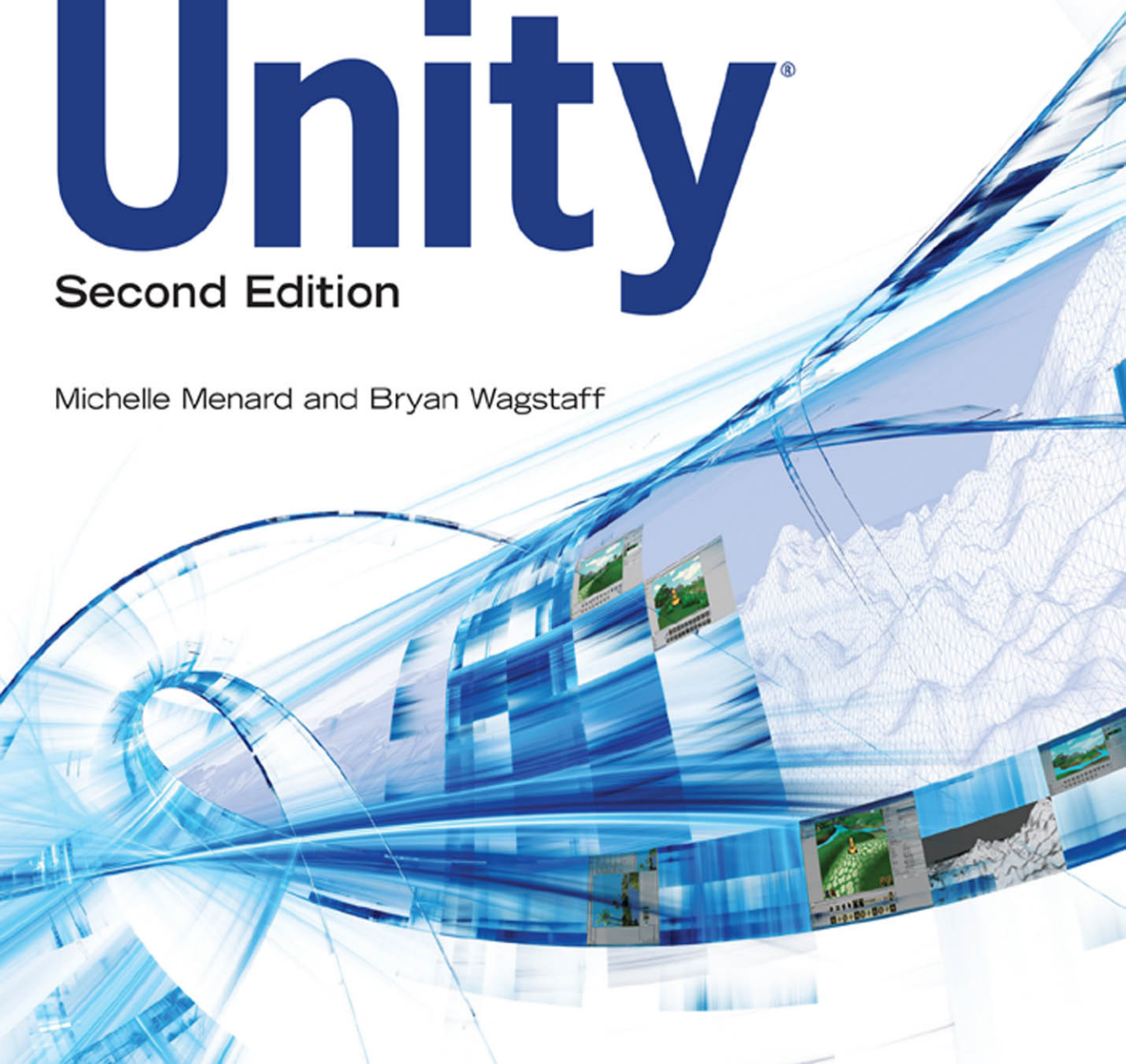



# Game Development with Unity®

## Second Edition

Michelle Menard and Bryan Wagstaff





# GAME DEVELOPMENT WITH UNITY<sup>®</sup>, SECOND EDITION

**MICHELLE MENARD  
AND  
BRYAN WAGSTAFF**

**Cengage Learning PTR**



Professional • Technical • Reference

Australia • Brazil • Japan • Korea • Mexico • Singapore • Spain • United Kingdom • United States

**Game Development with Unity®,  
Second Edition****Michelle Menard and  
Bryan Wagstaff****Publisher and General Manager,  
Cengage Learning PTR:** Stacy L. Hiquet**Associate Director of Marketing:**  
Sarah Panella**Manager of Editorial Services:**  
Heather Talbot**Senior Marketing Manager:**  
Mark Hughes**Senior Product Manager:** Emi Smith**Project Editor:** Kate Shoup**Technical Reviewer:** Michael Duggan**Copy Editor:** Kate Shoup**Interior Layout Tech:** MPS Limited**Cover Designer:** Mike Tanamachi**Indexer:** Larry Sweazy**Proofreader:** Sam Garvey

© 2015 Cengage Learning PTR.

CENGAGE and CENGAGE LEARNING are registered trademarks of Cengage Learning, Inc., within the United States and certain other jurisdictions.

ALL RIGHTS RESERVED. No part of this work covered by the copyright herein may be reproduced, transmitted, stored, or used in any form or by any means graphic, electronic, or mechanical, including but not limited to photocopying, recording, scanning, digitizing, taping, Web distribution, information networks, or information storage and retrieval systems, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without the prior written permission of the publisher.

For product information and technology assistance, contact us at  
**Cengage Learning Customer & Sales Support, 1-800-354-9706.**For permission to use material from this text or product, submit  
all requests online at **[cengage.com/permissions](http://cengage.com/permissions).**Further permissions questions can be emailed to  
**[permissionrequest@cengage.com](mailto:permissionrequest@cengage.com).**

Unity is a registered trademark of Unity Technologies. All other trademarks are the property of their respective owners.

All images © Cengage Learning unless otherwise noted.

Library of Congress Control Number: 2014939188

ISBN-13: 978-1-305-11054-0

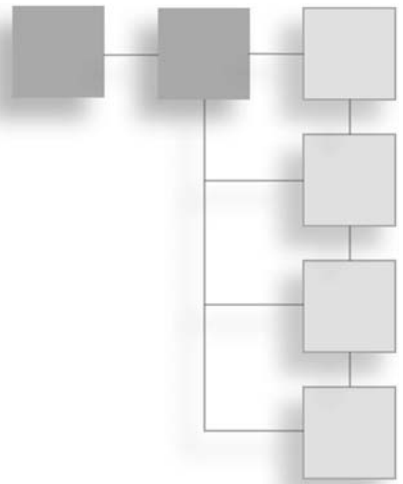
ISBN-10: 1-305-11054-4

eISBN-10: 1-305-11056-0

**Cengage Learning PTR**20 Channel Center Street  
Boston, MA 02210  
USACengage Learning is a leading provider of customized learning solutions with office locations around the globe, including Singapore, the United Kingdom, Australia, Mexico, Brazil, and Japan. Locate your local office at:  
**[international.cengage.com/region](http://international.cengage.com/region).**

Cengage Learning products are represented in Canada by Nelson Education, Ltd.

For your lifelong learning solutions, visit **[cengageptr.com](http://cengageptr.com).**Visit our corporate website at **[cengage.com](http://cengage.com).**



## ACKNOWLEDGMENTS

Revising this book has been quite an adventure. When Emi Smith of Cengage Learning contacted me about this project, I felt I could do the job easily. After all, I was comfortable with Unity, had a list of game credits, and had helped edit books in the past. It certainly loomed larger as time passed. The months revising this book and bringing it up to date have been an enjoyable challenge, and I could not have done it alone.

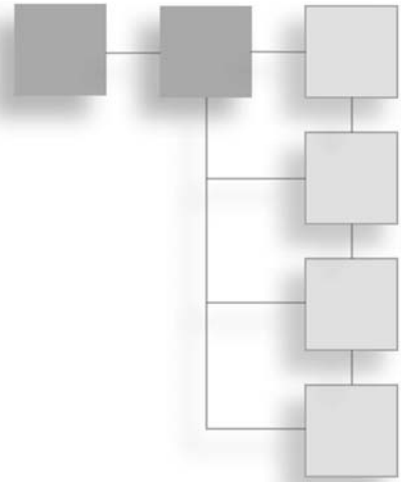
I need to thank the Unity team for their steady stream of updates and providing such a great engine. Unity is updated frequently. The first edition was written against early editions of the 3.x Unity series. Revising it meant every sentence needed to be verified against the current version, every screenshot updated, and every line of code validated. On finding any place that Unity's behavior had changed, sections needed to be rewritten. Unity's animation engine and particle system had been completely replaced, so the book required effort there. The number of supported platforms had grown from three in the first edition to over 15 potential platforms today. Even so, the book is already out of date. As this revision nears completion, the 5.x series is nearly here and will probably be released before this book. Thanks to all of you who continue to improve the tools.

I also need to thank Emi for putting up with me. Michael Duggan, Kate Shoup, Karen Gill, and the rest of the team have done an amazing job, and I have been impressed by their speed and professional skills. Then there are the people I've never met but still work hard to bring the book into reality; thank you to those behind the scenes at Cengage Learning. Next, my wife Sarah also deserves thanks for pushing through the days of

writer's block while putting her own book writing aside. Even though it was annoying at times, it could not have been finished without her reminders, "Turn that game off and go work on the book." And perhaps most importantly, thank you to the readers who will use this book. I hope you take what you learn, continue to grow, and develop the next generation of awe-inspiring entertainment.

—Bryan Wagstaff

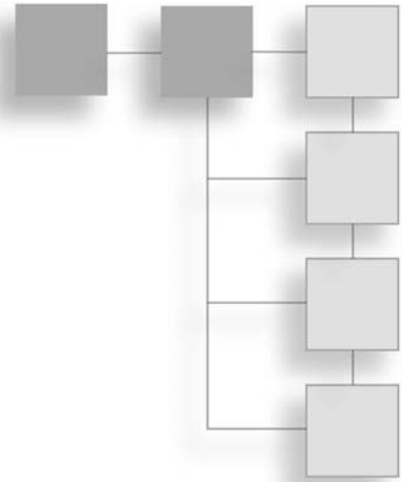
## ABOUT THE AUTHORS



**Michelle Menard** is a freelance writer and a game producer. After receiving a double bachelor of arts in applied mathematics and music from Brown University, she decided to jump into the games industry by getting a master of fine arts in game design from the Savannah College of Art and Design. She lives in Baltimore, MD, with her husband, two plants, and 3,000 pounds of yarn.

**Bryan Wagstaff** is a game programmer. He discovered his passion for programming in elementary school with “guess the number” style games and advanced from there. After earning a bachelor of science in computer science from Weber State University, and graduate studies in the 3D Graphics Lab at Brigham Young University, his professional career has included programming for video games, broadcast television, interactive meeting systems, and more. He currently lives in Salt Lake City, UT, with his wife, three daughters, and a flock of birds.

# CONTENTS



Introduction . . . . .	xiii
------------------------	------

**PART I      IN THE BEGINNING . . . . . 1**

**Chapter 1    An Overview of the Unity Engine . . . . . 3**

Getting Acquainted with the Interface . . . . .	3
The Project View. . . . .	5
The Hierarchy View . . . . .	8
The Inspector . . . . .	10
The Toolbar. . . . .	11
The Scene View. . . . .	12
The Game View . . . . .	23
The Animation and Animators Views . . . . .	26
The Profiler and Version Control . . . . .	27
Customizing the Editor. . . . .	28
Unity’s Basic Concepts . . . . .	28
Available Unity Licenses. . . . .	30

**Chapter 2    Your First Game: Where to Start? . . . . . 31**

Basic Design Theory . . . . .	31
Finding the Core Idea. . . . .	35
Brainstorming . . . . .	35
Researching Other Games . . . . .	36
Paper Prototyping: It’s Not Just for Business Software. . . . .	37

Planning It All Out . . . . .	38
A Basic Outline . . . . .	38
A Simple Level Document . . . . .	41
Getting Started . . . . .	43
<b>PART II ASSEMBLING THE GAME ASSETS . . . . .</b>	<b>45</b>
<b>Chapter 3 Setting the Stage with Terrain . . . . .</b>	<b>47</b>
Unity’s Terrain Engine . . . . .	48
Customizing Terrain . . . . .	52
Building Height Using a Heightmap . . . . .	52
Painting Height Using Brushes . . . . .	54
Painting Textures . . . . .	59
Placing Trees . . . . .	63
Cluttering It Up with Grasses and Detail Meshes . . . . .	67
Terrain Settings . . . . .	73
Lighting and Shadows . . . . .	75
Adding a Skybox and Distance Fog . . . . .	78
Adding Water to Your Terrain . . . . .	79
<b>Chapter 4 Building Your Environment: Importing Basic Custom Assets . . . . .</b>	<b>81</b>
Design First, Then Build . . . . .	81
Importing Textures . . . . .	82
More on Importing . . . . .	83
Supported Formats . . . . .	88
Importing Textures for <i>Widget’s</i> Terrain . . . . .	89
Importing Basic Meshes . . . . .	99
Setting Up Simple Shaders and Materials . . . . .	103
Unity-Provided Shaders . . . . .	105
Bumps, Spec, Cubes, and Details . . . . .	107
Assigning Shaders and Materials . . . . .	111
Making a Custom Skybox Material . . . . .	117
Adding Water . . . . .	119
Helpful Tips for Working with Assets . . . . .	121
Prefabs, Prefabs, Prefabs! . . . . .	122
Mass-Selecting and Grouping Objects . . . . .	123
Snapping to the Grid . . . . .	123
Reworking the Terrain . . . . .	124



<b>Chapter 5</b>	<b>Creating Characters</b> .....	<b>127</b>
	Basic PC 101 .....	127
	Character Capabilities in Unity .....	128
	Importing Characters and Other Non-Static Meshes .....	129
	Introducing Widget .....	129
<b>PART III</b>	<b>BRINGING YOUR PROPS TO LIFE WITH INTERACTIVITY</b> .....	<b>141</b>
<b>Chapter 6</b>	<b>Scripting in Unity</b> .....	<b>143</b>
	One Editor, Three Languages, a Whole Lotta Choice .....	143
	Picking a Script Editor—or, “Do You Want Autocompletion with That?” .....	145
	Fundamentals of Scripting in Unity .....	147
	Two Useful Items .....	147
	Variables .....	148
	Operators and Comparisons .....	159
	Conditionals .....	163
	Loops .....	167
	Functions .....	168
	Variable Scope .....	171
	Naming Conventions .....	172
<b>Chapter 7</b>	<b>Writing the Character and State Controller Scripts</b> .....	<b>173</b>
	Setting It Up and Laying It Out .....	173
	A Simple Third-Person Controller .....	174
	Controller Variables .....	176
	Unity’s <code>MonoBehaviour</code> Class .....	177
	Setting Up Unity’s Input Manager .....	183
	Hooking Up the Camera .....	189
	Updating the Character Controller .....	197
	Completed Scripts .....	198
<b>Chapter 8</b>	<b>Hooking Up the Animations</b> .....	<b>207</b>
	Animation in Unity .....	207
	Animation API .....	208
	The Mecanim Animation System .....	208
	The <code>Animation</code> Class .....	208
	Setting Up the PC’s Animations .....	211
	Defining the Problem .....	211
	Updating the Controller .....	211
	Creating the Animation State Manager .....	212

	Creating Animations Inside Unity . . . . .	216
	Some Basic Concepts . . . . .	216
	Animation View . . . . .	217
	Setting Up a New Animation Clip . . . . .	218
	Hooking It Up . . . . .	223
	Adding Animation Events . . . . .	224
	Completed Scripts . . . . .	226
<b>Chapter 9</b>	<b>Using Triggers and Creating Environment Interactions . . . . .</b>	<b>233</b>
	Triggers and Collision . . . . .	233
	Setting Up a Basic Trigger Object . . . . .	234
	Setting Up Other Kinds of Triggers . . . . .	242
	Completed Scripts . . . . .	248
<b>Chapter 10</b>	<b>Building Adversaries and AI . . . . .</b>	<b>255</b>
	Artificial Intelligence: Definitely Artificial, Not Much Intelligence . . . . .	255
	Some Simple AI Guidelines . . . . .	256
	A Simple Workflow . . . . .	258
	Setting Up a Simple Enemy . . . . .	259
	The AI Controller . . . . .	260
	A Simple State Manager for a Simple Bunny . . . . .	268
	Hooking Up Widget’s Attacks . . . . .	269
	Rewarding the Player for a Job Well Done . . . . .	272
	Spawning and Optimization . . . . .	273
	Completed Scripts . . . . .	275
<b>Chapter 11</b>	<b>Designing the Game’s GUI (Graphical User Interface) . . . . .</b>	<b>283</b>
	Basic Interface Theory . . . . .	283
	Steps of Interaction . . . . .	284
	Designing for Your Users . . . . .	284
	Unity’s GUI System . . . . .	286
	Buttons . . . . .	287
	Sliders . . . . .	287
	Labels and Boxes . . . . .	288
	Text Entry . . . . .	288
	Toggle . . . . .	289
	Toolbars and Selection Grids . . . . .	289
	Windows . . . . .	290

- A Custom Skin for *Widget* . . . . . 291
  - Creating the GUISkin . . . . . 292
  - Defining Custom Styles. . . . . 293
  - Importing New Fonts . . . . . 294
- Setting Up the HUD . . . . . 296
  - GUIContent(). . . . . 297
  - Character Displays . . . . . 301
  - Widget’s Character Display . . . . . 302
  - The Enemy’s Display Panel. . . . . 304
  - Resolution . . . . . 308
- A Sample Pop-Up Screen . . . . . 309
- Adding Full-Screen Menus . . . . . 313
- Completed Scripts. . . . . 316

**PART IV POLISH AND THE FINISHING TOUCHES . . . . . 327**

**Chapter 12 Creating Lighting and Shadows . . . . . 329**

- Types of Lights . . . . . 329
  - Light Properties . . . . . 331
  - Basics of Lighting . . . . . 333
- Lighting the World. . . . . 335
- Creating Shadows. . . . . 338
  - Lightmaps . . . . . 338
  - Projector-Made Shadows . . . . . 339
- Other Light Effects . . . . . 342
  - Lens Flares. . . . . 342
  - Cookies . . . . . 343

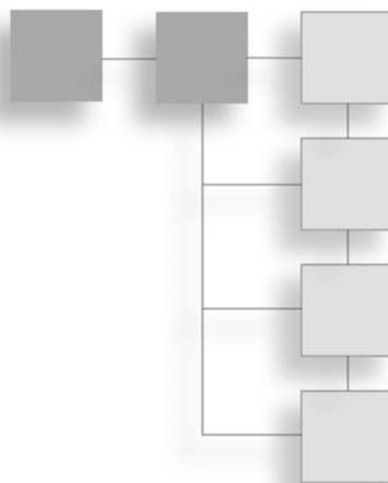
**Chapter 13 Using Particle Systems . . . . . 345**

- Particles: From Smoke to Stardust. . . . . 345
- Setting Up a Simple System . . . . . 347
  - Particle Systems. . . . . 347
  - Advanced Particle Systems. . . . . 352
- Particles for *Widget* . . . . . 353
  - Pickup Items . . . . . 353
  - Checkpoint Activation . . . . . 355
  - Widget’s Attack . . . . . 357
- Completed Scripts. . . . . 361

<b>Chapter 14</b>	<b>Adding Audio and Music</b> .....	<b>367</b>
	Feedback and Ambience .....	367
	Setting Up a Simple Audio Clip .....	369
	Ambient Sound Effects .....	371
	Controlling Sounds Through Scripts .....	372
	Adding Background Music .....	374
	The Whole Is Greater Than the Sum of Its Parts .....	375
	Completed Script .....	376
<b>PART V</b>	<b>PUBLISHING AND DISTRIBUTING BUILDS</b> .....	<b>379</b>
<b>Chapter 15</b>	<b>Basic Unity Debugging and Optimization</b> .....	<b>381</b>
	Debugging in Unity .....	381
	The Console .....	382
	The Log Files .....	383
	Optimization .....	384
	The Profiler .....	385
	Code Optimization .....	385
	Emulation .....	386
	Rendering Statistics Page .....	387
	Reducing File Size .....	388
	Other Ways to Optimize Graphics .....	389
<b>Chapter 16</b>	<b>Creating the Final Build</b> .....	<b>391</b>
	Prepping for the Build .....	391
	Setting Up the Player .....	391
	Finally, the Application Class .....	394
	Build Settings .....	394
	Other Build Features .....	396
	Asset Bundles .....	396
	Resource Folders .....	397
	Packing Up Assets for Later .....	397
	The End of the Road? .....	398
<b>PART VI</b>	<b>APPENDIXES AND OTHER RESOURCES</b> .....	<b>399</b>
<b>Appendix A</b>	<b>Shortcuts and Hotkeys</b> .....	<b>401</b>
<b>Appendix B</b>	<b>Common Classes</b> .....	<b>405</b>
	MonoBehaviour .....	405
	Transform .....	407

Rigidbody.....	408
CharacterController.....	408
Mathf.....	409
<b>Appendix C Going Forward.....</b>	<b>411</b>
Design Exercises.....	411
Scripting Exercises.....	412
Art and Animation Exercises.....	413
Audio Exercises.....	413
GUI Exercises.....	413
<b>Appendix D Resources and References.....</b>	<b>415</b>
Books.....	415
Programs.....	416
Scripting and General Unity Help.....	417
<b>Appendix E Glossary.....</b>	<b>419</b>
<b>Index.....</b>	<b>433</b>

# INTRODUCTION



First things first, welcome to the Unity Engine! Whether you're new to game development or a seasoned pro looking into new technology, the Unity Engine has a lot to offer. Available for Mac, Linux, and Windows, the engine can create games that can be deployed on just about any platform available, from the Web, to the Xbox and PlayStation (if you are a licensed developer), to mobile devices like smart phones and tablets. The easy interface, friendly development environment, and wide-ranging support of all popular gaming platforms make it a great choice for the student, indie, and larger developer team.

Unity's clients include such names as Ubisoft, Disney, and Electronic Arts, but the engine is also highly utilized by small independent studios, hobbyists, students, and even companies outside of the gaming industry for medical simulations and architectural walkthroughs. Whatever the end goal, Unity allows anyone, regardless of background, to create fun, interesting, and interactive content. Let's get started.

## WHAT WILL BE COVERED (AND WHAT WON'T)

This book is an introductory look into the engine. It explains what Unity has to offer and gives a few pointers on how to best use its capabilities for whatever it is you want to do. If you're a hobbyist or student, you'll probably want to start reading from the beginning and follow along with the example project. If you're using this book as a tool to evaluate whether the engine is right for you, you're probably best skipping around to the relevant chapters.

If you start from the beginning, you'll learn all the important interface commands, how to set up and organize your project, and all the basics of getting a 3D game up and running, from character importation to scripting to audio. After completing the sample project, you'll have all the skills necessary to go out and make your own games.

What this book *isn't* is a crash course in the Unified Theory of Game Development and Design. By that, I mean you won't be granted some mystical information or mad skills for everything there is to know in design, programming, art, or sound. Each topic covered (such as game design) does include some basic theory and information—enough to get you going on a working vocabulary and introductory concepts. This book won't make you a star designer or a world-class programmer, however. That requires years of study and practice.

If, after reading, you do find yourself interested in a particular field, check out Appendix D, “Resources and References,” for pointers on where to get more information. Think of this as a sampler course stretching across multiple cuisines, not an in-depth exploration of one particular food type. More advanced and singular topics such as network integration and discussions on Unity's shader language are also not covered.

## INTENDED AUDIENCE

So, who exactly is this book for, anyway? If you fall into any of the following categories, you've come to the right place:

- A solo developer or generalist looking for some well-rounded information on utilizing the engine
- A developer looking to evaluate the engine for use in future projects
- A hobbyist needing a how-to guide about some specific areas
- A student (or prospective student) wanting to know whether game development is right for you
- Anyone looking to build a game portfolio using an affordable (or in some cases free) professional engine

As stated, all the game development sections cover some basic background knowledge and go over a few key terms. However, the text does assume some knowledge or skills in a few areas if you plan to work away from the sample project. For example, creation of 3D art assets and how to use a 3D modeling package are not covered. All the required models used in the text (and then some) are included on the companion website (more on that

in a moment), but their creation is not described. If you stick to the sample project while reading the book, you won't really need any outside knowledge or skills (although any game development information is a plus). If you plan to work on your own project from the start using this book as a guide, then you'll need to educate yourself in the other areas of development or find other places and people to provide art and code. If creating models and animations isn't your thing, the Unity Store has a wide range of assets for free and for sale, and there are many communities out there who can help you develop your own.

## THE BOOK'S STRUCTURE

The information in the book is organized into five parts, each covering a general aspect of game development. Within each part, chapters are devoted to each single concept, such as one chapter for AI development and another for particle effects. If you need help on a specific area of Unity, go to the corresponding chapter or use the handy index. The appendixes include a list of common and helpful shortcut keys, a rundown of the most-used classes, and exercises for you to complete if you want a few pointers on what to do once you finish reading. A compiled glossary for all keywords introduced in the text is also contained there.

I've tried to make learning the engine a little more straightforward by using some general formatting guidelines. Steps for you to complete in the engine always appear in numbered lists. If you see such lists coming up on the page, you should open Unity to follow along.

Links between steps in a folder chain or nested menu are marked with the > symbol. So the line "My Documents > My Unity Project" would mean to open the My Documents folder and then open the folder My Unity Project contained within it. Pretty straightforward. Code to write in the engine is blocked off in its own formatting, as shown here:

```
//I'm a comment
Update()
{
    print("Hello World");
}
```

Finally, some extra information is included in the form of sidebars. These mostly cover more advanced technical data or engine specs and aren't required knowledge for using the engine on a day-to-day basis. They do tend to be helpful, however. Also be mindful of tips, notes, and warnings scattered throughout the text. These are often important, containing information about common pitfalls and helping to stave off potentially hard-to-fix



disasters. If time was taken to graphically embellish something, it's probably worth a second look.

## INSTALLATION INSTRUCTIONS

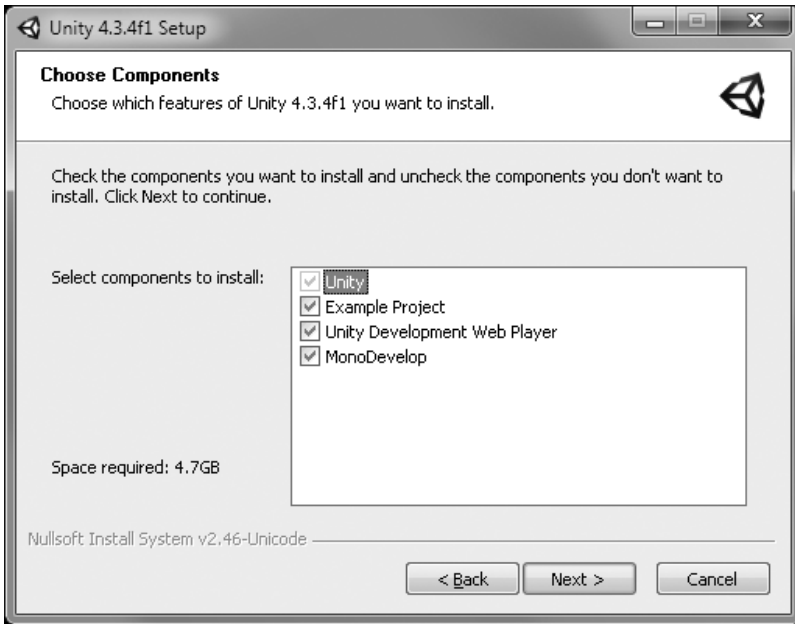
Installing Unity is quick and painless and technically requires an Internet connection. Unity comes in two flavors, Unity Basic, which is free, and Unity Pro. Both are regularly updated by developers. Although you won't need the Internet again after activation, it is advisable to have a connection if only for the patch updates and fixes.

### The Unity Engine

First up, install Unity. You can download Unity from the Unity Technologies website: [unity3d.com/](http://unity3d.com/). From the Download menu, click on either the Mac or Windows version button, whichever is right for you. (This book uses the Windows version for all its examples.) You can choose to download the free Unity Basic version directly or get a Unity Pro license trial version, which is free for 30 days. It doesn't really matter as far as the book is concerned, but it can be fun to see what goodies the Pro version includes.

Note that Unity has grown to include many features and these features require space. The installer is about 1 GB. The download time will depend on your Internet connection. A fast broadband connection can usually handle the transfer in a few minutes.

Once the download is complete, run the `UnitySetup-###.exe` file, accept the terms of agreement, and follow the onscreen command prompts. When you get to the Choose Components screen, shown in Figure I.1, make sure the Example Project checkbox is selected. You'll probably also want to select the Unity Development Web Player checkbox as well, in case you ever want to publish your games to the Internet. In addition, consider selecting the MonoDevelop checkbox even if you have your own development environment because the bundled version of the MonoDevelop editor has some specialized features that are hard to get in other editors. If you change your mind later you can re-run the installer to add or remove components. Then click Next and finish the installation.



**Figure I.1**

All components are selected by default, requiring nearly 5 GB of space.

Source: Unity Technologies.

Use the default install path or select your own, and then click Install. Unity takes nearly 5 GB of install space, so make sure your selected destination can handle this plus any other add-ons or projects you want to use later. Follow the other onscreen instructions to complete the install.

After the install has finished, Unity will prompt you to register your copy. For the free version and trial Pro, this is easy. Select the Internet registration version (if available) and fill out the form on the website the engine takes you to—usually it's just your name and email address. After this, Unity is yours to use.

Once the engine has finished installing, it's time to move on to the contents of the companion website.

## Using the Companion Website

To access the book's companion website, visit [www.cengageptr.com/downloads](http://www.cengageptr.com/downloads), and type the name of this book in the Search field. The companion website is divided into a few main sections:

- **Chapters:** This folder contains subfolders for each chapter in the book, whenever they require the use of files or assets. You can either copy the entire Chapters folder to your hard drive now or just grab the individual files when you need them. The text always specifies when a file is needed and where to grab it.
- **Design Documents:** This folder houses all the basic information for the sample project discussed in the text, a game called *Widget*. When the text says to view the Design Docs, they're located in here.
- **Shader Test:** A sample project detailing and comparing the basic shaders side-by-side that are available in Unity. If you're not sure which shader to use or how some shaders may interact in a specific lighting rig, modify and use this file as needed.
- **Final Project Files:** Unlike the Chapters folder, which houses all the individual files as they come up in the text, the Final Project Files folder is a complete Unity project for the *Widget* game. If you ever get stuck or want to see how something fits together later, you can always check out the game here. Extra assets such as more models, textures, and UI elements are also included here for any further expansion you may want to pursue.

## Optional Installs

Between Unity and the contents of the companion website, you can complete every exercise in this book and get the sample project up and running. However, you may find yourself wanting to tweak a graphic or texture here or there, or maybe even sculpt a new model to import. Many free software packages are described both in the text where appropriate and in Appendix D. If you don't already have something installed on your computer, check the appendix for information and a link.

Unity includes MonoDevelop, a free code editor to use for scripting, but you can use your own favorite coding environment if desired. Chapter 6, "Scripting in Unity," covers compatible ones in more detail, and Appendix D also provides links where appropriate.

## PARTING WORDS OF WISDOM

You probably have tons of great game ideas floating around in your head. Maybe you've even started working on one, the big one that'll net you that dream job or needed raise. Maybe you have built some small projects in the past and want to use a comprehensive engine rather than doing everything yourself. Or perhaps you've started two games, or three, or...you get the picture. Working on making your snippets of ideas and musings into playable games is great—but how many have you actually finished? Making a game is a huge commitment, fraught with tons of unforeseen setbacks, design changes, and software explosions, all for a tiny little bundle of ideas that you hope others will love as much as you do. It's far too easy once the first dragon rears its head to stop working, take an extended break, and never return to the field to try again. It's not procrastination, you tell others, it's just a short time away to rest your eyes, to let your ideas simmer free of worry. The short break becomes a week, and then a month, and years roll by as your little unfinished game collects dust in the corner.

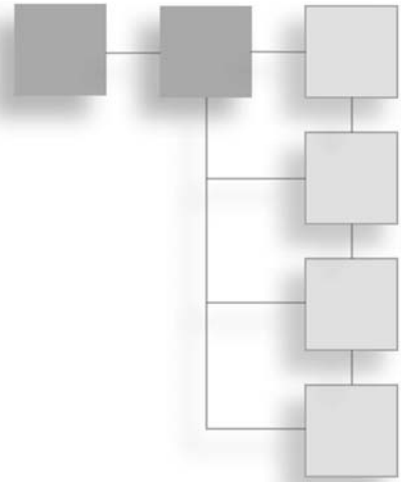
Tackling the problems in game development is hard work. Sometimes you follow a trail of ideas and discover barriers that need to be climbed, worked around, or pushed through. Other times you need to admit that the idea was wrong, it didn't work out as expected, or it's just plain un-fun. Don't walk away from the whole project and leave the game unfinished. Try something else, even if you're unsure where this new path will take you. Maybe it won't work out, but maybe it'll be the solution to a whole host of problems. Try, make mistakes, learn, and try again. Be willing to make mistakes and learn from them, and keep fighting your way through until the game is complete. No game is perfect and defect-free, but you can reach the point where you can say it is complete.

Finish that game, and then finish another. It doesn't matter if you think they're horrible, terrible piles of swill you'd be embarrassed to show your own mother. Show her anyway. They may suck, or they may not. Either way, analyze what you did that worked and didn't work. The process is as much about learning as it is about creating, and no one ever excelled by stopping halfway. If you get discouraged, it is okay to take a break, but always, always come back to it in the end. Support from friends can help greatly in this—involve some of your buddies in rounds of routine play tests. Make a party of it. Celebrate what you do and remember to have fun. If you're not having fun, you're probably not making fun!

One late night, as we were pushing to finish a game, I was sitting in the cafeteria, dining with co-workers on a large tray of studio-provided “mystery nuggets” for dinner. Someone said, “It is eight o’clock at night and I’m still at work on this crazy project that never seems to end. I could be spending time on so many other things. Will you remind me why we are still here?” Another co-worker spoke up, and his reply was heartfelt: “Because we love it. No matter how many problems we encounter and difficulties we overcome, we love it. We love creating these amazing games that entertain millions of people. We are artisans, passionate about doing our best on every task. We are not just working late and eating deep-fried mystery nuggets; we are building a game to inspire, to entertain, and to give the world something that they may savor and enjoy. Just like the mystery nuggets, most people will have no idea what is inside, but when they play the game they can be entertained and enjoy a moment of life. That makes the difficulties worthwhile.”

Be passionate. Development is work and sometimes it is difficult. Sometimes it is painful. Sometimes you will be disappointed and unsatisfied with the results. Just keep pushing, sharing your passion, and doing your best work. Game development is a powerful career. We create new worlds, teach, entertain, and inspire. We start with a blank file and finish with a product that can change the world. Be a game developer.

# PART I



## IN THE BEGINNING ...

As with any new endeavor, it's usually best to start somewhere in the beginning. And before you even start, it's best to lay out your tools and get your ideas in order. You wouldn't attempt (I hope) to build a house without a blueprint, and games are no different. The best game ideas in the world won't get you very far unless you have the knowledge, skills, and discipline to see them fully realized and implemented.

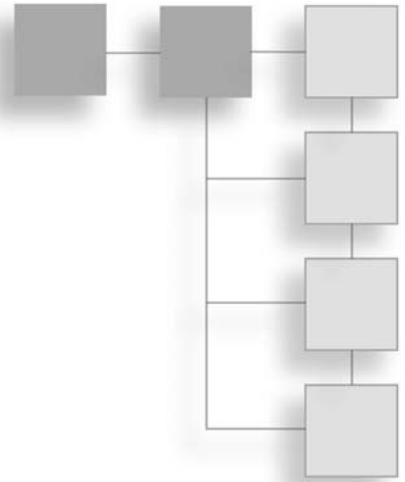
Before Unity, making a game from scratch for the newbie game designer was a rather daunting process. Engines, especially free ones, weren't terribly easy to come across, and those that were often suffered from poor execution or lack of documentation. Now with Unity, you can quickly get your ideas in motion, even if you lack a strong art or programming background.

In this part, you'll learn the basics of the engine and its interface, as well as how to refine your game idea from the get-go, hopefully saving you some time and energy later in the process.

*This page intentionally left blank*

# CHAPTER 1

## AN OVERVIEW OF THE UNITY ENGINE



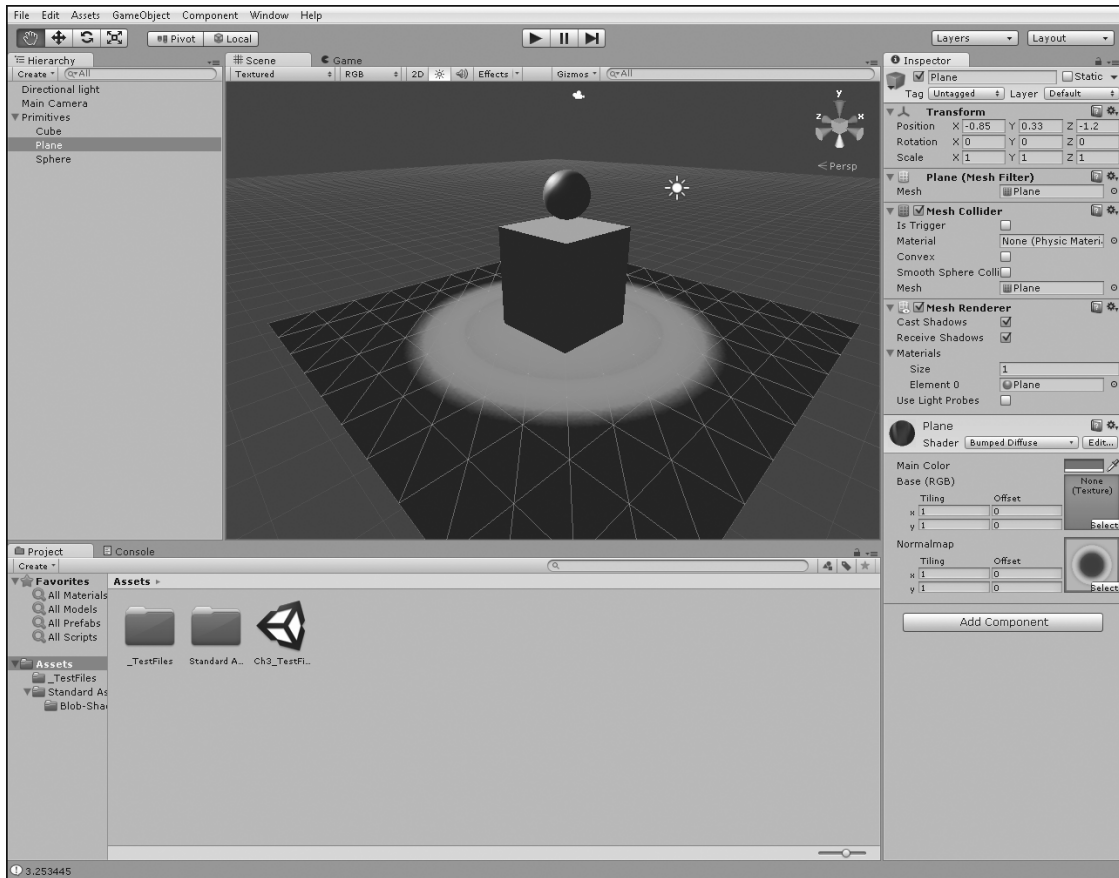
Unity is a powerful integrated game engine and editor, enabling you to quickly and efficiently create objects, import external assets, and link them all together with code. The editor is visually driven and built around the principles that you can do everything with a simple drag-and-drop motion—even connect scripts, assign variables, or create complicated multi-part assets. Unity also boasts an integrated scripting environment, built-in networking capabilities, and the ability to build and deploy for multiple platforms. All of this is wrapped up in a simple, intuitive, and customizable workspace.

### GETTING ACQUAINTED WITH THE INTERFACE

Before diving headfirst into making your first game with Unity, it's worth the time to first take a quick look at how to get around in the editor. Unity may look a bit daunting when you first open it, but you'll find that you can easily master its basics in a day.

If you haven't done so already, open Unity. Choose Start > Programs > Unity (C:\Program Files\Unity\Editor\Unity.exe) or click the desktop icon if you created one. Then load the Ch1\_TestFile.unity file located on this book's companion website to follow along. Figure 1.1 shows the Unity environment.





**Figure 1.1**

The Unity environment after you load the Chapter 1 sample. By the end of the chapter, you will understand all the parts you see on the screen.

Source: Unity Technologies.

## Note

---

If you installed one of the demos with the engine, a scene from the demo will load by default until you either load a different scene or create a new one.

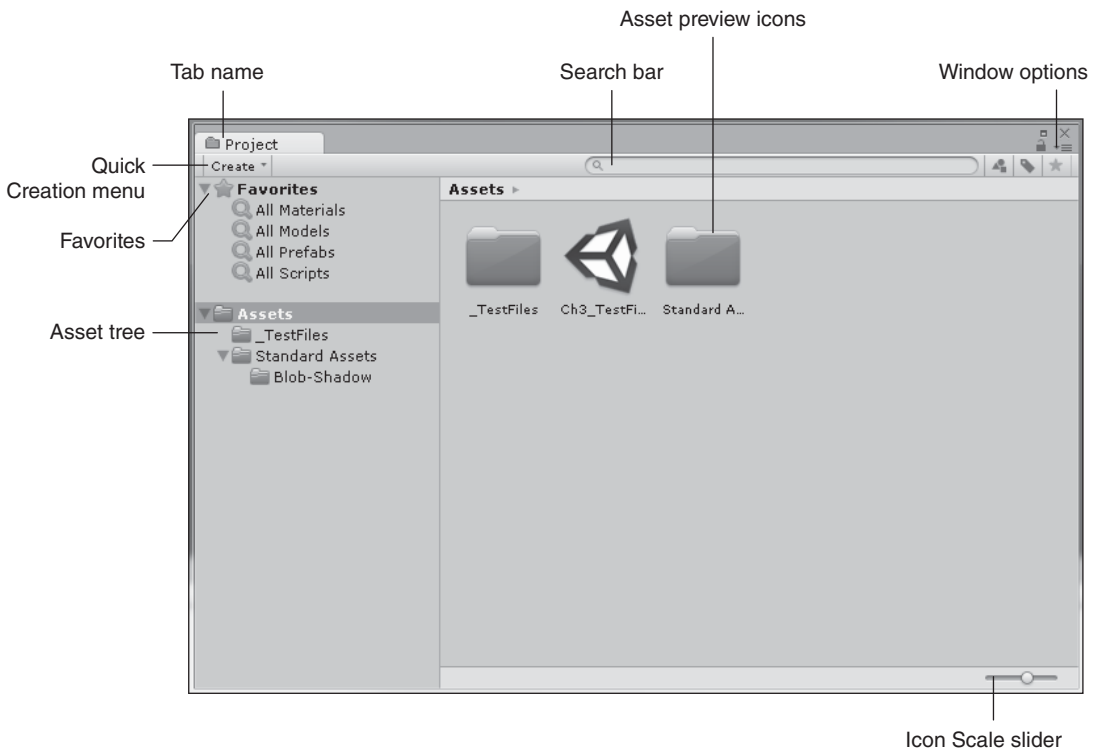
---

The editor's default layout is broken into a series of different panes and tabbed windows called views. Each view details a specific aspect of the editor and allows you to perform different functions when working on your game. If you've used a 3D modeling program or other game editor in the past, you may find some of this familiar.

You can rearrange the views to suit your preferences. Views can be repositioned, resized, and even broken off to their own separate windows. Several presets can be found in the menu under Window > Layouts. Depending on the size of your monitor, you might prefer the Wide layout or the Tall layout, or you might decide to create your own. Don't worry about getting the settings wrong; you can always reset the Unity views back to the default layout by selecting Window > Layouts > Revert Factory Settings.

## The Project View

All of a game's files—scripts, objects, scenes, anything and everything—are organized into a Project folder, each of which contains an Assets folder. The Assets folder houses *everything* you've created or imported to include in your game—meshes, textures, scripts, cameras, levels... everything. (See Figure 1.2.) The Project View panel displays this Project folder and the incorporated Assets folder.



**Figure 1.2**  
The Project view.

Source: Unity Technologies.

The Project view displays everything included in the game's Assets directly, exactly how they are organized and arranged on your computer's hard drive. If you are unsure, however, of where these files are located (or if you ever forget), you can simply right-click any selected asset in the Project view and select Show in Explorer.

Arrows next to folders indicate nested layers. Clicking any one of these will expand that folder's contents. You can also Shift-click an arrow to fully expand or contract its contents. Moving and organizing files into different folders can be achieved in the Project view with a simple click and drag.

### Caution

---

Be careful about moving your asset files around outside the Unity Editor. In fact, avoid it at all costs. If you need to reorganize or move an asset, do it from within the Project view. Not doing so could break or remove any metadata or links associated with that asset, possibly breaking your game in the process.

---

When you select an asset folder or a Favorites folder, the preview icons update to show you the folder's contents. Use the Icon Scale slider to change the size of the icons. If you slide the Icon Scale slider all the way to the left, the display will change from icons to a list.

Each type of object listed also has its own descriptive icon or thumbnail, making it easy to quickly scan the contents. Many assets, like directories, source code, and general data files, use a standard icon. Other assets, like Photoshop images, 3D models, textures, and materials, display a thumbnail preview of the asset in the file.

The Search bar enables you to quickly find assets directly by name. The three buttons on the right side of the Search bar let you search by type, search by tag, and save the search as a favorite. On a small project, you can usually remember where all your assets are located. But when a project contains hundreds or even thousands of assets, using the search tool is the easiest way to find what you need. The Project view's list will dynamically update after each letter you type, allowing for easier browsing if you don't quite remember the asset's exact name.

You can open and edit files directly from within the Project view. If you find you need to tweak or correct something in any file (like a Photoshop file), simply double-click the file to open it in its default editor. Save the file normally to have Unity import it back into your project.

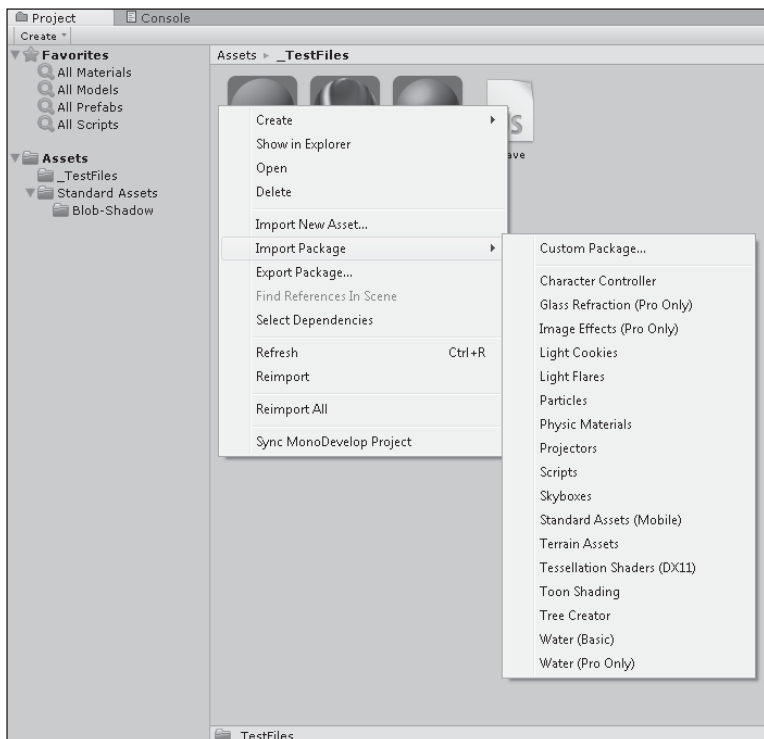
Sometimes you'll find it necessary to make new assets or objects that aren't contained already in your Project view. Unity makes this easy from within this view—simply click the Quick Creation menu to bring up the available options. This shortcut menu is located

right beneath the Project tab. From here, you can make new folders, empty script files, or other game-specific objects without having to leave the editor. Right-clicking in the Project view itself will also give you a link to the Quick Creation menu and its options, placing the new file at your current location in the file tree.

### Tip

To rename any file or folder, you can click twice slowly on the name (not a normal fast double-click) or select the desired file and press the F2 key. Then start typing. Press Enter when you are done renaming your file.

Right-clicking in the Project view will bring up a few advanced options, including asset importation, syncing with external project controllers, and asset package manipulation. See Figure 1.3. These are covered in more depth in later chapters.



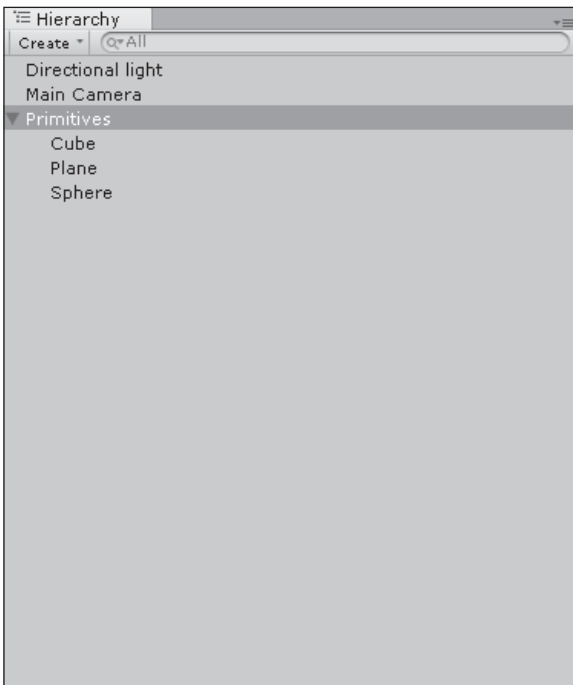
**Figure 1.3**  
The right-click menu and Import Package menu options.

Source: Unity Technologies.

All tabbed windows have a Windows Options drop-down list, allowing you to maximize the selected view, close the viewed tab, or add another tab view to the window. Click the icon to bring up the available options.

## The Hierarchy View

Whereas the Project view lists all the objects and files available in your game, the Hierarchy view lists just the ones you're actually using in the current scene. The objects in the scene are listed alphabetically. As you add or remove objects from your game, the Hierarchy view will update with each change. Selecting an object in the Hierarchy view and pressing the Delete key (or right-clicking and selecting Delete) will remove the object from your current scene in the game, but not from the project's Assets folder. Figure 1.4 shows the game's current contents as an example.



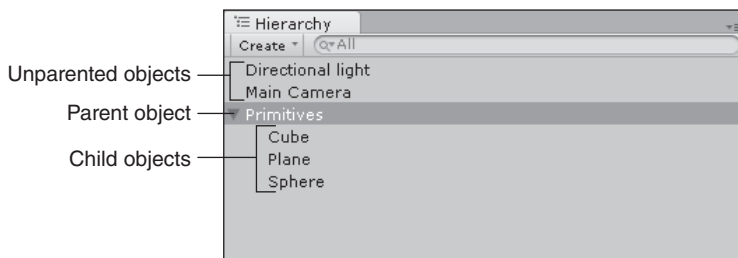
**Figure 1.4**  
The Hierarchy view, showing the game scene's current contents.

Source: Unity Technologies.

Each *instance* (a copy or occurrence) of an asset will be listed individually, so good naming conventions are especially important. If you have 30 instances of an object all named Cube,

you may have trouble finding the one you want later. You can rename any object in the Hierarchy view independently of its actual filename in the Project view. A simple mesh named Cube in your Project view can then be instantiated and renamed in the Hierarchy view to anything you want, like Crate, Box, or Mystery Pickup23, making it easier to find and use later. Note that this will not update the filename of the actual object in the Project view or on your computer. To do that, you must change the name from within the Project view.

*Parenting* objects together in the Hierarchy view can also help with organization and make editing your game easier. When you parent objects together, you are basically linking a collection of unrelated objects together in a group under a single object, the parent. All the objects under this parent are called its children, or child objects. See Figure 1.5.



**Figure 1.5**  
Parented and unparented objects.

Source: Unity Technologies.

In the example, the parent object is an asset called Primitives, under which three child objects are located: Sphere, Cube, and Plane. Clicking the arrow next to Primitives will expand or collapse the group, much like with the folders in the Project view. However, parenting gives you one other important benefit besides a speedy way to group like objects together: Moving or manipulating the parent object will in turn do the same to all the children underneath it. They are said to inherit the parent's data. The child objects can still be edited independently of each other and the parent object, giving you more control.

If you're still uncertain about parenting, think of a normal person's body. An arm is parented to the torso, and a hand is parented to the end of the arm. Moving the torso forward (the parent) will move the arm with it, which will in turn move the hand (the two children). However, you can move and rotate the hand around without moving the torso or the arm.

Using parented objects can make moving large numbers of objects around much easier and more precise, and should be used whenever possible. A few more advanced concepts of parenting will be covered in Chapter 4, "Building Your Environment: Importing Basic Custom Assets" and Chapter 6, "Scripting in Unity."