Praise for *Sams Teach Yourself Unity Game Development in 24 Hours, Second Edition*

“Rapid prototyping is one of the most valuable skills in the industry, and this book will help you get up and running with enough time left over to finish a weekend game jam. Despite being a long time Unity user, I learned a dozen new time-saving tricks in the first half of this book alone!”

—Andy Moore, Captain, Radial Games

“24 hours, 3 games, and a plethora of lessons on not only how to build games in Unity but how to be a game designer, programmer, and developer. *Sams Teach Yourself Unity Game Development in 24 Hours, 2/e* is a great foundation for budding game builders.”

—Tim J. Harrington, EdD, Higher Education Games and Social Learning Specialist

“*Sams Teach Yourself Unity Game Development in 24 Hours, 2/e* provides a terrific and thorough introductory look at the Unity development environment, game terminology, and game-making process, with plenty of hands-on examples, exercises and quizzes that will have readers creating their own games in no time!”

—Dr. Kimberley Voll, Game Developer/Researcher, ZanyT Games

“This is the book we have been waiting for! Ben and Mike don’t just explain how to use Unity, they explain how to use it properly so you won’t get stuck later. Every Unity developer should carry this around in their back pocket.”

—Efraim Meulenberg, Co-Founder, TornadoTwins

“Unity’s fun to play with and fun to learn. It’s become extremely popular as a platform for game studios ranging in size from one to one hundred people. Game engines are only as good as the games they enable; as a developer you need to ship games. That’s where this book will help you. I especially enjoyed the starter 2D and 3D games developed in this book. They gather the material learned in previous chapters and show you how the parts fit together into a working whole. Reading this book will inspire you to create your own experiences and share them with the world.”

—Jeff Somers, Developer on Rock Band, Guitar Hero, Phase and Dance Central

“This book will make all of your dreams come true, provided your dreams exclusively revolve around game development in Unity. Plus, I’m British, so it must be true.”

—Will Goldstone, Unity Technologies

“*Sams Teach Yourself Unity Game Development in 24 Hours, 2/e* is a comprehensive primer for learning Unity3D akin to eating dessert first—you get to the fun quickly!”

—Elliott Mitchell, Co-founder, Vermont Digital Arts/Boston Unity Group
Sams Teach Yourself
Unity® Game Development
Second Edition
in 24 Hours
Contents at a Glance

Preface ......................................................................................... xiii
HOUR 1 Introduction to Unity ....................................................... 1
HOUR 2 Game Objects .................................................................... 21
HOUR 3 Models, Materials, and Textures ...................................... 35
HOUR 4 Terrain .............................................................................. 49
HOUR 5 Environments .................................................................. 63
HOUR 6 Lights and Cameras .......................................................... 81
HOUR 7 Game 1: Amazing Racer .................................................. 103
HOUR 8 Scripting—Part 1 ............................................................... 119
HOUR 9 Scripting—Part 2 ............................................................... 141
HOUR 10 Collision ....................................................................... 161
HOUR 11 Game 2: Chaos Ball ....................................................... 173
HOUR 12 Prefabs ........................................................................... 189
HOUR 13 2D Games Tools .............................................................. 201
HOUR 14 User Interfaces ............................................................... 217
HOUR 15 Game 3: Captain Blaster ................................................ 237
HOUR 16 Particle Systems ............................................................. 257
HOUR 17 Animations ................................................................... 275
HOUR 18 Animators ..................................................................... 291
HOUR 19 Game 4: Gauntlet Runner .............................................. 317
HOUR 20 Audio ............................................................................ 339
HOUR 21 Mobile Development .................................................... 353
HOUR 22 Game Revisions ............................................................. 365
HOUR 23 Polish and Deploy .......................................................... 379
HOUR 24 Wrap Up ......................................................................... 393

Index ............................................................................................ 399
# Table of Contents

Preface xiii

**HOUR 1: Introduction to Unity** 1
- Installing Unity ........................................ 1
- Getting to Know the Unity Editor .......... 4
- Navigating the Unity Scene View .......... 17
- Summary .............................................. 19
- Q&A .................................................. 19
- Workshop .......................................... 19
- Exercise ........................................... 20

**HOUR 2: Game Objects** 21
- Dimensions and Coordinate Systems ...... 21
- Game Objects .................................... 25
- Transforms ....................................... 26
- Summary .......................................... 33
- Q&A .................................................. 33
- Workshop .......................................... 33
- Exercise ........................................... 34

**HOUR 3: Models, Materials, and Textures** 35
- The Basics of Models ......................... 35
- Textures, Shaders, and Materials .......... 41
- Summary .......................................... 46
- Q&A .................................................. 46
- Workshop .......................................... 47
- Exercise ........................................... 47

**HOUR 4: Terrain** 49
- Terrain Generation ............................ 49
- Terrain Textures ................................. 57
- Summary .......................................... 61
- Q&A .................................................. 61
## HOUR 12: Prefabs 189

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prefab Basics</td>
<td>189</td>
</tr>
<tr>
<td>Working with Prefabs</td>
<td>192</td>
</tr>
<tr>
<td>Summary</td>
<td>198</td>
</tr>
<tr>
<td>Q&amp;A</td>
<td>198</td>
</tr>
<tr>
<td>Workshop</td>
<td>198</td>
</tr>
<tr>
<td>Exercise</td>
<td>199</td>
</tr>
</tbody>
</table>

## HOUR 13: 2D Games Tools 201

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Basics of 2D Games</td>
<td>201</td>
</tr>
<tr>
<td>Orthographic Cameras</td>
<td>204</td>
</tr>
<tr>
<td>Adding Sprites</td>
<td>205</td>
</tr>
<tr>
<td>Draw Order</td>
<td>209</td>
</tr>
<tr>
<td>2D Physics</td>
<td>212</td>
</tr>
<tr>
<td>Summary</td>
<td>214</td>
</tr>
<tr>
<td>Q&amp;A</td>
<td>215</td>
</tr>
<tr>
<td>Workshop</td>
<td>215</td>
</tr>
<tr>
<td>Exercise</td>
<td>215</td>
</tr>
</tbody>
</table>

## HOUR 14: User Interfaces 217

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic UI Principles</td>
<td>217</td>
</tr>
<tr>
<td>The Canvas</td>
<td>218</td>
</tr>
<tr>
<td>UI Elements</td>
<td>223</td>
</tr>
<tr>
<td>Canvas Render Modes</td>
<td>230</td>
</tr>
<tr>
<td>Summary</td>
<td>232</td>
</tr>
<tr>
<td>Q&amp;A</td>
<td>233</td>
</tr>
<tr>
<td>Workshop</td>
<td>233</td>
</tr>
<tr>
<td>Exercise</td>
<td>233</td>
</tr>
</tbody>
</table>

## HOUR 15: Game 3: Captain Blaster 237

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design</td>
<td>237</td>
</tr>
<tr>
<td>The World</td>
<td>238</td>
</tr>
<tr>
<td>Controls</td>
<td>247</td>
</tr>
<tr>
<td>Improvements</td>
<td>255</td>
</tr>
<tr>
<td>Summary</td>
<td>255</td>
</tr>
<tr>
<td>Q&amp;A</td>
<td>255</td>
</tr>
</tbody>
</table>
The Controls .................................................. 329
Room for Improvement .................................................. 336
Summary .................................................. 336
Q&A .................................................. 336
Workshop .................................................. 336
Exercise .................................................. 337

**HOUR 20: Audio** .................................................. 339
Audio Basics .................................................. 339
Audio Sources .................................................. 341
Audio Scripting .................................................. 346
Summary .................................................. 349
Q&A .................................................. 349
Workshop .................................................. 349
Exercise .................................................. 350

**HOUR 21: Mobile Development** .................................................. 353
Preparing for Mobile .................................................. 353
Accelerometers .................................................. 357
Summary .................................................. 361
Q&A .................................................. 362
Workshop .................................................. 362
Exercise .................................................. 362

**HOUR 22: Game Revisions** .................................................. 365
Cross-Platform Input .................................................. 365
Amazing Racer .................................................. 368
Chaos Ball .................................................. 372
Captain Blaster .................................................. 374
Gauntlet Runner .................................................. 375
Summary .................................................. 376
Q&A .................................................. 376
Workshop .................................................. 376
Exercise .................................................. 377
HOUR 23: Polish and Deploy  379
  Managing Scenes ................................................................. 379
  Persisting Data and Objects .................................................. 381
  Unity Player Settings ......................................................... 384
  Building Your Game .......................................................... 387
  Summary ........................................................................... 391
  Q&A ................................................................................ 391
  Workshop ........................................................................ 391
  Exercise ........................................................................... 392

HOUR 24: Wrap Up  393
  Accomplishments .............................................................. 393
  Where to Go from Here ...................................................... 395
  Resources Available to You ................................................ 396
  Summary ........................................................................... 397
  Q&A ................................................................................ 397
  Workshop ........................................................................ 397
  Exercise ........................................................................... 398

Index  399
Preface

The Unity game engine is an incredibly powerful and popular choice for professional and amateur game developers alike. This book has been written to get readers up to speed and working in Unity as fast as possible (about 24 hours to be exact) while covering fundamental principles of game development. Unlike other books that only cover specific topics or spend the entire time teaching a single game, this book covers a large array of topics while still managing to contain four games! Talk about a bargain. By the time you are done reading this book, you won’t have just theoretical knowledge of the Unity game engine. You will have a portfolio of games to go with it.

Who Should Read This Book

This book is for anyone looking to learn how to use the Unity game engine. Whether you are a student or a development expert, there is something to learn in these pages. It is not assumed that you have any prior game development knowledge or experience, so don’t worry if this is your first foray into the art of making games. Take your time and have fun. You will be learning in no time.

How This Book Is Organized and What It Covers

Following the Sam’s Teach Yourself approach, this book is organized into 24 chapters that should take approximately 1 hour each to work through. The chapters include the following:

- Hour 1, “Introduction to Unity”—This hour gets you up and running with the various components of the Unity game engine.
- Hour 2, “Game Objects”—Hour 2 teaches you how to use the fundamental building blocks of the Unity game engine—the game object. You also learn about coordinate systems and transformations.
- Hour 3, “Models, Materials, and Textures”—In this hour, you learn to work with Unity’s graphical asset pipeline as you apply shaders and textures to materials. You also learn how to apply those materials to a variety of 3D objects.
Hour 4, “Terrain”—In Hour 4, you learn to sculpt game worlds using Unity’s terrain system. Don’t be afraid to get your hands dirty as you dig around and create unique and stunning landscapes.

Hour 5, “Environments”—In this hour, you learn to apply environmental effects to your sculpted terrain. Time to plant some trees!

Hour 6, “Lights and Cameras”—Hour 6 covers lights and cameras in great detail.

Hour 7, “Game 1—Amazing Racer”: Time for your first game. In Hour 7, you create Amazing Racer, which requires you to take all the knowledge you have gained so far and apply it.

Hour 8, “Scripting Part 1”—In Hour 8, you begin your foray into scripting with Unity. If you’ve never programmed before, don’t worry. We go slowly as you learn the basics.

Hour 9, “Scripting Part 2”—In this hour, you expand on what you learned in Hour 8. This time, you focus on more advanced topics.

Hour 10, “Collision”—Hour 10 walks you through the various collision interactions that are common in modern video games. You learn about physical as well as trigger collisions. You also learn to create physical materials to add some variety to your objects.

Hour 11, “Game 2—Chaos Ball”—Time for another game! In this hour, you create Chaos Ball. This title certainly lives up to its name as you implement various collisions, physical materials, and goals. Prepare to mix strategy with twitch reaction.

Hour 12, “Prefabs”—Prefabs are a great way to create repeatable game objects. In Hour 12, you learn to create and modify prefabs. You also learn to build them in scripts.

Hour 13, “2D Game Tools”—In Hour 13, you learn about Unity’s powerful tools for creating 2D games, including how to work with sprites and Box2D physics.

Hour 14, “User Interfaces”—In this hour, you learn how to use Unity’s powerful User Interface system, and how to create a menu for your game.

Hour 15, “Game 3—Captain Blaster”—Game number 3! In this hour, you make Captain Blaster, a retro-style spaceship shooting game.

Hour 16, “Particle Systems”—Time to learn about particle effects. In this chapter, you experiment with Unity’s particle system to create cool effects, and apply them to your projects.

Hour 17, “Animations”—In Hour 17, you get to learn about animations and Unity’s animation system. You experiment 2D and 3D animation, and some powerful animation tools.
Preface

- Hour 18, “Animators”—Hour 18 is all about Unity’s Mecanim animation system. You learn how to use the powerful state machine, and how to blend animations.
- Hour 19, “Game 4—Gauntlet Runner”—Lucky game number 4 is called Gauntlet Runner. This game explores a new way to scroll backgrounds and how to implement animator controllers to build complex blended animations.
- Hour 20, “Audio”—Hour 20 has you adding important ambient effects via audio. You learn about 2D and 3D audio and their different properties.
- Hour 21, “Mobile Development”—In this hour, you learn how to build games for mobile devices. You also learn to utilize a mobile device’s built-in accelerometer and multitouch display.
- Hour 22, “Game Revisions”—It’s time to go back and revisit the four games you have made. This time you modify them to work on a mobile device. You get to see which control schemes translate well to mobile and which don’t.
- Hour 23, “Polish and Deploy”—Time to learn how to add multiple scenes and persist data between scenes. You also learn about the deployment settings and playing your games.
- Hour 24, “Wrap Up”—Here, you look back and summarize the journey you went on to learn Unity. This hour provides useful information about what you have done and where to go next.

Thank you for reading my preface! We hope you enjoy this book and learn much from it. Good luck on your journey with the Unity game engine!

**Companion Files**

Bonus files include full source code listings from every chapter with author comments, all third party art assets (textures, fonts, models), and all third party sound assets.

To gain access to the companion files:

1. Register your product at informit.com/register.
2. Log in or create an account.
3. Enter the product ISBN: 9780672337512, click submit and answer any challenge questions.

Once the process is complete, you can find any available bonus content under “Registered Products.”
About the Authors

**Ben Tristem** is an internet entrepreneur, focusing on teaching technical subjects to beginners. Ben has been passionate about using computers since the days of the ZX81, and is now a world-class technology trainer. At the time of writing, Ben has over 60,000 students and more than 1,200 5-star reviews on his online courses. In previous lives, Ben has been an RAF pilot, financial trader, stunt man, helicopter pilot, franchise creator, and more. Now that he has two kids, Toby and Lucy, he has settled down to focus on what he loves—teaching.

**Mike Geig** is both an experienced teacher and game developer, with a foot firmly in both camps. Mike is a Trainer for Unity Technologies where he develops and delivers recorded, live, and onsite learning content. He enjoys loitering and accordions. His Pearson video series, Game Development Essentials with Unity 4 LiveLessons, is a key title on Unity and rumor has it that people really enjoyed the first edition of *Sams Teach Yourself Unity Game Development in 24 Hours*. Mike was once set on fire and has over a million “likes” on Facebook.
Dedication

From Ben:

To Lizzie: For being an amazing wife, enabling me to thrive.

From Mike:

To Dad: Everything worth learning, I learned from you.

Acknowledgments

From Ben:

I’ve had so much support in writing this book, thank you.

Firstly to Mike for writing the first edition of the book. Having this to work from was an amazing starting point for this second edition. You have been fantastic to work with, and I’m grateful for your time.

Thanks to Laura, our editor, for making it easy for me to write my first book. Thank you also for keeping us all on track so that it got written on time.

Thanks to my beautiful wife, Lizzie, and to my kids, Lucy and Toby, for your patience as I worked late to get the book finished. I’m very grateful for your understanding.

Last but not least to my Mum, without her I probably wouldn’t be writing this!

From Mike:

A big “thank you” goes out to everyone who helped me write this book.

First and foremost, thank you Kara for keeping me on track. I don’t know what we’ll be talking about when this book comes out, but whatever it is, you are probably right. Love ya babe.

Link and Luke: We should take it easy on mommy for a little while. I think she’s about to crack.
Thanks to my parents. As I am now a parent myself, I recognize how hard it was for you not to strangle or stab me. Thanks for not strangling or stabbing me.

Thanks to Angelina Jolie. Due to your role in the spectacular movie Hackers (1995), I decided to learn how to use a computer. You underestimate the impact you had on 10-year-olds at the time. You’re elite!

To the inventor of beef jerky: History may have forgotten your name, but definitely not your product. I love that stuff. Thanks!

Thank you to our technical editors: Tim and Jeff. Your corrections and insights played a vital role in making this a better product.

Thank you Laura for convincing me to write this book. Also thank you for buying me lunch at GDC. I feel that lunch, the best of all three meals, specifically enabled me to finish this.

Finally, a “thank you” is in order for Unity Technologies. If you never made the Unity game engine, this book would be very weird and confusing.
We Want to Hear from You

As the reader of this book, you are our most important critic and commentator. We value your opinion and want to know what we’re doing right, what we could do better, what areas you’d like to see us publish in, and any other words of wisdom you’re willing to pass our way.

You can email or write directly to let us know what you did or didn’t like about this book—as well as what we can do to make our books stronger.

Please note that we cannot help you with technical problems related to the topic of this book, and that due to the high volume of mail we receive, we might not be able to reply to every message.

When you write, please be sure to include this book’s title and author, as well as your name and contact information.

Email: feedback@samspublishing.com
Mail: Sams Publishing
    ATTN: Reader Feedback
    800 East 96th Street
    Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at www.informit.com/register for convenient access to any updates, downloads, or errata that might be available for this book.
This page intentionally left blank
This hour focuses on getting you ready to rock and roll in the Unity environment. We start by looking at the different Unity licenses, choosing one, and then installing it. Once that is installed, you learn how to create new projects as well as open existing ones. You open the powerful Unity editor, and we examine its various components. Finally, you learn to navigate a scene using mouse controls and keyboard commands. This chapter is meant to be hands-on, so download Unity while reading and follow along.

Installing Unity

To begin using Unity, you first need to download and install it. Software installation is a pretty simple and straightforward process these days, and Unity is no exception. Before we can install anything, though, we need to look at the two available Unity licenses: Unity Personal and Unity Professional. Unity Personal is free and more than sufficient to complete all the examples and projects in this book. In fact, Unity Personal contains everything you need to make games commercially, up to an annual revenue of $100,000! If you’re lucky enough to start earning more than this, or you want access to Unity Pro’s advanced features (mainly aimed at teams), then you can always upgrade in the future.

Downloading and Installing Unity

For the purposes of this chapter, we will assume you are sticking with the Unity Personal license. If you went with the Professional version, the process will be very similar, only deviating when it comes to time to choose the license. When you are ready to begin downloading and installing Unity, follow these steps:
1. Download the Unity installer from the Unity download page at http://unity3d.com/get-unity/download.

2. Run the installer and follow the prompts as you would with any other piece of software.

3. When prompted, be sure to leave the Unity 5, Web Player, and Standard Assets check boxes checked (see Figure 1.1). It is OK to install the Example Project if you have space; it won’t affect your experience of the book.

![Figure 1.1](image)

**FIGURE 1.1**
Prompt to choose the installed components.

4. Choose an install location for Unity. It is recommended that you leave the default unless you know what you are doing.

5. Unity 5 will take some time to download, during which time you’ll see a download screen (see Figure 1.2).
Installing Unity

6. If you already have a Unity account, you may be asked to login with it. If you don’t yet have a Unity account, follow the instructions to create one. You will need access to your email to verify your address.

7. That’s it! Unity installation is now complete.

NOTE

Supported Operating Systems and Hardware

To use Unity, you must be using a Windows PC or a Macintosh computer. Although it is possible to build your projects to run on a Linux machine, the Unity editor itself will not. Your computer must also meet the minimum requirements outlined here (taken from the Unity website at the time of writing):

* Windows: XP SP2 or later. Mac OS X: Intel CPU and Snow Leopard 10.8 or later. Note that Unity was not tested on server versions of Windows and OS X.
* Graphics card with DirectX 9 (Shader Model 2.0) capabilities. Any card made since 2004 should work.
* Using occlusion culling requires a GPU with occlusion query support (some Intel GPUs do not support that).

Note that these are minimum requirements.
Getting to Know the Unity Editor

Now that you have Unity installed, you can begin exploring the Unity editor. The Unity editor is the visual component that enables you to build your games in a “what you see is what you get” fashion. Because most interaction we have is actually with the editor, we often just refer to it as Unity. The next portion of this chapter examines all the different elements of the Unity editor and how they fit together to make games.

The Project Dialog

When opening Unity for the first time, you will see the Project dialog (see Figure 1.3). This window is what we use to open recent projects, browse for projects that have already been created, or start new projects.

If you have created a project in Unity already, whenever you open Unity, it will go directly into that project. To get back to the Project dialog, you go (from inside Unity) to File > New Project to get to the Create New Project dialog, or you go to File > Open Project to get to the Open Project dialog.
Creating Our First Project

Let's go ahead and create a project now. You want to pay special attention to where you save the project so that you can find it easily later if necessary. Figure 1.4 shows you what the dialog window should look like before creating the project:

1. Open the New Project dialog.
2. Select a location for your project. We recommend you create a folder called Unity to keep all your book projects together. If you are unsure where to put your project, you can leave the default location.
3. Name your project Chapter 1_Trial. Unity will create a folder with the same name as the project, in the Location specified.
4. Leave 3D selected, and ignore the Asset Packages... button for now.
5. Click Create Project.

![FIGURE 1.4](image)
The settings used for our first project.
CAUTION

Projects and Packages

At first, you might be tempted to select a bunch of “Asset packages” in the Create New Project dialog. We want to caution you against frivolously adding packages to your project, however, because unneeded items can add size and lag. Unused packages just take up space and provide no real benefit. With that in mind, it is better to wait until you actually need a package to import it. Even then, only import the parts of the package that you intend to use.

The Unity Interface

So far, we have installed Unity and looked at the Project dialog. Now it is time to dig in and start playing around. When you open a new Unity project for the first time, you will see a collection of gray windows (called views), and everything will be rather empty (see Figure 1.5). Never fear, we will quickly get this place hopping. In the following sections, we look at each of the unique views one by one. First, though, we want to talk about the layout as a whole.

For starters, Unity allows the user to determine exactly how they want to work. This means that any of the views can be moved, docked, duplicated, or changed. For instance, if you click the word Hierarchy (on the left) to select the Hierarchy view and drag it over to the Inspector (on the right), you can tab the two views together. You can also place your cursor on any line
between views and resize the windows. In fact, why don’t you take a moment to play around
and move things so that they are to your liking. If you end up with a layout that you don’t
much care for, never fear. You can quickly and easily switch back to the built-in default view by
going to Window > Layouts > Default Layout. While we are on the topic of built-in layouts, go
ahead and try out a few of the other layouts (we’re a fan of the Wide layout). If you create a cus-
tom layout you like, you can always save it by going to Window > Layouts > Save Layout. Now
if you accidentally change your layout, you can always get it back.

NOTE
Finding the Right Layout
No two people are alike, and likewise, no two ideal layouts are alike. A good layout will help you work
on your projects and make things much easier for you. Be sure to take the time to fiddle around with
the layout to find the one that works best for you. You will be working a lot with Unity. It pays to set
your environment up in a way that is comfortable.

If you would like to duplicate a view, it is a fairly straightforward process as well. You can sim-
ply right-click any view tab (the tab is the part sticking up with the views name on it), hover
the mouse cursor over Add Tab, and a list of views will pop up for you to choose from (see
Figure 1.6). You may wonder why you would want to duplicate a view. It is possible that in your
view-moving frenzy, you accidentally closed the view. Re-adding the tab will give it back to you. Also, consider the capability to create multiple Scene views. Each Scene view could align with a specific element or axis within your project. If you want to see this in action, check out the four Split built-in layout by going to Window > Layouts > 4 Split. (If you created a layout that you like, be sure to save it first.)

Now, without further ado, let's look at the specific views themselves.

**The Project View**

Everything that has been created for a project (files, scripts, textures, models, and so on) can be found in the Project view (see Figure 1.7). This is the window into which all the assets and organization of our project go. When you create a new project, you will notice a single folder item called Assets. If you go to the folder on your hard drive where you save the project, you will also find an Assets folder. This is because Unity mirrors the Project view with the folders on the hard drive. If you create a file or folder in Unity, the corresponding one appears in the explorer (and vice versa). You can move items in the Project view simply by dragging and dropping. This enables you to place items inside folders or reorganize your project on the fly.

![Diagram of Project View](image)

**Figure 1.7**
The Project view.

**NOTE**

**Assets and Objects**

An *asset* is any item that exists as a file in your assets folder. All textures, meshes, sound files, scripts, and so on are considered assets. In contrast, if you create a game object, but it doesn’t create a corresponding file, it is not an asset.
Getting to Know the Unity Editor

CAUTION

**Moving Assets**

Unity maintains links between the various assets associated with projects. As a result, moving or deleting items outside of Unity could cause potential problems. As a general rule, it is a good idea to do all of your asset management inside Unity.

Whenever you click a folder in the Project view, the contents of the folder will be displayed under the Assets section on the right. As you can see in Figure 1.7, the Assets folder is currently empty, and therefore nothing is appearing on the right. If you would like to create assets, you can do so easily by clicking the Create drop-down menu. This menu enables you to add all manner of assets and folders to your project.

TIP

**Project Organization**

Organization is extremely important for project management. As your projects get bigger, the number of assets will start to grow until finding anything can be a chore. You can help prevent a lot of frustration by employing some simple organization rules:

- Every asset type (scenes, scripts, textures, and so on) should get its own folder.
- Every asset should be in a folder.
- If you are going to use a folder inside another folder, make sure that the structure makes sense. Folders should become more specific and not be vague or generalized.

Following these few, simple rules will really make a difference.

Favorites buttons enable you to quickly select all assets of a certain type. This makes it possible for you to get an “at a glance” view of your assets quickly. When you click one of the Favorites buttons (**All Models**, for instance) or perform a search with the built-in search bar, you will see that you can narrow down the results between Assets and Asset Store. If you click **Asset Store**, you will be able to browse the assets that fit your search criteria from the Unity Asset Store (see Figure 1.8). You can further narrow your results down by free and paid assets. This is a fantastic addition because it enables you to go and grab assets that you need for your project without ever leaving the Unity interface.
The Hierarchy View

In many ways, the Hierarchy view (see Figure 1.9) is a lot like the Project view. The difference is that the Hierarchy view shows all the items in the current scene instead of the entire project. When you first create a project with Unity, you get the default scene, which has just two items in it, the Main Camera and a Directional Light. As you add items to your scene, they will appear in the Hierarchy view. Just like with the Project view, you can use the Create menu to quickly add items to your scene, search using the built-in search bar, and click and drag items to organize and “nest” them.
TIP

Nesting

Nesting is the term for establishing a relationship between two or more items. In the Hierarchy view, clicking and dragging an item onto another item will nest the dragged item under the other. This is commonly known as a parent-child relationship. In this case, the object on top is the parent, and any objects below it are children. You will know when an object is nested because it will become indented. As you will see later, nesting objects in the Hierarchy view can affect how they behave.

TIP

Scenes

A scene is the term Unity uses to describe what you might already know as a level. As you develop a Unity project, each collection of objects and behaviors should be its own scene. Therefore, if you were building a game with a snow level and a jungle level, those would be separate scenes. You will see the words scene and level used interchangeably as you look for answers on the Internet.

TIP

Scene Organization

The first thing you should do when working with a new Unity project is create a Scenes folder under Assets in the Project view. This way, all your scenes (or levels) will be stored in the same place. Be sure to give your scenes a descriptive name. Scene1 may sound like a great name now, but when you have 30 scenes, it can get confusing.

The Inspector View

The Inspector view enables you to see all of the properties of a currently selected item. Simply click any asset or object from the Project or Hierarchy view, and the Inspector view automatically propagates with information.

In Figure 1.10, we can see the Inspector view after the Main Camera object was selected from the Hierarchy view.

Let's break down some of this functionality:

- If you click the check box next to the object's name, it will become disabled and not appear in the project.
- Drop-down lists (such as the Layer or Tag lists; more on those later) are used to select from a set of predefined options.
- Text boxes, drop-downs, and sliders can have their values changed, and the changes will be automatically and immediately reflected in the scene—even if the game is running!
Each game object acts like a container for different components (such as Transform, Camera, and GUILayer in Figure 1.10). You can disable these components by unchecking them or remove them by right-clicking and selecting **Remove Component**.

Components can be added by clicking the **Add Component** button.

**FIGURE 1.10**
The Inspector view.

---

**CAUTION**

**Changing Properties While Running a Scene**
The capability to change the properties of an object and seeing those changes reflected immediately in a running scene is very powerful. It enables you to tweak things like movement speed, jumping height, collision power, and so on, all on the fly without stopping and starting the game. Be wary, though. Any changes you make to the properties of an object while the scene is running will be changed back when the scene finishes. If you make a change and like the result, be sure to remember what it was so that you can set it again when the scene is stopped.
The Scene View

The Scene view is the most important view you work with because it enables you to see your game visually as it is being built (see Figure 1.11). Using the mouse controls and a few hotkeys, you can move around inside your scene and place objects where you want them. This gives you an immense level of control.

In a little bit, we will talk about moving around within a scene, but, first, let’s focus on the controls that are a part of the Scene view:

- **Drawmode:** This controls how the scene is drawn. By default, it is set to Shaded, which means objects will be drawn with their textures in full color.

- **2D/3D view:** This control changes from a 3D view, to a 2D view. Note in 2D view the scene gizmo does not show.

- **Scene lighting:** This control determines whether objects in the Scene view will be lit by default ambient lighting, or only by lights that actually exist within the scene. The default is to include the built-in ambient lighting.

- **Audition mode:** This control sets whether an audio source in the Scene view functions or not.
The Scene Gizmo

The scene gizmo gives you a lot of power over the Scene view. As you can see, the control has an X, Y, and Z indicator that aligns with the three axes. This makes it easy to tell exactly which way you are looking in the scene. We discuss axes and 3D space more in a later chapter. The gizmo also gives you active control over the scene alignment. If you click one of the gizmo’s axes, you will notice that the Scene view immediately snaps to that axis and gets set to a direction like top or left. Clicking the box in the center of the gizmo toggles you between Iso and Persp modes. Iso stands for Isometric and is the 3D view with no perspective applied. Inversely, Persp stands for Perspective and is the 3D view with perspective applied. Try it out for yourself and see how it affects the Scene view. You’ll notice the icon before the word change from parallel lines for isometric and diverging lines like crow’s feet for perspective.

The Game View

The last view to go over is the Game view. Essentially, the Game view allows you to “play” the game inside the editor by giving you a full simulation of the current scene. All elements of a game will function in the Game view just as they would if the project were fully built. Figure 1.12 shows you what a Game view looks like. Note that although the Play, Pause, and Step buttons are not technically a part of the Game view, they control the Game view and therefore are included in the image.
TIP

Missing Game View

If you find that the Game view is hidden behind the Scene view, or that the Game view tab is missing entirely, don’t worry. As soon as you click the Play button, a Game view tab will appear in the editor and begin displaying the game.

The Game view comes with some controls that assist us with testing our games:

- **Play**: The Play button enables you to play your current scene. All controls, animations, sounds, and effects will be present and working. Once a game is running, it will behave just like the game would if it were being run in a standalone player (such as on your PC or mobile device). To stop the game from running, click the Play button again.

- **Pause**: The Pause button pauses the execution of the currently running Game view. The game will maintain its state and continue exactly where it was when paused. Clicking the Pause button again will continue running the game.

- **Step**: The Step button works while the Game view is paused and causes the game to execute a single frame of the game. This effectively allows you to “step” through the game slowly and debug any issues you might have. Pressing the Step button while the game is running will cause the game to pause.
HOUR 1: Introduction to Unity

- **Aspect drop-down:** From this drop-down menu, you can choose the aspect ratio you want the Game view window to display in while running. The default is Free Aspect, but you can change this to match the aspect ratio of the target platform you are developing for.

- **Maximize on Play:** This button determines whether the Game view takes up the entirety of the editor when run. By default, this is off, and a running game will only take up the size of the Game view tab.

- **Mute Audio:** This button turns off the sounds when playing the game. Handy when the person sitting next to you is getting tired of hearing your repeated play-testing!

- **Stats:** This button determines whether rendering statistics are displayed on the screen while the game is running. These statistics can be useful for measuring the efficiency of your scene. This button is set to off by default.

- **Gizmos:** This is both a button and a drop-down menu. The button determines whether gizmos are displayed while the game is running. The button is set to off by default. The drop-down menu (the small arrow) on this button determines which gizmos appear if gizmos are turned on.

**NOTE**

**Running, Paused, and Off**

It can be difficult at first to determine what is meant by the terms **running**, **paused**, and **off**. When the game is not executing in the Game view, the game is said to be off. When a game is off, the game controls do not work and the game cannot be played. When the Play button is pressed and the game begins executing, the game is said to be running. Playing, executing, and running all mean the same thing. If the game is running and the Pause button is pressed, the game stops running but still maintains its state. At this point, the game is paused. The difference between a paused game and an off game is that a paused game will resume execution at the point it was paused, while an off game will begin executing at the beginning.

**Honorable Mention: The Toolbar**

Although not a view, the toolbar is an essential part of the Unity editor. Figure 1.13 shows the toolbar components:

- **Transform tools:** These buttons enable you manipulate game objects and are covered in greater detail later. Pay special attention to the button that resembles a hand. This is the Hand tool and is described later in this chapter.

- **Transform gizmo toggles:** These toggles manipulate how gizmos appear in the Scene view. Leave these alone for now.
Navigating the Unity Scene View

The Scene view gives you a lot of control over the construction of your game. The ability to place and modify items visually is very powerful. None of this is very useful though if you cannot move around inside the scene. This section covers a couple of different ways to change your position and navigate the Scene view.

The Hand Tool

The Hand tool (hotkey: Q) provides you a simple mechanic to move about the Scene view with the mouse (see Figure 1.14). This tool proves especially useful if you are using a mouse with only a single button (because other methods require a two-button mouse). Table 1.1 briefly explains each of the Hand tool controls.

**Table 1.1** The Hand Tool Controls

<table>
<thead>
<tr>
<th>Action</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Click-drag</td>
<td>Drags the camera around the scene</td>
</tr>
<tr>
<td>Hold Alt and click-drag</td>
<td>Orbits the camera around the current pivot point</td>
</tr>
<tr>
<td>Hold Ctrl (Command on Mac) and right-click-drag</td>
<td>Zooms the camera</td>
</tr>
</tbody>
</table>

**Figure 1.14**
The Hand tool.
CAUTION

Different Cameras

When working in Unity, you will be dealing with two types of cameras. The first is the standard game object camera. You can see that you already have one in your scene (by default). The second type is more of an imaginary camera. It is not a camera in the traditional sense. Instead, it is what determines what we can see in the Scene view. In this chapter, when the camera is mentioned, it is the second type that is being referred to. You will not actually be manipulating the game object camera.

Flythrough Mode

Flythrough mode enables you to move about the scene using a tradition first-person control scheme. This mode will feel right at home for anyone who plays first-person 3D games (such as the first-person shooter genre). If you don’t play those games, this mode might take a little getting used to. Once you become familiar with it, though, it will be second nature.

Holding down the right mouse button will put you into Flythrough mode. All the actions laid out for you in Table 1.2 require that the right mouse button be held down.

TABLE 1.2  Flythrough Mode Controls

<table>
<thead>
<tr>
<th>Action</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move the mouse</td>
<td>Causes the camera to pivot, which gives the feeling of “looking around” within the scene.</td>
</tr>
<tr>
<td>Press the WASD keys</td>
<td>The WASD keys move you about the scene. Each key corresponds with a direction: forward, left, back, and right, respectively.</td>
</tr>
<tr>
<td>Press the QE keys</td>
<td>The QE keys move you up and down, respectively, within the scene.</td>
</tr>
<tr>
<td>Hold Shift while pressing WASD or QE keys</td>
<td>Has the same effect as before, but it is much faster. Consider Shift to be your “sprint” button.</td>
</tr>
</tbody>
</table>

TIP

Zoom

Regardless of what method you are using for navigation, scrolling the mouse wheel will always zoom the view within a scene. By default, the scene zooms in and out of the center of the Scene view. If you hold Alt while scrolling, however, you zoom in and out of wherever the mouse is currently pointing. Go ahead and give it a try!
Snap Controls

You have many ways to attain precious control over the scene navigation. Sometimes, you just want to quickly get around the scene though. For times like these, it is good to use what we call **snap controls**. If you want to quickly navigate to, and zoom in on, a game object in your scene, you can do so by highlighting the object in the Hierarchy view and pressing F. You will notice that the scene “snaps” to that game object. Another snap control is one you have seen already. The scene gizmo allows you to quickly snap the camera to any axis. This way, you can see an object from any angle without having to manually move the scene camera around. Be sure to learn the snap controls and navigating your scene quickly with snap!

**Summary**

In this hour, you took our first look at the Unity game engine. You started off by downloading and installing Unity. From there, you learned how to open and create projects. Then you learned about all the different views that make up the Unity editor. You also learned how to navigate around the Scene view.

**Q&A**

**Q.** Are assets and game objects the same?
**A.** Not exactly. Basically the big difference is that assets have a corresponding file or group of files on the hard drive, whereas a game object does not. An asset may or may not contain a game object.

**Q.** There are a lot of different controls and options. Will I need to memorize them all right away?
**A.** Not at all. Most controls and options will already be set to a default state that covers most situations. As your knowledge of Unity grows, you can continue to learn more about the different controls that you have available to you. This chapter is just meant to show you what’s there and to give you some level of familiarity.

**Workshop**

Take some time to work through the questions here to ensure that you have a firm grasp of the material.

**Quiz**

1. True or False: You must purchase Unity Professional to make commercial games.
2. Which view enables us to manipulate objects in a scene visually?
3. True or False: You should always move your asset files around within Unity and not use the operating system’s file explorer.

4. True or False: When creating a new project, you should include every asset that you think is awesome.

5. What mode do you enter in the Scene view when you hold down the right mouse button?

Answers

1. False. Up to $100,000 of revenue you can use the free Personal addition.

2. The Scene view

3. True. This helps Unity keep track of the assets.

4. False. This will take up space, and slow your project down.

5. Flythrough mode

Exercise

Take a moment and practice the concepts studied in this chapter. It is important to have a strong foundational understanding of the Unity editor because everything you will learn from here on out will utilize it in some way. To complete this exercise, do the following:

1. Create a new scene by going to File > New Scene or by pressing Ctrl+N (Command+N on a Mac).

2. Create a Scene folder under Assets in the Project view.

3. Save your scene by going to File > Save Scene or by pressing Ctrl+S (Command+S on a Mac). Be sure to save the scene in the Scenes folder you created and name it something descriptive.

4. Add a cube to your scene. To do this, click the GameObject menu at the top, place your mouse over Create Other, and select Cube from the pop-up menu.

5. Select the newly added cube in the Hierarchy view and experiment with its properties in the Inspector view.

6. Practice navigating around the Scene view using Flythrough mode, the Hand tool, and snap controls. Use the cube as a point of reference to help you navigate.
A

accelerometers, 357-361
Add Grass Texture dialog, 67
Add Key frame (Animation window), 282
Add Terrain Texture dialog, 59
Add Tree dialog, 65
Albedo property (shader), 44
Alt Negative Button/Alt Positive Button property (axis), 147
Amazing Racer, 103-115, 395
adding scripts, 111-113
creating world, 106-108
design, 103-106
concept, 104
rules, 104-105
game control objects, adding, 109-111
playtesting, 114-115
requirements, 105-106
revisions, 368-371
disappearing joystick fixing, 370-371
tilt control, 368-369
using touch joystick, 369-370
scripts, connecting, 113-114
anchor button, 222, 223
anchors, canvas, 220-223
Angular Drag property (rigidbody), 162
Animation Clip Dropdown (Animation window), 282
Animation property (Texture module), 269

animations, 275-288
assets, 277-279
creating, 279-281, 283-284
Curves editor, 287-288
3D artists, 277
idle, 298-300, 308-309
models, preparing for, 276-277
preparation, 298
Record Mode, 285-287
rigs, 275-276
spritesheet slicing, 278-279
timing, 285
tools, 281-288
2D, 277-279
types, 277-281
walk, 300-302
walk turn, 302-305
window, 281-283

animators, 291-315
animation preparation, 298
assets, configuring, 296-305
blend trees, 310-312
creating, 305-314
idle animation, 298-300, 308-309
parameters, 309-310
rigging models, 293-296
rig preparation, 296-298
scripting, 314-315
states, 310-312
transitions, 312-314
walk animation, 300-302
walk turn animation, 302-305
Animator view, 307-308
apps, Unity Remote, 355
area lights, 86
arenas, Chaos Ball, 175-179
bouncy material, 178
texturing, 177-178
arithmetic operators, 130-131
Aspect drop-down menu (Game view), 16
asset packages, 6
assets, 8
animations, 277-279
configuring, animators, 296-305
importing, 78
terrain, 64
importing, 57-58
Asset Store, models, 39-40
assignment operators, 131
attaching scripts, 123-124
audio, 339-349
audio listener, 339-340
changing clips, 349
priorities, 342
scripting, 346-349
sources, 341-346
starting and stopping, 347-348
testing, 343, 344
Scene view, 343, 344
3D, 341, 345-346
2D, 341, 346
Audio Clip property (audio source), 341
audio clips, importing, 342
audio listeners, 77, 339-340
Audition mode (Scene view), 13
axis
properties, 147-148
rotation, 29
Axis property (axis), 148

B
background, Captain Blaster, 240-241
Background property
(cameras), 91
baking objects, 83
Baking property (point lights), 82
base terrain settings, 69
Best Fit property (text objects, Uls), 226
billboards, 67
blend trees, 310-312
blocks, methods, 143
bool variable, 128
Bounce Combine property (physics material), 166
Bounce Intensity property (point lights), 82
Bounce property (Collision module), 266
Bounciness property (physics material), 166
bouncy material, Chaos Ball arena, 178
Box Collider 2D, 213
breaking prefabs, 197
bugs, halos
building games, 387-391
Build Settings dialog, 387-388
built-in 3D objects, 36-37
built-in methods, 127
built-in objects, 25
bullets, Captain Blaster, 245
scripts, 254-255
Bursts property (Emission module), 262
buttons
changing scenes via, 380-381
Uls, 226-227
properties, 226-227
Bypass Effects property (audio source), 341
Bypass Listener Effects property (audio source), 341
Bypass Reverb Zones property (audio source), 341
calling methods, 145-146
cameras, 18, 81, 91-95, 99
adding skyboxes to, 71
anatomy of, 91-92
Captain Blaster, 239-240
falling through the world, 77
layers, 95-99
lens flares, 73-74
multiple, 92-93
orthographic, 204-205
size of, 205
picture in picture, 93-95
properties, 91-92
screen-space, 231
sorting layer, 204, 209-211
split screens, 93-95
canvas, UI, 218-223
adding, 218
anchor button, 222, 223
anchors, 220-223
components, 223
EventSystem game object, 218
Rect Transform, 218-219, 220
Render Mode, 230-232
screen-space camera, 231
screen-space overlay, 230-231
world space, 231-232
Canvas Scaler, 223
Captain Blaster, 237-255, 395
background, 240-241
bullets, 245
script, 254-255
camera, 239-240
controls, 247-255
design, 237-238
concept, 237-238
requirements, 238
rules, 238
DestroyOnTrigger script, 251
improvements, 255
meteors, 244-245
script, 249-250
spawn, 250-251
players, 242-243
revisions, 374-375
ShipControl script, 252-253
triggers, 246
script, 251
UI, 246-247
world, 238-239
Cast Shadows property (Renderer module), 270
Center property (colliders), 164
Chaos Ball, 173-187, 395
arena, 175-179
bouncy material, 178
texturing, 177-178
chaos balls, 181-182
colored balls, 182-183
control objects, 183-187
design, 173-174
concept, 174
requirements, 174
rules, 174
game controller, 185-187
improving, 187
players, 179-180
revisions, 372-374
character controllers, 75-78
adding, 108
char variable, 128
Circle Collider 2D, 213
class declaration section, scripts, 125-126
classes, contents, 126-127
clear flags property (cameras), 91
Clipping Planes property (cameras), 91
code. See also scripts
commments, 126
scripting, 119
scripts, 109, 120
adding, 111-113
attaching, 123-124
basic, 125
class declaration section, 125-126
classes contents, 126-127
conditionals, 133-135
connecting, 113-114
creating, 120-123
Game Control Script, 186
GoalScript.cs, 184-185
iteration, 136-137
methods, 141-146
operators, 130-133
player input, 146-151
using section, 125
variables, 128-130
VelocityScript.cs, 182
code listings
Default Script Code, 125
Demonstration of Class and Local Block Level, 129
Game Control Script, 186
GoalScript.cs, 184-185
VelocityScript.cs, 182
Collaborative groups, 396
colliders, 163-165
collision, 161-171
colliders, 163-165
physics materials, 165-166
trigger, 167-169
raycasting, 169-171
rigidbodies, 161-163
Collision Detection property (rigidbody), 162
Collision module (particle system), 266-268
Collision Quality property (World mode), 267
Color by Speed module (particle system), 264-265
Color over Lifetime module (particle system), 264
Color property (images, UIs), 224
Color property (Color by Speed module), 265
Color property (point lights), 82
comments, 126
concept
Amazing Racer, 104
Captain Blaster, 237-238
Chaos Ball, 174
Gauntlet Runner, 317
Conditionals, 133-135
Constraint property (rigidbody), 162
Controllers
character, 75-78
game, 185-187
control objects, Chaos Ball, 183-187
controls
Captain Blaster, 247-255
Gauntlet Runner, 329-335
script, 330-332
particle systems, 259
virtual, 365-366
Cookie property (point lights), 82
cookies, 89-90
Coordinate systems, 23-24
world versus local coordinates, 24-25
Create New Project dialog, 6,
175-176
cross-platform input, 365-368
projects to mobile conversion, 366-368
virtual controls, 365-366
cross-platform settings, players, 384-385
Culling Mask property (cameras), 91
Culling Mask property (point lights), 83, 98
Current Frame (Animation window), 282
curve editor, particle systems, 270-272
curves editor, animations, 287-288
C# variable types, 128
Cycles property (Texture module), 269
D
Dampen property (Collision module), 266
Dampen property (Limit Velocity over Lifetime module), 263
data
persisting, 381-384
saving, 382-384
Dead property (axis), 148
default particle system, 261-262
Default Script Code listing, 125
deltaPosition property (touch), 359
deltaTime property (touch), 359
Demonstration of Class and Local Block Level listing, 129
Depth property (cameras), 92
Descriptive Name/Descriptive Negative Name property (axis), 147
design
accelerometers, 357-358
Amazing Racer, 103-106
concept, 104
requirements, 105-106
rules, 104-105
Captain Blaster, 237-238
concept, 237-238
requirements, 238
rules, 238
Chaos Ball, 173-174
concept, 174
requirements, 174
rules, 174
Gauntlet Runner, 317-318
concept, 317
requirements, 318
rules, 318
UI, 217
DestroyOnTrigger script, Captain Blaster, 251
detail object settings, 70
dialogs
Add Grass Texture, 67
Add Terrain Texture, 59
Add Tree, 65
Build Settings, 387-388
Create New Project, 6, 175-176
Game Settings, 388-391
Importing Packages, 57-58
Project, 4-5
dimensions, 21-22
coordinate systems, 23-24
world versus local coordinates, 24-25
directional lights, 85-86
cookies, 89
disappearing grass, 68
documentation, 396
Doppler Level property (3D audio), 346
double variable, 128-130
downloading
models, 39-40
Unity, 1-3
Drag property (rigidbody), 162
Draw Halo property (point lights), 82
Draw mode (Scene view), 13
draw order, 2D games, 209-212
in layer, 212
sorting layers, 209-211
dual touch, 371
Duration property (particle system), 261
Dynamic Friction property (physics material), 166
Dynamic Friction 2 property (physics material), 166

E
Edge Collider 2D, 213
Edit Collider property (colliders), 164
editor, 4-17
Console window, 126-127
Game view, 14-16
Hierarchy view, 10-11
Inspector view, 11-12
Project view, 8-10
Scene view, 13-14
toolbars, 16-17
effects
environment, 71-75
fog, 73
lens flares, 73-74
skyboxes, 71-73
water, 74-75
particles, 259
Emission module (particle system), 262
emissive materials, 86
emitter versus particle settings, 268
environments, 63, 78. See also
terrain; worlds
adding, 107
billboards, 67
character controllers, 75-78
effects, 71-75
fog, 73
lens flares, 73-74
skyboxes, 71-73
water, 74-75
game control
Captain Blaster, 248-249
game controller
Chaos Ball, 185-187
game control objects, adding, 109-111

F
factories, methods, 143
Field of View property (cameras), 91
fingertip property (cameras), 360
first project, creating, 5
Flare property (point lights), 83
flares, lens, 73-74
float variable, 128
Flythrough mode (Scene view), 18-19
fog, 73
Force over Lifetime module (particle system), 264
for loop, 137
formats, heightmaps, 54
Friction Combine property (physics material), 166
Friction Direction 2 property (physics material), 166

G
game control
Captain Blaster, 248-249
game controller
Chaos Ball, 185-187
game control objects, adding, 109-111
Game Control script, Gauntlet Runner, 330-332
Game Control Script listing, 186
game overlay (Scene view), 14
games
adding terrain to, 49-51
Amazing Racer, 103-115, 395
adding scripts, 111-113
connecting scripts together, 113-114
creating world, 106-108
design, 103-106
game control objects, adding, 109-111
playtesting, 114-115
attaching scripts, 123-124
building, 387-391
Captain Blaster, 237-255, 395
background, 240-241
bullets, 245
camera, 239-240
controls, 247-255
design, 237-238
improvements, 255
meteors, 244-245
players, 242-243
revisions, 374-375
triggers, 246
UI, 246-247
world, 238-239
Chaos Ball, 173-187, 395
arena, 175-179
chaos balls, 181-182
colored balls, 182-183
control objects, 183-187
design, 173-174
game controller, 185-187
improving, 187
players, 179-180
revisions, 372-374
creating, 396
object, 5
Gauntlet Runner, 317-336, 395
controls, 329-335
design, 317-318
entities, 321-329
improvement, 335
move script, 333
obstacles, 322
player, 323-329
power ups, 321-322
revisions, 375-376
spawn script, 333-334
trigger zone, 322
world, 318-320
organization, 9
revisions, 365-376
Amazing Racer, 368-371
Captain Blaster, 374-375
Chaos Ball, 372-374
cross-platform input,
365-368
Gauntlet Runner, 375-376
2D, 201-214
adding sprites, 205-209
basics, 201-204
challenges, 202
colliders, 212-214
design principles, 201
draw order, 209-212
orthographic cameras,
204-205
grid, 212
scene view, 202-204
setting, 201-202
writing about, 396
Game Settings dialog, 388-391
Game view, 14-16
Gauntlet Runner, 317-336, 395
controls, 329-335
script, 330-332
design, 317-318
concept, 317
requirements, 318
rules, 318
entities, 321-329
Game Control script, 330-332
improvement, 335
player, 323-329
script, 332-333
power ups, 321-322
revisions, 375-376
spawn script, 333-334
trigger zone, 322
script, 329
world, 318-320
generating terrain, 49-56
Geometric properties (colliders), 164
GetComponent, using, 151-152
gizmo (Scene), 14
Gizmos button (Game view), 16
Gizmo selector (Scene view), 14
GoalScript.cs listing, 184-185
Graphical Raycaster, 223
glass
disappearing, 68
painting, 66-68
realistic, 68
Gravity Modifier property (particle system), 262
Gravity property (axis), 148
Grass, 54
sculpting, 51-54
Hierarchical view, 10-11
prefab instances, 190, 191
Horizontal and vertical Overflow property (text objects, UIs), 226
Halo, 87-89
Hand tool, 17-18
HDR property (cameras), 92
heightmaps
formats, 54
sculpting, 51-54
Hierarchy view, 10-11
prefab instances, 190, 191
Horizontal and vertical Overflow property (text objects, UIs), 226
idle animation, 298-300, 308-309
if/else if statement, 134-135
if/else statement, 134
if statement, 133-134, 135
images, Uls, 224-225
properties, 224
importing
assets, 78
audio clips, 342
models, 37-39
sprites, 206
mode, 206-208
sizes, 208-209
terrain assets, 57-58
Importing Packages dialog, 57-58
inheritance, prefabs, 190, 196-197
Inherit Velocity property (particle system), 262
input
basics, 147-148
cross-platform, 365-368
projects to mobile conversion, 366-368
virtual controls, 365-366
key, 147, 149-150
mouse, 150-151
multi-touch, mobile devices, 359-361
scripting, 146-151
Input Manager, 147, 149
Inspector view, 11-12
curve editor, 271
rig preparation, 296
script preview, 120, 121
Installing unity, 1-4
instances, prefabs, 190
adding to scenes, 195
Instantiate() method, 197
instantiating prefabs through code, 197
Intensity property (point lights), 82
Interactable property (buttons, Uls), 226
Interpolate property (rigidbody), 162
int variable, 128
Invert property (axis), 148
invisible items, scenes, 97
Is Kinematic property (rigidbody), 162
Is Trigger property (colliders), 164
iteration, 136-137
for loop, 137
while loop, 136
Joy Num property (axis), 148
Key frames (Animation window), 282
key input, 149-150
layers, 95-99
order in, 212
overloading, 95
sorting, 204, 209-211
Layers drop-down menu, 17, 96
Layout drop-down menu, 17
lens flares, 73-74
license, Unity, activating, 1-3
Lifetime Loss property (Collision module), 266
lighting scenes, 13
lights, 81-90, 99
area, 86
baking objects, 82
cookies, 89-90
creating out of objects, 86
directional, 85-86
halos, 87-89
layers, 95-99
point, 82-84
properties, 82-83
repeat properties, 81
spotlights, 84-85
Limit Velocity over Lifetime module (particle system), 263
listings
Default Script Code, 125
Demonstration of Class and Local Block Level, 129
Game Control Script, 186
GoalScript.cs, 184-185
VelocityScript.cs, 182
LoadLevel() method, 381, 391
local components, accessing, 151-153
local coordinates, versus world coordinates, 24-25
logical operators, 132-133
Looping property (particle system), 261
Loop property (audio source), 342
loops
for, 137
while, 136
managing scenes, 379-381
maps, heightmaps
formats, 54
sculpting, 51-54
Mass property (rigidbody), 162
Material property (images, Uls), 224
Material property (colliders), 164
Material property (Renderer module), 270
materials, 41, 43
  emissive, 86
  models, applying to, 45-46
  UIs, 225
Max Distance property (3D audio), 346
Maximize on Play button (Game view), 16
Max Particles (particle system), 262
Max Particle Size property (Renderer module), 270
Mesh colliders, 165
metas
  versus models, 36
  scaling of, 39
  simple modeling, 37
Metallic property (shader), 44
meteors, Captain Blaster, 244-245
  script, 249-250
  spawn, 250-251
methods, 141-146
  blocks, 143
  built-in, 127
  calling, 145-146
  as factories, 143
  identifying parts, 143
  Instantiate(), 197
  LoadLevel(), 381, 391
  names, 142
  OnGUI(), 361
  parameter list, 142-143
  return type, 142
  writing, 144-145
Min Distance property (3D audio), 346
minimum requirements, Unity, 3
Min Kill Speed property (Collision module), 266
mobile development, 353-361
  accelerometers, 357-361
  environments, setting up, 354-355
  preparing for, 353-356
  Unity Remote app, 355
mobile devices
  multi-touch input, 359-361
  multitudes of, 354
testing, 356
models, 35-40, 45
  applying textures, shaders, and materials, 45-46
Asset Store, 39-40
  built-in 3D objects, 36-37
downloading, 39-40
importing, 37-39
  versus meshes, 36
model asset workflow, 41
  preparing for animation, 276-277
rigging, animators, 293-296
modules, particle systems, 259-270
Collision, 266-268
Color by Speed, 264-265
Color over Lifetime, 264
default, 261-262
Emission, 262
External Forces, 265
Force over Lifetime, 264
Limit Velocity over Lifetime, 263
Renderer, 269-270
Rotation by Speed, 265
Rotation over Lifetime, 265
shape, 263
Size by Speed, 265
Size over Lifetime, 265
Sub Emitter, 269
Texture Sheet, 269
Velocity over Lifetime, 263
MonoDevelop, 120, 122, 123
mouse input, 150-151
move script, Gauntlet Runner, 333
multiple cameras, 92-93
multiple skyboxes, cameras, 71
multi-touch input, mobile devices, 359-361
mute audio button, 344
Mute property (audio source), 341

N
Name property (axis), 147
names, methods, 142
Navigation property (buttons, UIs), 227
Negative Button/Positive Button property (axis), 147
nested objects, transformations, 32-33
nesting, 11
Normal Direction property (Renderer module), 270
Normalized View Port Rect property (cameras), 91
Normal Map property (shader), 44

O
objects, 21, 25. See also specific objects
  baking, 83
  built-in, 25
  built-in 3D objects, 36-37
  consolidating, 176
  control, 183-187
  creating lights out of, 86
detail settings, 70
dimensions, 21-22
EventSystem, 218
finding, 153-156
game control, adding, 109-111, 334-335
keeping, 381-382
layers, 95-99
overloading, 95
preparing for animation, 276-277
rigging, animators, 293-296
sub systems, 36-37
perspective, 381-384
picture in picture, 94-95
prefabs, 189
breaking, 197
creating, 193-194
inheritance, 190, 196-197
instances, 190
instantiating through code, 197
structure, 190-192
updating, 196-197
rotation, 29-30
scaling, 30-31
spin, 283-284
target, transforming, 157
textures, 41
transformations, 26-33
hazards, 31-32
nested objects, 32-33
transforming, 153
translation, 27-29
obstacles, Gauntlet Runner, 322
Occlusion Culling property (cameras), 92
Offset property (shader), 44
On Click () property (UIs), 227-229
OnGUI() method, 361
operating systems, supported, 3
operators, 130-133
arithmetic, 130-131
assignment, 131
equality, 131-132
logical, 132-133
order, in layer, 212
order, scenes, establishing, 380-381
organization, projects, 9
orthographic cameras, 204-205
size of, 205
Output property (audio source), 341

P
painting
grass, 66-68
textures, terrain, 60
trees, 63-66
parameter list, methods, 142-143
parameters, animators, 309-310
Particle Radius property (Plane mode), 266
particles, 257
effects, 259
making collide, 267-268
particle settings versus emitter, 268
particle systems, 257-272
tools, 259
creating, 258
curve editor, 270-272
modules, 259-270
Collision, 266-268
Color by Speed, 264-265
Color over Lifetime, 264
default, 261-262
Emission, 262
External Forces, 265
Force over Lifetime, 264
Limit Velocity over Lifetime, 263
Renderer, 269-270
Rotation by Speed, 265
Rotation over Lifetime, 265
shape, 263
Size by Speed, 265
Size over Lifetime, 265
Sub Emitter, 269
Texture Sheet, 269
Velocity over Lifetime, 263
particles, 257
making collide, 267-268
Unity, 257-258
Pause button (Game view), 15
per-platform settings, players, 385-387
persisting objects, 382
phase property (touch), 360
physics
collision, 161-171
colliders, 163-165
physics materials, 165-166
raycasting, 169-171
rigidbodies, 161-163
trigger, 167-169
2D, 212-214
colliders, 212-214
rigidbody, 212
physics materials, 165-166
picture in picture, 93-95
Pitch property (audio source), 342
Place Tree tool, 64-65
Planes property (Plane mode), 266
Planes/World property (Collision module), 266
Play button (Game view), 15
player input
basics, 147-148
key, 147, 149-150
mouse, 150-151
scripting, 146-151
PlayerPrefs file, saving data to, 382-383
players
Captain Blaster, 242-243
Chaos Ball, 179-180
Gauntlet Runner, 323-329
script, 332-333
settings, 384-387
Play On Awake property (audio source), 342
Play On Wake property (particle system), 257-272
Play button (Game view), 15
player input
basics, 147-148
key, 147, 149-150
mouse, 150-151
scripting, 146-151
PlayerPrefs file, saving data to, 382-383
players
Captain Blaster, 242-243
Chaos Ball, 179-180
Gauntlet Runner, 323-329
script, 332-333
settings, 384-387
Play On Awake property (audio source), 342
Play On Wake property (particle system), 257-272
playtesting Amazing Racer, 114-115
point lights, 82-84
properties, 82-83
scenes, adding to, 83
points, 23
Polygon Collider 2D, 213
position property (touch), 360
power up, Gauntlet Runner, 321-322
script, 333
prefabs, 189-197
breaking, 197
creating, 193-194
inheritance, 190, 196-197
instances, 190
adding to scenes, 195
scaling objects, 30-31
scenes, 11, 379, 395
changing via buttons, 380-381
character controllers, adding to, 76-78
directional lights, adding to, 85-86
establishing order, 380-381
fog, adding to, 73
Gauntlet Runner, 318-319
gizmo, 14
invisible items, 97
lens flares, adding to, 74
lighting, 13
managing, 379-381
point lights, adding to, 83
prefabs instances, adding to, 195
Scene view, 17-19
skyboxes, adding to, 72-73
spotlights, adding to, 85
switching, 381
Scene view, 13-14, 17-19
audio testing, 343, 344
Flythrough mode, 18-19
2D games, 202-204
scope, variables, 129
screen-space camera, 231
screen-space overlay, 230-231
scripting, 119-137, 141-157
accessing local components, 151-153
animators, 314-315
audio, 346-349
conditionals, 133-135
finding objects, 153-156
iteration, 136-137
language, 120
methods, 141-146
modifying object components, 157
operators, 130-133
arithmetic, 130-131
assignment, 131
equality, 131-132
logical, 132-133
player input, 146-151
scripts, 109, 120
adding, 111-113
attaching, 123-124
basic, 125
Captain Blaster
bullets, 254-255
DestroyOnTrigger, 251
moteors, 249-250
meteor spawn, 250-251
ShipControl, 252-253
triggers, 251
class declaration section, 125-126
classes contents, 126-127
connecting, 113-114
creating, 120-123
Game Control Script, 186
Gauntlet Runner
controls, 330-332
move script, 333
players, 322-333
spawn script, 333-334
trigger zone, 329
GoalScript.cs, 184-185
using section, 125
variables, 128-130
VelocityScript.cs, 182
Scrubber (Animation window), 282
sculpting
heightmaps, 51-54
terrain, tools, 54-56
worlds, 106-107
Send Collision Messages property
(Collision module), 266
Sensitivity property (axis), 148
Separate Axis property (Limit Velocity over Lifetime module), 263
Set Native Size property
(images, Uls), 224
settings, players, 384-387
shaders, 41, 42
models, applying to, 45-46
properties, 44
Shadow Type property (point lights), 82
shape module (particle system), 263
Shininess property (shader), 44
ShipControl script, Captain Blaster, 252-253
Simulation Space property
(particle system), 262
Size by Speed module (particle system), 265
Size over Lifetime module
(particle system), 265
Size property (colliders), 164
skyboxes, 71-73
cameras, adding to, 71
scenes, adding to, 72-73
Smoothness property (shader), 44
Snap property (axis), 148
Sorting Fudge property (Renderer module), 270
sorting layers, 204, 209-211
creating, 209-211
Sort Order property (Render module), 270
Source Image property (images, Uls), 224
sources, audio, 341-346
Space property (Velocity over Lifetime module), 263
Spatial Blend property (audio source), 342
spawn script, Gauntlet Runner, 333-334
specific key presses, reading, 150
Speed property (Limit Velocity over Lifetime module), 263
Speed Range property (Color by Speed module), 265
split screens, 93-95
spotlights, 84-85
cookies, 89-90
Spread property (3D audio), 346
Sprite Editor, 206-208
sprite mode, 206-208
sprites, 2D games
adding, 205-209
triggers
trigger zone, Gauntlet Runner, 322
script, 329
2D animations, 277-279
2D audio, 341, 346
2D games, 201-214
adding sprites, 205-209
importing, 206
sizes, 208-209
sprite mode, 206-208
basics, 201-204
challenges, 202
colliders, 212-214
design principles, 201
draw order, 209-212
in layer, 212
sorting layers, 209-211
orthographic cameras, 204-205
rigidbody, 212
scene view, 202-204
setting, 201-202
2D objects, 22
2D scene view, 13
Type property (axis), 148
Type property (point lights), 82

U
UiS. See user interfaces (UiS)
Unity
2D colliders, 213
downloading, 1-3
downloading, 4-17
downloading, 1-4
editor, 6-8
installing, 6-8
interface, 6-8
license, activating, 1-3
minimum requirements, 3
modifying public variables in, 129-130
Unity particle systems, 257-258
Unity Remote app, 355, 371
unwrap texture, 42
updating prefabs, 196-197
Use Gravity property (rigidbody), 162
user input. See player input
user interfaces (UiS), 217-232
canvas, 218-223
adding, 218
anchor button, 222, 223
anchors, 220-223
components, 223
EventSystem game object, 218
Rect Transform, 218-219, 220
Render Mode, 230-232
Captain Blaster, 246-247
controls, 223-230
design, 217
elements, 223-230
buttons, 226-227
On Click () property, 227-229
images, 224-225
sorting, 230
text, 225-226
materials, 225
principles, 217-218
using section, scripts, 125

V
value curve, 261
variables, 128-130
creating, 128
private, 129-130
public, 129-130
scope, 129
Velocity over Lifetime module (particle system), 263
VelocityScript.cs listing, 182
View Port Rect property (cameras), 91
views
duplicating, 7
Game, 14-16
Hierarchy, 10-11, 190, 191
Inspector, 11-12
curve editor, 271
rig preparation, 296
script preview, 120, 121

W
walk animation, 300-302
walk turn animation, 302-305
water, 74-75
while loop, 136
wind, settings, 70
window (Animation), 281-283
world coordinates, versus local coordinates, 24-25
worlds
Amazing Racer, 106-108
Captain Blaster, 238-239
Chaos Ball, arena, 175-176
Gauntlet Runner, 318-320
sculpting, 106-107
world space, 231-232
writing methods, 144-145

X
XYZ property (Velocity over Lifetime module), 263
This page intentionally left blank
This video course is a fantastic compliment to what you have learned in the book. As well as recapping the basics of scripting and using Unity, you will...

- Build seven more brand-new 2D and 3D games.
- Take the course anytime, anywhere, at your own pace.
- Interact direct with Ben in the discussion forums.
- Get help from a thriving community of students.
- Host your games online and share with your friends.

Simply visit https://www.udemy.com/unitycourse to find out more.