are both available for download from their respective product websites:

- Blender: `www.blender.org`
- Unity: `www.unity3d.com`

To run these programs, you should check your hardware against the specifications listed on the Blender and Unity websites at these sites:

```
www.blender.org/features-gallery/requirements/
http://unity3d.com/unity/system-requirements.html
```

Generally, you will want a computer with at least a 2 GHZ dual-core processor, 2 GB of RAM, and an open GL graphics card with a minimum of 64 MB of RAM. As the Unity website so wonderfully puts it, "any card made in this millennium will work." For Blender, you will want a three-button mouse with a scroll wheel, because many navigation functions are mapped to the middle mouse button (MMB).

Additionally, you may want a 2D art program with which to create textures for the character. Adobe Photoshop is the program of choice, but ArtRage Studio Pro and GIMP are both viable substitutions for artists on a budget. They can be found here:

- ArtRage: `www.artrage.com/artrage-studiopro.html`
- GIMP: `www.gimp.org`

Online Resources

Online resources for this book can be found at:

```
www.sybex.com/go/gameblenderunity
```

These project files will help guide you through the exercises if you wish to skip around or even if you would like to compare your own work to the example files.

It's important to understand that both game design and game art are not easy. They can be a hectic business of long hours and hard work. However, they can also be incredibly rewarding when you see something you created bring a smile to another person's face. Game art and especially 3D modeling can be frustrating at times, but press on, if not only to improve your own skills but also for the chance to unleash your frustration on a video game version of the author (the zombie is me, after all). What other software books offer that?

With that, let's dive headfirst into your adventures in Blender and Unity. It's going to be fun.

What's Inside

Your missions, should you choose to accept them, are thus:

Chapter 1, "Basic Game Art Concepts" This chapter describes the process of preparing art for video games and important things that you, the 3D artist, should know. Game artists

don't have the technical leeway that animators and filmmakers have, so this chapter will make sure you are aware of concepts such as polygon count, normals, game engines, and yes, even scripting.

Chapter 2, "Blender Basics for Game Characters" Here you will get your feet wet in the open-source modeling program Blender. If you are a beginner with the program, this is the chapter for you—it will introduce you to the interface and to some of the tools you will be using during your quest as a 3D game artist. If you already have experience with Blender, feel free to skip this chapter and move ahead. Or not—despite my experience I often discover something new from reading these types of chapters, so you may too.

Chapter 3, "Modeling the Character" This chapter introduces you to the character you will be modeling and walks you through the step-by-step process of modeling it in Blender. You will learn two different modeling methods—box modeling and poly-by-poly modeling—that you will use to create different parts of the character.

Chapter 4, "Prepping for Zombie Details with UV Unwrapping" This chapter begins the process of bringing your model to life with Blender's sophisticated UV unwrapping tools. You will discover how to draw seams on your model that will become the basis for UV layouts. You'll then take these layouts into the UV editor interface for some fine-tuning and export them for painting in other programs.

Chapter 5, "Sculpting for Normal Maps" In this action-packed chapter, you will make the zombie appear to have high-definition detailing with Blender's Multires modifier and sculpting mode. Intuitive for artists, the sculpting features in Blender will be useful for creating gory details on the zombie model, such as facial damage and rotting flesh. Additionally, you will learn how to add detail and wrinkles to clothing. Finally, you will learn how these high-resolution details can be brought back to the low-polygon model that will be imported into the game through the magic of normal maps.

Chapter 6, "Digital Painting Color Maps" In this chapter, you and your UV layout will venture beyond the borders of Blender into an external painting program and create the texture that will bring your zombie to life (or afterlife). Here you will learn some best practices for another important aspect of game art and digital painting and understand how this will help you build better game characters.

Chapter 7, "Rigging for Realistic Movement" In this chapter, you will learn to use Rigify, one of the more powerful plug-ins now available in Blender. This will help you make your zombie posable and capable of (re)animation. Like modeling, rigging is a step-by-step

process that will be covered here, from lining up the rig with the model to weight-painting the bones so they influence only certain parts of your zombie's body. Finally, you will take a brief moment to pose and render your model for future portfolio display.

Chapter 8, "Animating the Zombie" In this chapter, you will learn how the Blender animation system works and create three simple animations that will be used in your zombie game: idle, chase, and attack. These animations will be utilized later when you enter the Unity engine.

Chapter 9, "Unity Engine Basics" Your quest has now brought you to the Unity engine, where you will put your character in a level and fight hordes of zombies. Before you do this, however, you must learn how the engine works. This chapter introduces you to the interface and basic functions of the Unity game engine and shows you how to set up a simple prototype level with Unity's native primitive geometry.

Chapter 10, "Implementing Your Zombie in a Unity Game" In this final chapter, you will bring your zombie into Unity and utilize scripts to make him chase and attack you in large hordes. In this chapter you will also gain the weapons to defend yourself from the oncoming zombie horde and hopefully make it out alive.

How to Contact the Author

I'd love to hear back from you with feedback on the book or questions about the exercises, or even just to get in touch. I love hearing from others involved in gaming. You can reach me at `chris.totten1985@gmail.com`. You can also visit my website at `www.ChrisTotten3D.com` to find images from my latest projects.

Game Character Creation with Blender and Unity

CHAPTER 1

Basic Game Art Concepts

Some people don't want to take games seriously. This isn't a lament against parents lashing out over gore or people who think that games are a waste of time. It is instead a statement about a popular assumption that making video games doesn't involve a lot of work. This assumption is a problem in many game schools, whether they focus on the art, programming, or overall design of games.

In many ways, the art produced for a video game is not "art" in the traditional sense of "fine art"—content created solely for enjoyment by the senses or for aesthetic value. The art created for video games is, rather, part of the process of design or "applied art." But there are two important reasons why video game art is different from either animation or fine art. First, game art is interactive for a player. Second, game art is rendered in real time.

Many quests begin not with an instant foray into death and danger, but instead with an initial gathering of knowledge. Indeed, Sun Tzu's *The Art of War* does not begin by discussing battle right away, but with a chapter on planning. This is how we will approach your own quest to be a game character artist—and perhaps your leap from game consumer to creator.

This chapter covers the following topics:

- **Game design workflows**
- **Creating game assets**
- **Understanding and optimizing 3D game art**
- **Working with game engines**
- **Scripting happens**

Game Design Workflows

As a young field, game design has no prescribed "workflow" or "design method." This fact means that game studios can establish their own ways of working—within certain parameters, of course. Unless you are the rare person who can program like John Carmack and draw like da Vinci, you will most likely be working with a team. In many ways, this is one of the current strengths of the gaming industry: Game designers come from all fields and all walks of life. Consequently, team members have a broad range of influences to pull from that helps them create the best game possible.

Of course, games have to be fun. Game designers test this aspect with *playtesting*: inviting people to play early versions of a video game. Typically, games may see outside playtesters coming in at a stage of near completion known as the "beta" stage. At this point, many of the mechanics are already set in stone and the game has its artwork added to the engine. Because of this, some designers urge playtesting earlier in the process.

Other "set in stone" parts of the game design process involve the business end of things. This aspect is most directly embodied in the relationship between developers and publishers. The development team, which puts the game together, can consist of programmers, artists, writers, testers, producers, and composers—anyone involved in the legwork of making the game. Publishers oversee financial affairs, legal issues, public relations, and marketing of the game, turning it into a viable product.

Projects are often structured in milestone-based schedules, where the publisher pays the developer in phases. These phases generally include:

Concept Concept planning, budget, and contract negotiation

Preproduction Prototypes, design documentation, sketches, and basic design

Production

- **"Alpha" Development** Assets, levels, and early code
- **Beta/Quality Assurance** Playtesting with outside testers, utilizing beta code, and trying to reach launch or "gold" status by eliminating bugs
- **Gold** The point where the game is ready for mass production and retail and review copies are sent out to press

Beyond these phases, the developer has freedom to follow their own design methods as long as their obligations to the publishers are met. This has led to several distinctive design methods employed by studios; we'll look at some useful ones next. Try them in your projects and mix and match as you find appropriate.

Phase-Based Design

Phase-based design is a literal interpretation of the milestone-based schedule employed by many game publishers. It is also one of the most commonly used methodologies in the