

Mike Geig



In **Full Color**

Fourth Edition

Sams **Teach Yourself**

Unity[®] Game Development

in **24**
Hours



FREE SAMPLE CHAPTER

SHARE WITH OTHERS



Mike Geig

Sams **Teach Yourself**

Unity

Game

Development

in **24**
Hours

SAMS

Sams Teach Yourself Unity Game Development in 24 Hours

Copyright © 2022 by Pearson Education, Inc.

All rights reserved. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, request forms, and the appropriate contacts within the Pearson Education Global Rights & Permissions Department, please visit www.pearson.com/permissions/. No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein.

ISBN-13: 978-0-13-744508-0

ISBN-10: 0-13-744508-3

Library of Congress Control Number: 2021943446

ScoutAutomatedPrintCode

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Sams Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Unless otherwise indicated on the page, © Unity Technologies for screenshots.

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an “as is” basis. The author and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

Special Sales

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact intlcs@pearson.com.

Editor-in-Chief

Mark Taub

Signing Editor

Malobika

Chakraborty

Development Editor

Chris Zahn

Managing Editor

Sandra Schroeder

Senior Project Editor

Tonya Simpson

Copy Editor

Kitty Wilson

Indexer

Timothy Wright

Proofreader

Abigail Manheim

Technical Editor

Rocio Chongtay

Editorial Assistant

Cindy Teeters

Cover Designer

Chuti Prasertsith

Compositor

codeMantra

Contents at a Glance

Preface	xi
HOUR 1 Introduction to Unity	1
HOUR 2 Game Objects	25
HOUR 3 Models, Materials, and Textures	41
HOUR 4 Terrain and Environments	57
HOUR 5 Lights and Cameras	81
HOUR 6 Game 1: <i>Amazing Racer</i>	101
HOUR 7 Scripting, Part 1	119
HOUR 8 Scripting, Part 2	141
HOUR 9 Collision	161
HOUR 10 Game 2: <i>Chaos Ball</i>	175
HOUR 11 Prefabs	191
HOUR 12 2D Game Tools	205
HOUR 13 2D Tilemaps	221
HOUR 14 User Interfaces	239
HOUR 15 Game 3: <i>Captain Blaster</i>	259
HOUR 16 Particle Systems	281
HOUR 17 Animations	301
HOUR 18 Animators	317
HOUR 19 Timeline	341
HOUR 20 Game 4: <i>Gauntlet Runner</i>	357
HOUR 21 Audio	377
HOUR 22 Mobile Development	391
HOUR 23 Polish and Deploy	403
HOUR 24 Wrap-up	417
Index	423

Table of Contents

Preface	xi
HOUR 1: Introduction to Unity	1
Installing Unity	1
Getting to Know the Unity Editor	5
Navigating the Unity Scene View	18
Summary	21
Q&A	21
Workshop	22
Exercise	22
HOUR 2: Game Objects	25
Dimensions and Coordinate Systems	25
Game Objects	29
Transforms	30
Summary	38
Q&A	38
Workshop	39
Exercise	39
HOUR 3: Models, Materials, and Textures	41
The Basics of Models	41
Textures, Shaders, and Materials	47
Summary	53
Q&A	53
Workshop	54
Exercise	54
HOUR 4: Terrain and Environments	57
Terrain Generation	57
Terrain Textures	65

Generating Trees and Grass	69
Character Controllers	76
Summary	77
Q&A	78
Workshop	78
Exercise	78
HOUR 5: Lights and Cameras	81
Lights	81
Cameras	90
Layers	94
Summary	99
Q&A	99
Workshop	99
Exercise	100
HOUR 6: Game 1: <i>Amazing Racer</i>	101
Design	101
Creating the Game World	104
Gamification	108
Playtesting	114
Summary	115
Q&A	115
Workshop	116
Exercise	117
HOUR 7: Scripting, Part 1	119
Scripts	120
Variables	128
Operators	130
Conditionals	133
Iteration	137
Summary	138
Q&A	139
Workshop	139
Exercise	140

HOUR 8: Scripting, Part 2	141
Methods	142
Input	147
Accessing Local Components	153
Accessing Other Objects	155
Summary	159
Q&A	159
Workshop	159
Exercise	160
HOUR 9: Collision	161
Rigidbody	161
Enabling Collision	163
Triggers	167
Raycasting	170
Summary	171
Q&A	172
Workshop	172
Exercise	172
HOUR 10: Game 2: <i>Chaos Ball</i>	175
Design	175
The Arena	177
Game Entities	181
The Control Objects	185
Improving the Game	189
Summary	189
Q&A	189
Workshop	190
Exercise	190
HOUR 11: Prefabs	191
Prefab Basics	191
Working with Prefabs	194
Summary	201

Q&A	201
Workshop	202
Exercise	202
HOUR 12: 2D Game Tools	205
The Basics of 2D Games	205
Orthographic Cameras	208
Adding Sprites	209
Draw Order	213
2D Physics	216
Summary	218
Q&A	218
Workshop	218
Exercise	219
HOUR 13: 2D Tilemaps	221
The Basics of Tilemaps	221
Palettes	225
Tiles	227
Tilemaps and Physics	233
Summary	236
Q&A	237
Workshop	237
Exercise	237
HOUR 14: User Interfaces	239
Basic UI Principles	239
The Canvas	240
UI Elements	246
Canvas Render Modes	252
Summary	255
Q&A	255
Workshop	256
Exercise	256

HOUR 15: Game 3: <i>Captain Blaster</i>	259
Design	259
The World	260
Controls	270
Improvements	278
Summary	278
Q&A	279
Workshop	279
Exercise	280
HOUR 16: Particle Systems	281
Particle Systems	281
Particle System Modules	283
The Curves Editor	297
Summary	299
Q&A	299
Workshop	299
Exercise	300
HOUR 17: Animations	301
Animation Basics	301
Animation Types	303
Animation Tools	306
Summary	314
Q&A	315
Workshop	315
Exercise	315
HOUR 18: Animators	317
Animator Basics	317
Configuring Your Assets	321
Creating an Animator	329
Scripting Animators	338
Summary	339
Q&A	339

Workshop	340
Exercise	340
HOUR 19: Timeline	341
Timeline Basics	341
Working with Timelines	344
Going Beyond Simple Control	350
Summary	354
Q&A	354
Workshop	354
Exercise	355
HOUR 20: Game 4: Gauntlet Runner	357
Design	357
The World	359
The Entities	362
The Controls	367
Room for Improvement	375
Summary	375
Q&A	375
Workshop	375
Exercise	376
HOUR 21: Audio	377
Audio Basics	377
Audio Sources	379
Audio Scripting	384
Audio Mixers	386
Summary	388
Q&A	388
Workshop	389
Exercise	389
HOUR 22: Mobile Development	391
Preparing for Mobile	391
Accelerometers	396

Summary	401
Q&A	401
Workshop	401
Exercise	402
HOUR 23: Polish and Deploy	403
Managing Scenes	403
Persisting Data and Objects	406
Unity Player Settings	410
Building Your Game	412
Summary	414
Q&A	414
Workshop	414
Exercise	415
HOUR 24: Wrap-up	417
Accomplishments	417
Where to Go from Here	420
Resources Available to You	421
Summary	421
Q&A	421
Workshop	421
Exercise	422
Index	423

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 (in about 24 hours, to be exact) while covering fundamental principles of game development. Unlike other books that cover only 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.

You can find all the code assets used in this book on GitHub at <http://fixbyproximity.com/Downloads/UnityBook.html>.

About the Author

Mike Geig works on the production team at Unity Technologies, where he helps democratize game development by developing and delivering high-impact learning resources. Mike has experience as an indie game developer, a university educator, and an author. A gamer at heart, Mike works to make the development of interactive entertainment fun and accessible for all skill sets. Hi, Mom!

Acknowledgments

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

This edition comes at a very strange time in my life. Thank you to everyone who helps make it the good kind of strange.

Link and Luke: You continue to be the tornado that carries me to Oz...while also destroying most of Kansas.

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 Rocio Chongtay for being an awesome technical editor for this book. Also, thanks for reaching out and asking me when I am going to update this book. “Now.”

Thank you to Chris Zahn for helping with all of the formatting and structural edits. If the readers find any problems in this book, I will tell them to blame you!

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.

Dedication

To all the people who read the dedications page. Nothing can stop you now.

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.

We welcome your comments. You can email or write 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 better.

Please note that we cannot help you with technical problems related to the topic of this book.

When you write, please be sure to include this book's title and author as well as your name and email address. We will carefully review your comments and share them with the author and editors who worked on the book.

Email: community@informit.com

Reader Services

HOUR 1

Introduction to Unity

What You'll Learn in This Hour:

- ▶ How to install Unity
- ▶ How to create a new project or open an existing project
- ▶ How to use the Unity editor
- ▶ How to navigate inside the Unity Scene view

This hour focuses on getting you ready to rock and roll in the Unity environment. It starts by looking at the different Unity licenses and installing the one you choose. This hour you'll also learn how to create new projects as well as open existing ones. You'll open the powerful Unity editor and examine its various components. Finally, you'll learn to navigate a scene by using mouse controls and keyboard commands. This lesson is meant to be hands-on, so download Unity while reading and follow along.

Installing Unity

Before you can 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 you can install anything, though, you need to look at the three available Unity license options: Unity Personal, Unity Plus, and Unity Pro. Unity Personal is free and has everything you need 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 if you want to access the advanced features of Unity Plus or Unity Pro (mainly aimed at teams), then you can always upgrade in the future.

NOTE**Unity Hub**

From the Unity website you need to download and install the Unity Hub. The Hub is a launcher that acts as a centralized...well...hub for all your Unity editor installations and projects. (The Unity editor is the software that enables you to make games.) If you're coming to this book after having used a previous version of Unity without the Hub, don't worry. The concepts behind Unity installations and projects are exactly the same as they've been in the past. As the saying goes, "new packaging, same great taste!"

NOTE**Unity Updates**

The images and instructions in this book are accurate as of this writing. That being said, the Unity Hub and editor are living and constantly evolving pieces of software. As such, new versions may come out, and the images or text you see here might not match what you see onscreen. Although the fine specifics of the software may change, the general steps and concepts outlined in this book will still be accurate and relevant!

Downloading and Installing Unity Hub

As mentioned earlier in this lesson, when you want to program games using Unity, the Unity Hub is your starting point. When you are ready to begin downloading and installing Unity Hub, follow these steps:

- 1.** Go to the Unity Store website, at <https://store.unity.com>, and choose your license type.
- 2.** If you chose the Unity Personal edition, you have the option to download the Unity Hub directly or to go through a guided installation experience aimed at assisting brand-new users. Either way will get you where you need to be.
- 3.** Run the installer and follow the prompts as you would with any other piece of software.
- 4.** Open up the Unity Hub application (see Figure 1.1). You may be prompted to sign in or create a new account. Doing so takes only a moment, and you will need an account later, so go ahead and do it now.

In addition to being the place where you manage all projects and installations, the Unity Hub is also a place where you can find community news and learning resources to assist with your development journey.

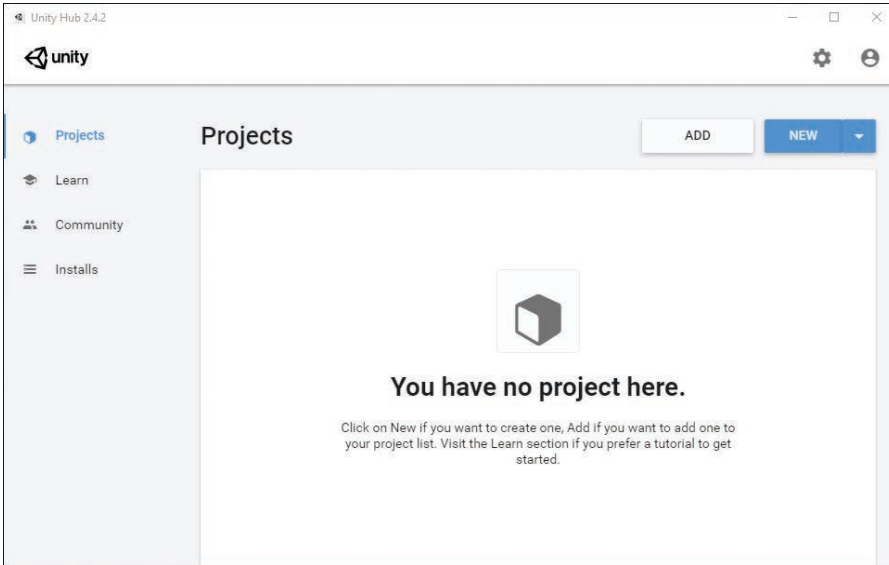


FIGURE 1.1
The Unity Hub.

NOTE

Internet Links

All Internet URLs in this book are current as of the time this book was published. Web locations do change sometimes, though. If the material you are looking for is no longer provided at the links listed, a good Internet search should turn up what you are looking for.

Installing the Unity Editor

Now that the hub is installed, it is time to install the Unity editor. Using the Hub, you can have as many versions of Unity installed as you'd like (and that your hard drive allows). To install the Unity editor, follow these steps:

1. In the Hub, click **Installs** and then click the **ADD** button.
2. Select the 2020 LTS release (see Figure 1.2). (The note "Why 2020 LTS?" explains why you want this version.) Click **Next**.

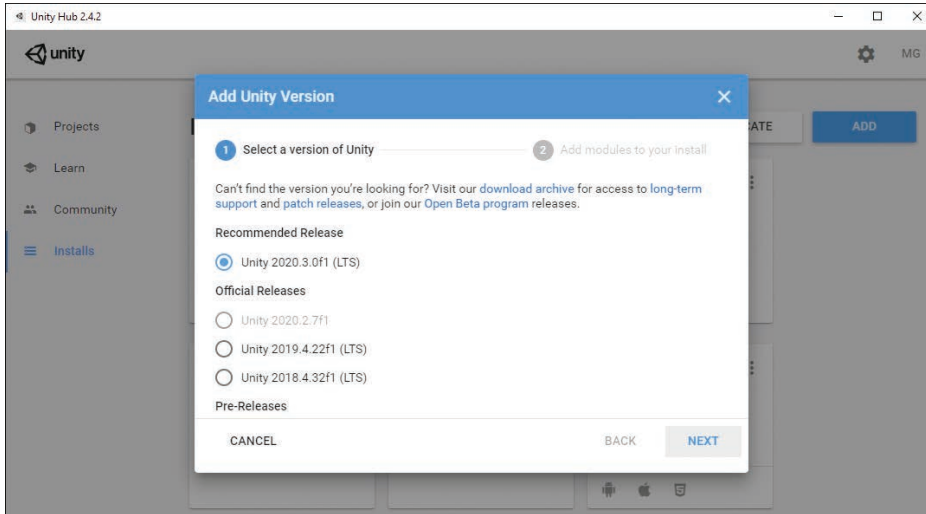


FIGURE 1.2
Selecting the 2020 LTS release of Unity.

3. In the next window, notice that you can select any add-ons or build platforms you'd like to support (see Figure 1.3). You can add platforms now, but you don't have to select any of them in order to continue with this book. You can also come back here later and add new items. So for now, just click **Done**.

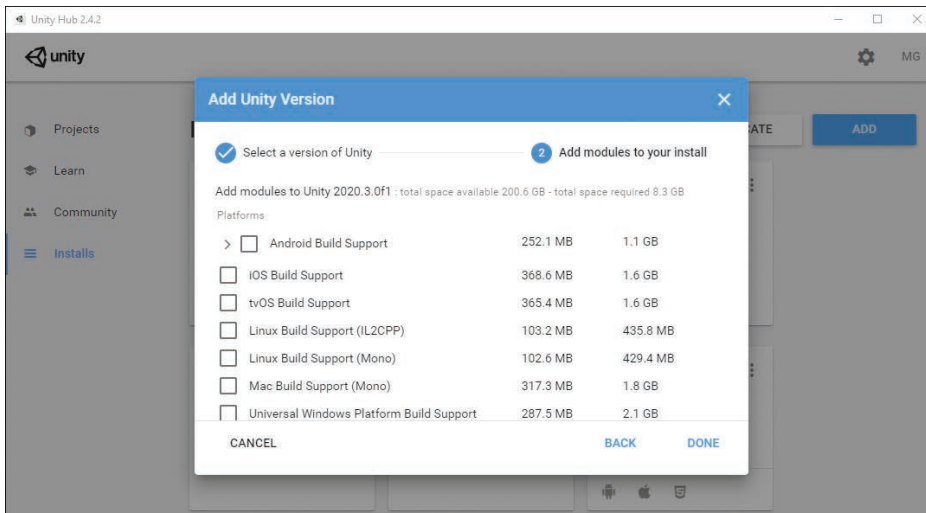


FIGURE 1.3
Additional installation modules.

4. Now you just need to wait for the installation to complete. When it is done, you will be good to go!

NOTE

Why 2020 LTS?

You might be wondering why you installed a 2020 version of the software and what LTS means. With Unity releases, the LTS (Long Term Support) version is the version that combines all previous features, is fully stable, and will be supported for at least the next two years. The LTS version of the engine usually doesn't come out until the beginning of the next year, so Unity 2020 LTS came out at the beginning of 2021 and was the newest version when this book was published.

NOTE

Supported Operating Systems and Hardware

To use Unity, you must be using a Windows PC or a Mac computer. Although there is a version of the editor that runs on Linux machines, Linux is not a directly supported OS. Your computer must also meet the minimum requirements outlined here (taken from the Unity website at the time this book was published):

- ▶ Windows 7 SP1+, Windows 8, or Windows 10, 64-bit versions only; OS X 10.12.6+. Note that Unity was not tested on server versions of Windows and OS X.
- ▶ Graphics card with DX9 (shader model 3.0) or DX11 with feature level 9.3 capabilities.
- ▶ A CPU that supports the SSE2 instruction set (most modern CPUs).

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 games in a “what you see is what you get” fashion. Because most interaction you have is actually with the editor, many people refer to it as simply *Unity*. This section examines the various elements of the Unity editor and how they fit together to make games.

The Projects Section

Not only is the Unity Hub where you manage your editor installations, it is where you create and select projects. At this point, the Projects section of the Hub is probably blank for you (see Figure 1.4).

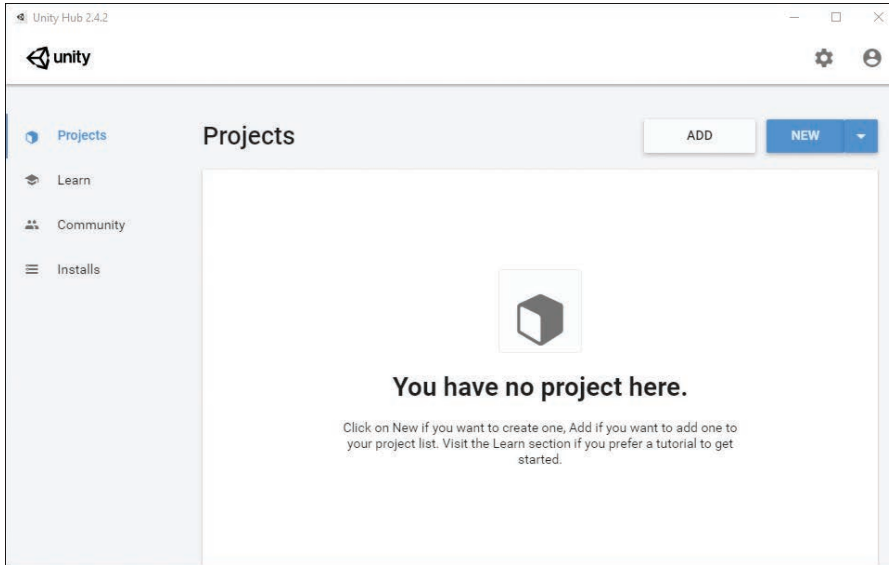


FIGURE 1.4
The empty Projects section of the Hub.

In order to create a new project, all you have to do is click **New**. If you have multiple editor versions installed and you'd like to pick the version, you can do so by clicking the downward arrow next to the New button. Finally, if you want to open a project you've already created (and that isn't in the project list already), you can do so by clicking **ADD**. In the following exercise, you will see exactly how to create a new project.

▼ TRY IT YOURSELF

Creating Your First Project

You are ready to create a project. Pay special attention to where you save the project so that you can easily find it later. Figure 1.5 shows the dialog box you use to create a project. Follow these steps:

1. Open the Unity Hub and click **NEW**. The New Project dialog pops up.
2. Select a location for your project. I recommend that you create a folder called Unity Projects 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 **Hour 1 TIY**. Unity creates a folder with the same name as the project, in the location specified in this dialog.

4. Leave **3D** selected for now. You'll learn about some of the other options in the future.
5. Click **CREATE**.

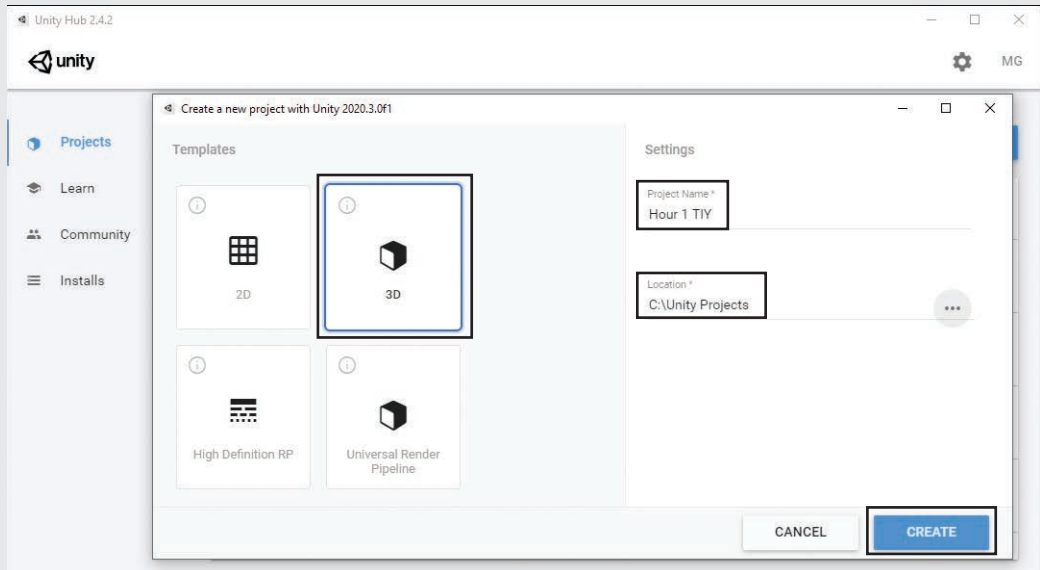


FIGURE 1.5
The settings for your first project.

NOTE

2D, 3D, Rendering Pipelines?

You might be wondering what the other options on the New Project dialog are for. The 2D and 3D buttons allow you to select which type of game you are planning on making. Don't worry about picking the wrong one or being unsure. These options just control editor settings and can be changed at any time. The High Definition RP and Universal Render Pipeline options are a part of Unity's Scriptable Rendering Pipeline, which gives developers a lot of control over exactly how their projects render but requires knowledge that is beyond the scope of this lesson.

The Unity Interface

So far, you have installed Unity and looked at the New 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 see a collection of gray windows (called *views*), and everything is rather empty (see Figure 1.6). Never fear; you will quickly get this place hopping. The following sections look at each of the unique views, one by one. First, though, let's talk about the layout.

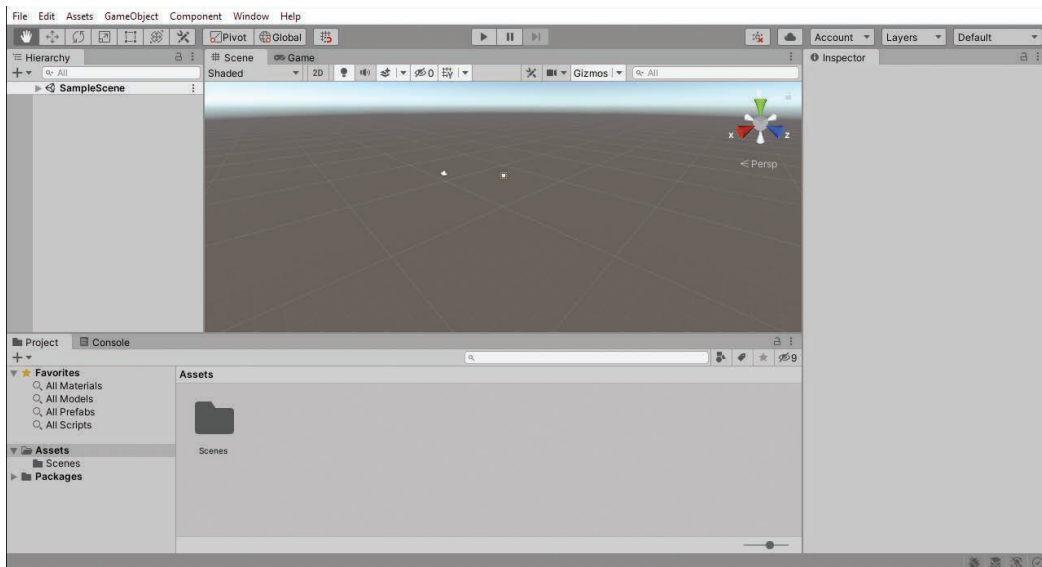


FIGURE 1.6
The Unity interface.

For starters, Unity allows you to determine exactly how you want to work. 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, you can quickly and easily switch back to the built-in default view by going to **Window > Layouts > Default Layout**. While you are playing around, go ahead and try out a few of the other layouts. (I'm a fan of the Wide layout.) If you create a custom layout you like, you can save it by going to **Window > Layouts > Save Layout**. (I used a custom layout called Pearson for the writing of this book.) After you've saved a custom layout, if you accidentally change the layout, you can always get it back. It is worth noting that you can also control the layouts through the Layouts drop-down, located in the upper right of the Unity editor; it is the drop-down that says Default in Figure 1.6.

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 layouts to find the one that works best for you. You will be working with Unity a lot, and it pays to set up your environment in a way that is comfortable for you.

Duplicating a view is a fairly straightforward process as well. You can simply right-click any view *tab* (such as Inspector in Figure 1.7) and hover the mouse cursor over **Add Tab**, and a list of views pops up for you to choose from (see Figure 1.7). You may wonder why you would want to duplicate a view. Say that in a view-moving frenzy, you accidentally close a 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 4 Split built-in layout by going to **Window > Layouts > 4 Split**. (If you have already created a layout that you like, be sure to save it before you check out 4 Split.)

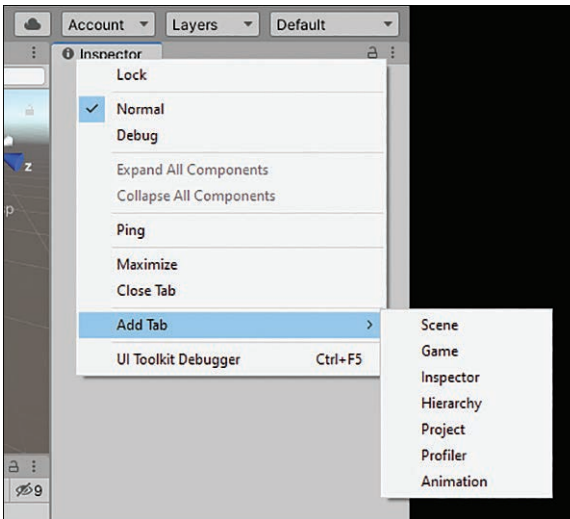


FIGURE 1.7
Adding a new tab.

Now, without further ado, let's look at the specific views in the Unity editor.

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 1.8). This is the window that shows all the assets and organization of a project. When you create a new project, you see a folder section called Assets. If you go to the folder on your hard drive where you save the project, you 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 file or folder appears in the explorer (and vice versa). You can move items in the Project view simply by dragging and dropping. Unity enables you to place items inside folders or reorganize your project on the fly.

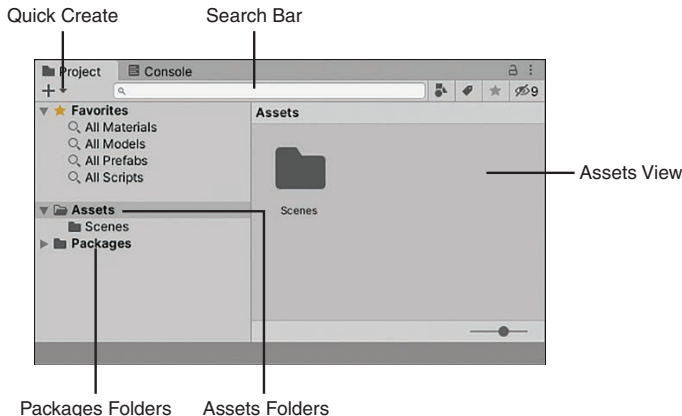


FIGURE 1.8
The Project view.

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, a *game object* is an object that is part of a scene or a level. You can create assets from game objects, and you can create game objects from assets.

NOTE

Packages

When looking in the folder for a project, you can see the Assets folder just as it appears in the editor. You may also notice, however, that there isn't a Packages folder, even though one appears in the Project view in the editor. This is a special section where you can view the packages that have been added to a project. You can think of a package as a type of add-on or module that can change the functionality of a project. Unity has made many of them available, and you can even make your own. You will learn more about packages later, as they become important.

CAUTION

Moving Assets

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

When you click a folder in the Project view, the contents of the folder are displayed under the Assets section on the right. As you can see in Figure 1.8, the Assets section contains a single

folder, named Scenes. If you open the Scenes folder, you see the contents of the folder—a single scene—listed on the right. To create assets, you simply click the **Create** drop-down (that is, the + drop-down). This menu enables you to add all manner of assets and folders to a project.

TIP

Project Organization

Organization is extremely important in project management. As your projects get bigger, the number of assets will grow, and eventually finding anything will be a chore. You can 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 use a folder inside another folder, make sure the structure makes sense. Folders should become more specific and should not be vague or generalized.

Following these few simple rules will really make a difference.

The Favorites buttons enable you to quickly select all assets of a certain type. This makes it possible to get an “at a glance” view of your assets. When you click one of the Favorites buttons (All Models, for instance) or perform a search with the built-in search bar, you can narrow down the results between assets and packages (or both; see Figure 1.9). With a little practice, finding exactly what you need will become a breeze!

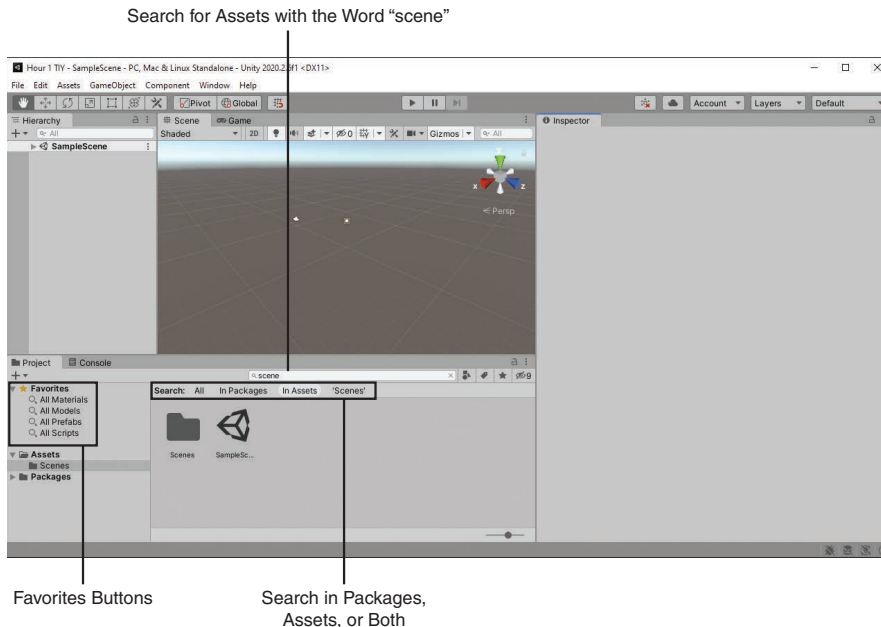


FIGURE 1.9
Searching the Project view.

The Hierarchy View

In many ways, the Hierarchy view (see Figure 1.10) is a lot like the Project view. The difference is that the Hierarchy view shows all the items in the current scene instead of in 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 Directional Light game objects. As you add items to a scene, they appear in the Hierarchy view. Just as 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.

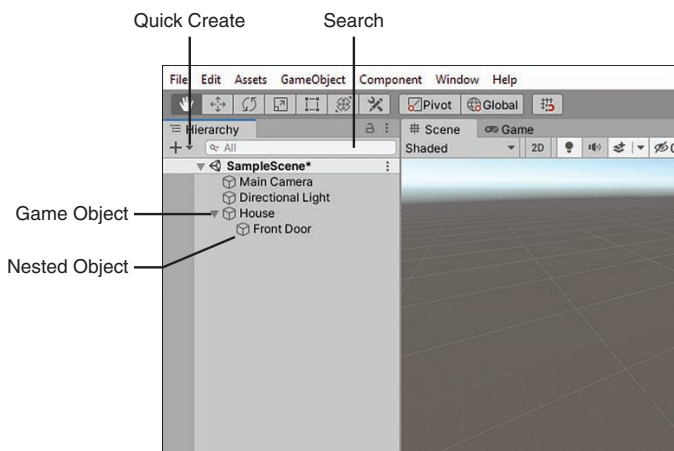


FIGURE 1.10
The Hierarchy view.

TIP

Nesting

Nesting is a way of establishing a relationship between two or more items. In the Hierarchy view, clicking and dragging an item onto another item nests the dragged item under the other one. This creates what 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 can tell when an object is nested because it becomes indented. As you will see later, nesting objects in the Hierarchy view can affect how they behave.

TIP

Scenes

Scene is the term Unity uses to describe what you might already know as a level or map. As you develop a Unity project, each collection of objects and behaviors should be its own scene. For

example, 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.

The Inspector View

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

In Figure 1.11, you can see the Inspector view after the Main Camera object is selected from the Hierarchy view.

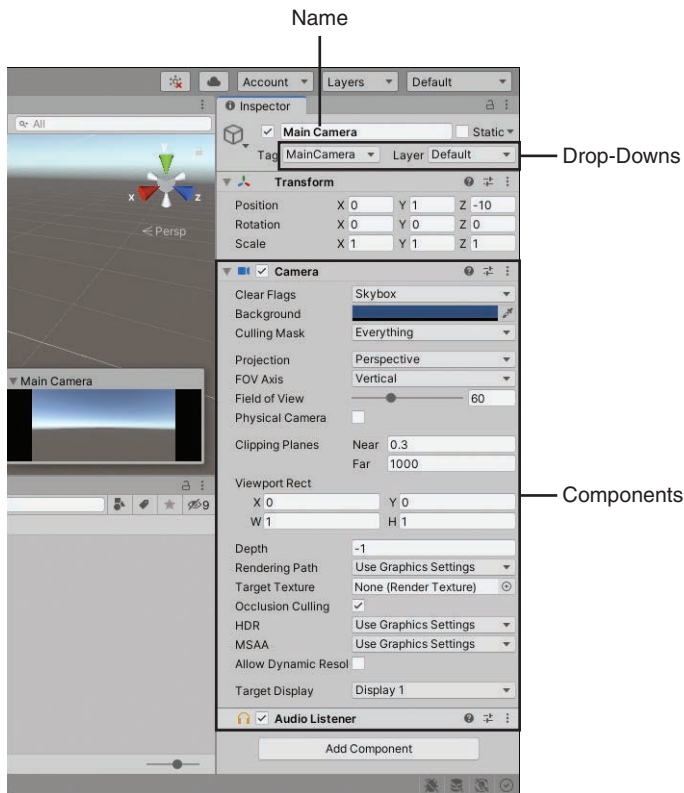


FIGURE 1.11
The Inspector view.

Let's break down some of the functionality available in the Inspector view:

- ▶ Unchecking the box next to an object's name disables it and ensures that it does not appear in the scene. Objects are enabled (that is, their boxes are checked) by default.

- ▶ Drop-down lists (such as the Layer and Tag lists, which are discussed 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 are 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 Camera and Audio Listener in Figure 1.11). You can disable these components by unchecking them or remove them by right-clicking and selecting **Remove Component**.
- ▶ You can add components by clicking the **Add Component** button.

CAUTION

Changing Properties While Running a Scene

The capability to change the properties of an object and see those changes reflected immediately in a running scene is very powerful. It enables you to tweak things like movement speed, jumping height, and collision power on the fly, without stopping and starting the game. Keep in mind, though, that any changes you make to the properties of an object while a scene is running are reverted when the scene finishes. If you make a change and like the result, be sure to remember what it was so you can set it again when the scene is stopped.

The Scene View

The Scene view is the most important view you work with in Unity because it enables you to see your game visually as it is being built (see Figure 1.12). Using the mouse controls and a few hotkeys, you can move around inside a scene and place objects where you want them. This gives you an immense level of control.

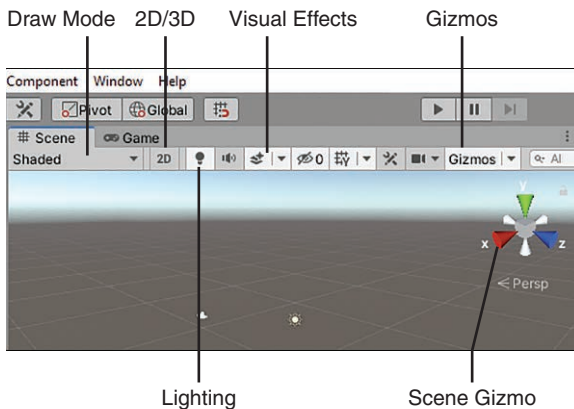


FIGURE 1.12
The Scene view.

In a little bit, you will learn about moving around within a scene, but for now, let's focus on the controls that are part of the Scene view:

- ▶ **Draw mode:** 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 that in 2D view, the scene gizmo (described later this hour) is not visible.
- ▶ **Scene lighting:** This control determines whether objects in the Scene view are 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.
- ▶ **Visual effects:** This control determines whether items like skyboxes, fog, and other effects appear in the Scene view.
- ▶ **Hidden objects:** This control toggles the visibility of hidden objects in the Scene view.
- ▶ **Scene grid:** This control allows you to configure the Scene grid.
- ▶ **Tools:** This control toggles the Component Editor tool panel.
- ▶ **Scene camera:** This control allows you to configure the Scene camera (the camera used to view the scene while editing).
- ▶ **Gizmo selector:** This control enables you to choose which *gizmos*—that is, indicators that help with visual debugging or aid in setup—appear in the Scene view. This control also determines whether the placement grid is visible.
- ▶ **Scene gizmo:** This control shows which direction you are currently facing and aligns the Scene view with an axis.

NOTE

The Scene Gizmo

The scene gizmo gives you a lot of power over the Scene view. As you can see in Figure 1.12, the control has X, Y, and Z indicators that align with the three axes. This makes it easy to tell exactly which way you are looking in the scene. You will learn more about axes and 3D space in Hour 2, “Game Objects.” The scene gizmo also gives you active control over the scene alignment. If you click one of the gizmo's axes, the Scene view immediately snaps to that axis and gets set to a direction such as top or left. Clicking the box in the center of the gizmo toggles between Iso and Persp modes.

Iso, which stands for *isometric*, is the 3D view with no perspective applied. *Persp*, which stands for *perspective*, is the 3D view with perspective applied. Try out these settings for yourself and see how they affect the Scene view. You'll notice that the icon changes from parallel lines for isometric to diverging lines like crow's feet for perspective.

The Game View

The last view we need 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 function in the Game view just as they would if the project were fully built. Figure 1.13 shows what the Game view looks like. Note that although the Play, Pause, and Step buttons are not technically part of the Game view, they control the Game view and therefore are included in this figure.

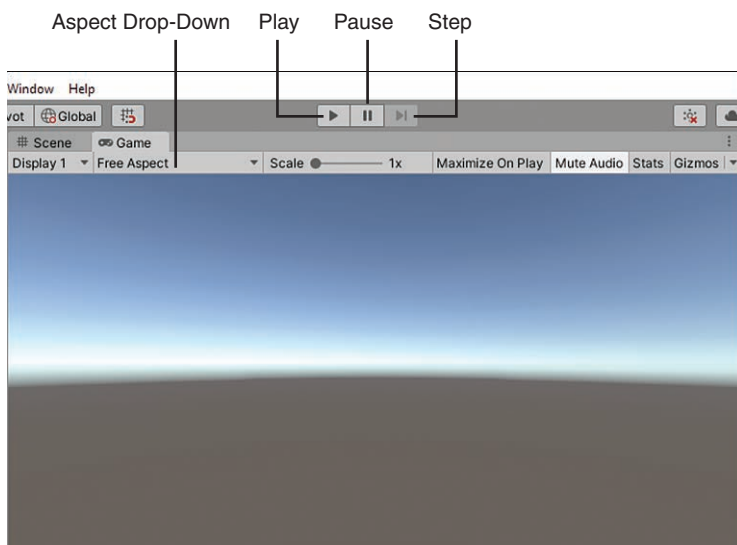


FIGURE 1.13
The Game view.

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 appears in the editor and begins displaying the game.

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

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

- ▶ **Pause:** The Pause button pauses the execution of the currently running Game view. When a game is paused, it maintains its state and stays exactly where it was when the Pause button was clicked. Clicking the Pause button again causes the game to continue running.
- ▶ **Step:** The Step button works while the Game view is paused and causes a single frame of the game to execute. This effectively allows you to “step” through the game slowly and debug any issues that exist. Clicking the Step button while the game is running causes the game to pause.
- ▶ **Aspect drop-down:** From this drop-down menu, you can choose the aspect ratio you want for the Game view window while running. The default is Free Aspect, but you can change it 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 option is turned off, and a running game is only the size of the Game view tab.
- ▶ **Mute Audio:** This button turns off the sounds when playing the game. This is handy when the person sitting next to you is getting tired of hearing your repeated playtesting!
- ▶ **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 a scene. The stats are turned 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. Game view gizmos are not displayed 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 a 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 a game is running and the Pause button is pressed, the game stops running but 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 resumes execution at the point at which it was paused, whereas an off game begins executing at the beginning.

Honorable Mention: The Toolbar

Although not a view, the toolbar is an essential part of the Unity editor. Figure 1.14 shows the toolbar components:

- ▶ **Transform tools:** These buttons enable you to manipulate game objects and are covered in greater detail in later hours. Pay special attention to the button that resembles a hand. This is the Hand tool, and it is described later this hour.
- ▶ **Transform gizmo toggles:** These toggles allow you to manipulate how gizmos appear in the Scene view. You can leave them alone for now.
- ▶ **Game view controls:** These buttons control the Game view.
- ▶ **Account and Services controls:** These buttons allow you to manage the Unity account you are using as well as the services you are using in your project.
- ▶ **Layers drop-down:** This menu determines which object layers appear in the Scene view. By default, everything appears in the Scene view. Leave this alone for now. Layers are covered in Hour 5, “Lights and Cameras.”
- ▶ **Layout drop-down:** This menu allows you to quickly change the layout of the editor.



FIGURE 1.14
The toolbar.

Navigating the Unity Scene View

The Scene view gives you a lot of control over the construction of a 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 different ways to change your position and navigate the Scene view.

TIP

Zoom

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

The Hand Tool

The Hand tool (hotkey: **Q**) provides a simple way to move about the Scene view with the mouse (see Figure 1.15). This tool is 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. (Don't worry about the other buttons next to the Hand tool yet. They are covered a little bit later.)



FIGURE 1.15
The Hand tool.

TABLE 1.1 The Hand Tool Controls

Action	Effect
Click-drag	Drags the camera around the scene
Hold Alt and click-drag	Orbits the camera around the current pivot point
Hold Alt (Command on Mac) and right-click-drag	Zooms the camera

You can find information on all the Unity hotkeys at <https://docs.unity3d.com/Manual/SceneViewNavigation.html>.

CAUTION

Different Cameras

When working in Unity, you deal with two types of cameras. The first type 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 than a camera in the traditional sense. It determines what you can see in the Scene view. When this hour's instructions mention the camera, they are talking about the second type. You do not actually manipulate the game object camera.

Flythrough Mode

Flythrough mode enables you to move about a scene using a traditional first-person control scheme. This mode feels like home for anyone who plays first-person games (such as first-person shooters). 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 while your mouse cursor is over the Scene view puts you into Flythrough mode. All the actions laid out in Table 1.2 require that the right mouse button be held down.

TABLE 1.2 Flythrough Mode Controls

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

TIP

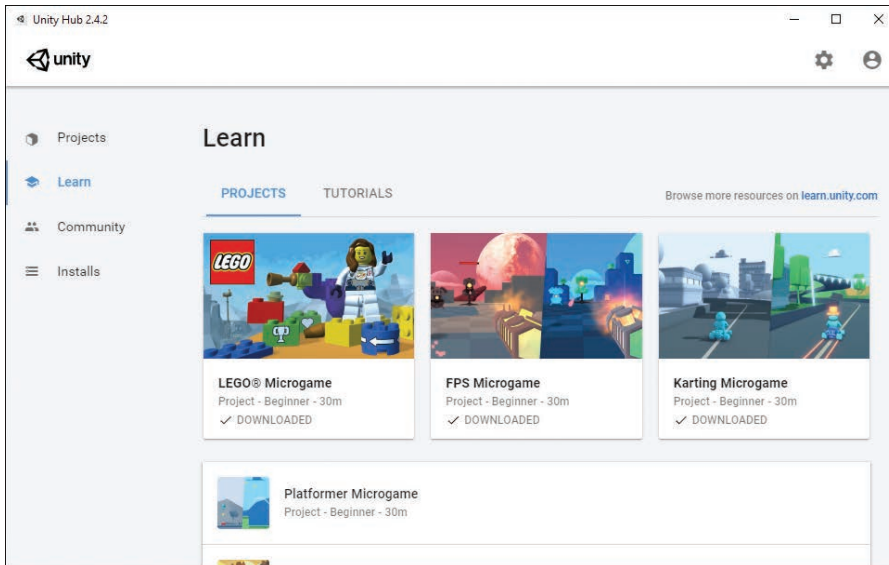
Snap Controls

You have many ways to attain precise control over scene navigation. Sometimes, you just want to quickly get around a scene, though. For times like these, it is good to use what I call *snap controls*. If you want to quickly navigate to, and zoom in on, a game object in a scene, you can do so by highlighting the object in the Hierarchy view and pressing **F** (which stands for *Frame Select*). The scene then “snaps” to that game object. You can also achieve the same effect by double-clicking any object in the Hierarchy view. 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 practice using the snap controls, and navigating through your scene quickly will become a snap!

TIP

Additional Learning

When using the Unity Hub, your eyes may have magically been drawn to the Learn section (see Figure 1.16). You can click the **Learn** button to see Unity’s new learning resources. These resources are a fantastic augmentation and are well worth your time if you’d like even more practice getting to know the basics of the Unity engine. (I am a little biased: I helped make them.)

**FIGURE 1.16**

The Learn section of the Unity Hub.

Summary

In this hour, you took your first look at the Unity game engine. You started 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. The big difference is that an asset has 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. Do I need to memorize them all right away?

A. Not at all. Most controls and options are already set to default states that cover most situations. As your knowledge of Unity grows, you will continue to learn more about the different controls available to you. This lesson is just meant to show you what's there and to give you a bit 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 Pro to make commercial games.
2. Which view enables you to manipulate objects in a scene visually?
3. True or false: You should always move your asset files around within Unity instead of using the operating system's file explorer.
4. True or false: You manage Unity projects and installations from within the editor.
5. What mode do you enter in the Scene view when you hold down the right mouse button?

Answers

1. False. You can make games with Unity Personal or Unity Plus.
2. Scene view
3. True. This helps Unity keep track of the assets.
4. False. You manage projects and editor installations through the Unity Hub.
5. Flythrough mode

Exercise

Take a moment to practice the concepts presented in this hour. 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 folder in the Project view by right-clicking **Assets** and selecting **Create > Folder**. Name the folder **Scenes**.
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 give it a descriptive name.

4. Add a cube to your scene. You can do this in one of three ways:
 - ▶ Click the **GameObject** menu at the top of the editor and select **3D Object > Cube**.
 - ▶ Click **Create > 3D Object > Cube** in the Hierarchy view.
 - ▶ Right-click in the Hierarchy view and select **3D Object > Cube**.
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 by using Flythrough mode, the Hand tool, and snap controls. Use the cube as a point of reference as you navigate.

Index

Symbols & Numerics

- “2 Audio Listeners” message, 77
- 2D audio, 378, 383
- 2D games, 205–206
 - animations, 306
 - colliders, 216–218
 - objects, 41, 42
 - rigidbodies, 216
 - tilemaps, 221
 - adding to a scene, 222–223
 - angled, 224
 - creating, 222
 - palettes, 225
- 2D Scene view, 206–207
- 3D artists, 303
- 3D audio, 378, 383
 - properties, 383
- 3D models, importing, 44
- 3D objects, 41
 - built-in, 42–43
 - triangles, 42

A

- accelerometers, 396, 398
 - axis mismatches, 397
 - designing for, 397
 - using, 397
- access modifiers, 129
- accessing the transform, 154
- Amazing Racer
 - adding fog, 106
 - adding textures, 105
 - character controller, 107–108
 - design, 101, 103–104
 - game control objects, 109–111
 - gamification, 108–109
 - playtesting, 114–115
 - rules, 102
 - scripts
 - adding, 112
 - connecting, 113–114
 - sculpting the world, 104–105
- anchors, 243–244
- Android, 392
- angled tilemaps, 224

animation tools

- Curves Editor, 313–314
- Record mode, 311–313

Animation window, 306–308**animation(s), 301, 303. See also****timelines; visual scripting**

- 2D, 303
- configuring, 321
- creating, 305–306, 308
- Idle, 323–325, 332–334
- making an object spin, 309–310
- moving the timeline, 310
- preparations for, 323
- rigs, 301–302, 319
- settings, 325
- splicing a sprite sheet, 304
- timing, 310
- WalkForwardStraight, 326–327
- WalkForwardTurnRight, 327–329

Animator view, 331–332**animators, 317–318**

- blend trees, 335–337
- creating, 329–331
- importing a model, 319
- parameters, 334
- scripting, 338–339
- states, 335–337
- transitions, 337–338

Apple, 392**apps, Unity Remote, 394****area lights, 86****arena (Chaos Ball)**

- creating, 177
- finishing, 180

arithmetic operators, 130–131**aspect ratio, 241–242****Asset Store**

- downloading models from, 46–47
- models and, 45
- workflow changes, 45

assets, 10, 47

- applying, 52–53
- importing, 65
- materials, 50
- prefabs, 191, 192
 - adding instances to a scene, 196–197
 - breaking links, 200
 - creating, 194, 195–196
 - inheritance, 192
 - instances, 192
 - instantiating, 201
 - modifying, 197–199
 - nesting, 195
 - structure, 192–193
 - terminology, 192
 - updating, 199–200
 - variants, 195

- shaders, 49
 - properties, 51
 - Standard, 427

- textures, 48, 49
 - creating, 69

- timeline, creating, 343–344

assignment operators, 131–132**async scene loading, 406****attaching scripts, 113, 123–124****audio, 377**

- 2D, 378, 383

- 3D, 378, 383

- properties, 383

clips, 378

- changing, 386
- importing, 380

listeners, 377–378**mixers, 386**

- creating, 386–387
- sending audio to, 387–388

Mute Audio button, 380–381**priorities, 380****scripting, 384****sources, 379**

- starting and stopping, 384–386

testing, 380–382**Audio Source component, properties, 379–380****axis, properties, 149****B****baking, 82****billboards, 72****blend trees, 335–337****Boolean operators, 133****bounce physics, 179–180****Box2D, 216****breaking prefab links, 200****Build Settings window, 413–414****building**

- versus deployment, 404
- games, 412–413

built-in 3D objects, 42–43

built-in game objects, 29

built-in methods, 127

buttons, 248–249

- On Click(), 250
- creating, 250–251
- properties, 249

C

C#

- syntax, 128
- variable types, 128

calling methods, 146–147

cameras, 19, 90, 92–93

- layers and, 98
- multiple, 92
- orthographic, 208–209
- picture-in-picture, 94
- properties, 90–91
- split-screen, 93

canvas, 240, 245

- adding to a scene, 240
- performance and, 240
- previewing changes, 244
- render modes
 - Screen Space–Camera, 253–254
 - Screen Space–Overlay, 253
 - World Space, 254

Captain Blaster, 259

- 2D world, 260–261
- bullets
 - creating, 267
 - script, 276–278

camera, 261–262

concept, 260

design, 259

DestroyOnTrigger script, 274

game entities, 263

game manager, 270–271

improvements, 278

meteors

- creating, 265–267

- script, 271–273

- spawn script, 273–274

organization, 263

requirements, 260

rules, 260

scrolling background,
262–263

ShipControl script, 274–276

spaceship

- creating, 264–265

triggers, 267

UI (user interface), 268–269

Chaos Ball, 175–176, 182–184

arena, 177

- creating, 177

- finishing, 180

bounce physics, 179–180

character controllers,
181–182

colored balls, 184–185

control objects, 185

game manager, 187–188

goals, 185–187

improving, 189

requirements, 176

texturing, 178–179

character controllers

- “2 Audio Listeners” message,
77

- adding to a scene, 76

- Amazing Racer, 107–108

- Chaos Ball, 181–182

- First Person, 108

clips, 342, 348, 378

- blending on a track, 351–352

- changing, 386

- importing, 380

- sequencing, 349

**colliders, 163–164, 165–166,
169**

- 2D, 216–218

- composite, 235–236

- properties, 164

- tilemap, 233–235

Collision module, 292–293

Color by Speed module, 290

**Color over Lifetime module,
289–290**

comments, 126

components

- Audio Source, 379–380

- GetComponent, 153–154

- modifying, 158

- presets, 252

- transform, 154

composite colliders, 235–236

conditionals, 133

- if statement, 134

- single-line, 136

- if/else if, 135–136

- if/else statement, 134–135

- troubleshooting, 134

conditions, 103

configuring

- animations, 321
- sprites, 228–229

connecting scripts, 113–114**consolidating objects, 178****control objects, Chaos Ball, 185****controls**

- Gauntlet Runner, 367
- particle system, 282–283

cookies, 88–89

- adding to a spotlight, 89

coordinate systems, 27

- local coordinates, 28–29
- origin, 27
- syntax, 27
- translation, 32–33, 36
- world coordinates, 28–29

creating

- animation(s), 305–306
- animators, 329–331
- arena (Chaos Ball), 177
- audio mixers, 386–387
- buttons, 250–251
- game objects, 30
- palettes, 226
- particle systems, 281–282
- prefabs, 194, 195–196
- scripts, 120–121, 122
- sorting layers, 214–215
- sprites, 207–208
- tilemaps, 222
- tiles, 228–229
- timeline assets, 343–344
- variables, 128

cross-platform settings, 410–411**Curves Editor, 297–298, 313–314****Custom Data module, 296****custom tiles, 227****D****dark scenes**

- troubleshooting, 408

Default module, 285–286**deployment, versus building, 404****dimensions, 26–27****directional lights, 85**

- adding to a scene, 86

downloading

- models from the Asset Store, 46–47
- Unity Hub, 2–3

draw order, 213**duplicating, views, 9****E****effects, subtlety, 69****emission module, 286–287****emissive material, 86****equality operators, 132–133****errors, 69****EventSystem object, 240****External Forces module, 291****F****falling through the world, 108**

- troubleshooting, 77

Find method, 156–158**finding, objects, 155–158****First Person Character Controller, 108****flattening terrain, 63****Flythrough mode, 19–20****fog, 106**

- properties, 106

Force over Lifetime module, 289**G****game control objects, Amazing Racer, 109–111****game development**

- design, 101
- rules, 102

game manager, 103

- Captain Blaster, 270–271
- Chaos Ball, 187–188
- Gauntlet Runner, 368–370

game objects, 29, 36

- built-in, 29
- creating, 30
- nested, 38
- rotation, 33–34
- scaling, 34–35
- transforms, 30–31, 38

Game view, 16–17**games**

- Amazing Racer
 - adding fog, 106
 - adding textures, 105
- character controller, 107–108

- design, 101, 103–104
- game control objects, 109–111
- gamification, 108–109
- playtesting, 114–115
- rules, 102
- scripts, 112, 113–114
- sculpting the world, 104–105
- building, 412–413
- Captain Blaster, 259, 271–273
 - 2D world, 260–261
 - bullets, creating, 267
 - bullets, script, 276–278
 - camera, 261–262
 - concept, 260
 - design, 259
 - DestroyOnTrigger script, 274
 - game entities, 263
 - game manager, 270–271
 - improvements, 278
 - meteors, creating, 265–267
 - meteors, spawn script, 273–274
 - organization, 263
 - requirements, 260
 - rules, 260
 - scrolling background, 262–263
 - ShipControl script, 274–276
 - spaceship, creating, 264–265
 - triggers, 267
 - UI (user interface), 268–269
- Chaos Ball, 175–176, 182–184
 - arena, 177
 - bounce physics, 179–180
 - character controllers, 181–182
 - colored balls, 184–185
 - control objects, 185
 - creating the arena, 177
 - finishing the arena, 180
 - game manager, 187–188
 - goals, 185–187
 - improving, 189
 - requirements, 176
 - texturing, 178–179
- Gauntlet Runner
 - animating the player, 365–367
 - Collidable script, 371–372
 - concept, 357
 - controls, 367
 - design, 357
 - entities, 362
 - game manager script, 368–370
 - ground, creating, 360–361
 - ground, scrolling, 361
 - improvements, 375
 - linking scripts, 373–375
 - obstacles, 363–364
 - player script, 370–371
 - power-ups, 362, 363
 - preparing the scene, 359
 - requirements, 358
 - rules, 358
- Spawner script, 372–373
- trigger script, 367
- trigger zone, 364
- world, 359
- off, 17
- paused, 17
- running, 17
- Gauntlet Runner**
 - animating the player, 365–367
 - concept, 357
 - controls, 367
 - design, 357
 - entities, 362
 - ground
 - creating, 360–361
 - scrolling, 361
 - improvements, 375
 - obstacles, 363–364
 - power-ups, 362, 363
 - preparing the scene, 359
 - requirements, 358
 - rules, 358
 - scripts
 - Collidable, 371–372
 - game manager, 368–370
 - linking, 373–375
 - player, 370–371
 - Spawner, 372–373
 - trigger, 367
 - trigger zone, 364
 - world, 359
- GetComponent, 153–154**
- gizmos, positions, 36–37**
- grass**
 - applying to terrain, 72–73
 - distance and, 74

painting, 71–72
 realism and, 73
 Wind Settings, 76
grid object, 224

H

halos, 87–88
Hand tool, 19
hazards of transformations, 35–36
heightmap
 applying to terrain, 60–61
 calculating height, 61
 formats, 61
 sculpting, 59–60
HelloWorldScript, 125
 class declaration section, 126
 methods, 126
 using section, 125
Hierarchy view, 12

I

IDEs (integrated development environments), 122
Idle animation, 323–325, 332–334
if statement, 134
 single-line, 136
if/else if statement, 135–136
if/else statement, 134–135
images, 246, 247, 248
 properties, 246

importing
 3D models, 44
 audio clips, 380
 models, 319
 scripts, 113
 sprites, 210
 terrain assets, 65
Inherit Velocity module, 288
inheritance, 192
input, 147–148
 key codes, 150–151
 mouse, 152–153
 multi-touch, 398–399
 reading in, 150
 scripting, 149
 specific key, 150, 151–152

Input Manager, 148

Inspector view, 13–14

installing
 mobile platform modules,
 393–394
 Unity, 1–2
 Unity editor, 3–5
 Unity Hub, 2–3

instances, 192

instantiating, prefabs, 201

iteration, 137
 for loop, 138
 while loop, 137

J-K-L

key codes, 150–151

layers, 94–98

layouts, 8

Lifetime by Emitter Speed module, 288

lights, 81, 82
 area, 86
 baking, 82
 cookies, 88–89
 directional, 85
 adding to a scene, 86
 halos, 87–88
 layers and, 98
 point, 83
 adding to a scene, 84
 properties, 83
 properties, 81
 spotlights, 84–85
 adding a cookie, 89
 adding to a scene, 85

Lights module, 295

Limit Velocity over Lifetime module, 288

listeners, 377–378
 multiple, 92

LoadScene() method, 405

local coordinates, 28–29

locking the Timeline window, 345, 350

logical operators, 133

for loop, 138

loops
 for, 138
 while, 137

M

materials, 50. See also physics materials
 applying, 52–53
 emissive, 86

mesh. See *also* **terrain**

terrain, 57

mesh(es), 43

default scaling, 45

models and, 42

previewing, 320

.meta files, 360

method block, 143

methods, 126, 142, 144

built-in, 127

calling, 146–147

Find, 156–158

LoadScene(), 405

naming, 142

OnGUI(), 400

parameters, 143

return type, 143

saving data, 408

signature, 143

syntax, 146

troubleshooting, 147

using, 146

writing, 144–146

mobile development

accelerometers, 396, 398

axis mismatches, 397

designing for, 397

using, 397

installing mobile platform

modules, 393

multi-touch input, 398–399

preparing for, 391

setting up your environment,
392–393

testing device setup, 395

tracking touches, 399–400

Unity Remote, 394

models, 41, 43

Asset Store and, 45

components, 158

downloading from the Asset
Store, 46–47

importing, 44, 319

mesh and, 42, 43

modifying

prefabs, 197–199

public variables, 130

modules

Collision, 292–293

Color by Speed, 290

Color over Lifetime,
289–290

Custom Data, 296

Emission, 286–287

External Forces, 291

Force over Lifetime, 289

Inherit Velocity, 288

Lifetime by Emitter Speed,
288

Lights, 295

Limit Velocity over Lifetime,
288

mobile platform, installing,
393–394

Noise module, 291

particle systems, 283–284

Default, 285–286

Renderer, 296–297

Rotation by Speed, 290

Rotation over Lifetime, 290

Shape, 287

Size by Speed, 290

Size over Lifetime, 290

Sub Emitter, 294–295

Texture Sheet Animation, 295

Trails, 295–296

Triggers, 294

Velocity over Lifetime,
287–288

mouse input, 152–153

multiple cameras, 92

multi-touch input, 398–399

Mute Audio button, 380–381

muting tracks, 350

N

naming

methods, 142

scripts, 123

navigating, Scene view, 18

nested objects, 38

nesting, 12

prefabs, 195

New Project dialog, 6–7

Noise module, 291

O

objects

consolidating, 178

EventSystem, 240

finding, 155–158

grid, 224

keeping, 406

persistent, 407–408

persistent, 406

transforming, 154–155

OnGUI() method, 400

operators, 130

arithmetic, 130–131

assignment, 131–132

Boolean, 133

equality, 132–133

logical, 133

Order in Layer setting, 215

origin, 27

orthographic cameras, 208–209

P

package files, 66

packages, 10

Paint Details tool, 71–72

Paint Holes tool, 63

Paint Texture tool, 66–67

Paint Trees tool, 70–71

painting

grass, 71–72

textures, 68

tiles, 231–232

trees, 70–71

palettes, 225

creating, 226

customizing, 232–233

parameters, 143

animator, 334

particle effects, 283

particle systems, 281

Collision module, 292–293

collisions, 293–294

Color by Speed module, 290

Color over Lifetime module,
289–290

controls, 282–283

creating, 281–282

Custom Data module, 296

Default module, 285–286

Emission module, 286–287

External Forces module, 291

Force over Lifetime module,
289

Gauntlet Runner, 362

Inherit Velocity module, 288

Lifetime by Emitter Speed
module, 288

Lights module, 295

Limit Velocity over Lifetime
module, 288

modules, 283–284

Noise module, 291

Renderer module, 296–297

Rotation by Speed module, 290

Rotation over Lifetime module,
290

Shape module, 287

Size by Speed module, 290

Size over Lifetime module, 290

Sub Emitter module, 294–295

Texture Sheet Animation
module, 295

Trails module, 295–296

Triggers module, 294

troubleshooting, 362–363

value curve, 285

Velocity over Lifetime module,
287–288

particles, custom, 282–283

performance, 73

canvases and, 240

per-platform settings, 410–412

**persistent data and objects, 406,
407–408**

physics

2D colliders, 216–218

rigidbody 2D, 216

physics materials, 166

properties, 167

picture-in-picture, 94

placement options, gizmos, 36–37

**player settings, 410, 412-
C23.01180**

cross-platform, 410–411

per-platform, 410–412

PlayerPrefs, 408, 409

security and, 410

playtesting, 103

Amazing Racer, 114–115

point lights, 83

adding to a scene, 84

properties, 83

prefabs, 191, 192

breaking links, 200

creating, 194, 195–196

inheritance, 192

instances, 192

adding to a scene,
196–197

instantiating, 201

modifying, 197–199

nesting, 195

structure, 192–193

terminology, 192

updating, 199–200

variants, 195

private variables, 129

programming, 120

comments, 126

scripts and, 120

Project view, 9–11

projects

adding terrain, 57–59

organization, 11

Projects section, Unity editor, 5–6

properties

3D audio, 383

Audio Source component,
379–380

axis, 149

of buttons, 249

of cameras, 90–91

of colliders, 164

Collision module, 292–293

Default module, 285–286

Emission module, 286–287

fog, 106

of images, 246

of lights, 81

Limit Velocity over Lifetime
module, 288

Noise module, 291

of physics materials, 167

of point lights, 83

Renderer module, 296–297

of rigidbodies, 162–163

of shaders, 51

of text, 248

Texture Sheet Animation
module, 295

Touch variable, 399

Velocity over Lifetime module,
287–288

public variables, 129

modifying, 130

Q-R

**Raise or Lower Terrain tool,
62–63**

raycasting, 170–171

Record mode, 311–313

Rect, 31, 241, 242–243

render modes

Screen Space–Camera,
253–254

Screen Space–Overlay, 253

World Space, 254

Renderer module, 296–297

rendering, 48

rigidbodies, 161–162, 163

2D, 216

properties, 162–163

rigs, 301–302, 319

preparing, 321–323

rotation, 33–34

Rotation by Speed module, 290

Rotation over Lifetime module, 290

S

saving data, 408

methods, 408

PlayerPrefs, 409

scaling, 34–35

sprites, 213

scene gizmo, 15

Scene view, 14–15

navigating, 18

scenes, 12–13, 403

adding a canvas, 240

adding character controllers,
76

adding directional lights, 86

adding point lights, 84

adding prefab instances,
196–197

adding spotlights, 85

adding sprites, 209

adding tilemaps, 222–223

adding to Build Settings, 405

async loading, 406

change properties and, 14

dark, 408

establishing order, 404

Gauntlet Runner, 359

managing, 403

switching, 405

Screen Space–Camera, 253–254

Screen Space–Overlay, 253

scripts, 109, 119, 120, 141

adding to Amazing Racer, 112

animators and, 338–339

attaching, 123–124

audio and, 384

Captain Blaster, 271–274

bullets, 276–278

DestroyOnTrigger, 274

ShipControl, 274–276

connecting, 113–114

creating, 120–121, 122

- Collidable, 371–372
 - game manager, 368–370
 - linking, 373–375
 - player, 370–371
 - Spawner, 372–373
 - trigger zone, 367
 - HelloWorldScript, 125
 - class declaration section, 126
 - using section, 125
 - importing, 113
 - methods, 126
 - built-in, 127
 - naming, 123
 - programming languages and, 120
 - timelines and, 352–353
 - variables, 128
 - creating, 128
 - scrolling background**
 - Captain Blaster, 262–263
 - Gauntlet Runner, 361
 - sculpting tools**
 - Paint Holes, 63
 - Raise or Lower Terrain, 62–63
 - Set Height, 63
 - Smooth Height, 63
 - security, PlayerPrefs, 410**
 - sending audio to mixers, 387–388**
 - sequencing clips, 349**
 - Set Height tool, 63**
 - shaders, 49**
 - applying, 52–53
 - properties, 51
 - Standard, 427
 - Shape module, 287**
 - ShipControl script, 274–276**
 - shortcuts, terrain, 63**
 - single-line if statements, 136**
 - Size by Speed module, 290**
 - Size over Lifetime module, 290**
 - skyboxes, 106–107**
 - Smooth Height tool, 63**
 - snap controls, 20**
 - sorting, UI elements, 252**
 - sorting layers, 213–215**
 - spawn point, 103**
 - spawning, 103**
 - specific key input, 150, 151–152**
 - splicing a sprite sheet, 304**
 - split-screen cameras, 93**
 - spotlights, 84–85**
 - adding a cookie, 89
 - adding to a scene, 85
 - sprite modes, 210–212**
 - sprites. *See also* tiles**
 - adding to a scene, 209
 - configuring, 228–229
 - creating, 207–208
 - draw order, 213
 - importing, 210
 - missing, 215
 - scaling, 213
 - sorting layers, 213–215
 - turning into tiles, 229–230
 - Standard shader, 427**
 - starting and stopping audio, 384–386**
 - Sub Emitter module, 294–295**
 - switching scenes, 405**
- terrain, 57**
- adding to a project, 57–59
 - Amazing Racer, 104–105
 - applying a heightmap to, 60–61
 - applying grass to, 72–73
 - assets, importing, 65
 - flattening, 63
 - grass, painting, 71–72
 - heightmap sculpting, 59–60
 - placing trees on, 71
 - sculpting, 64
 - sculpting tools, 62–63
 - Paint Holes, 63
 - Raise or Lower Terrain, 62–63
 - Set Height, 63
 - Smooth Height, 63
 - settings, 74
 - shortcuts, 63
 - size, 59
 - textures, 65
 - creating, 69
 - painting, 68
 - texturing, 66–67
- Terrain Settings tool, 74–75**
- Tree & Detail Objects settings, 75
 - Wind Settings, 76
- syntax**
- C#, 128
 - coordinate systems, 27
 - methods, 146

testing audio, 381–382

text, 248

properties, 248

Texture Sheet Animation module, 295

textures, 48, 49

for Amazing Racer, 105

applying, 52–53

applying to terrain, 66–67

Chaos Ball, 178–179

cookies, 88–89

adding to a spotlight, 89

creating, 69

painting, 68

unwrap, 49

Tile Palette window

enhanced controls, 232

tools, 230–231

tilemaps, 221

adding to a scene, 222–223

angled, 224

colliders, 233–235

creating, 222

palettes, 225

creating, 226

tiles, 227

creating, 228–229

custom, 227

painting, 230, 231–232

Timeline window, 342, 345

creating a timeline asset,
343–344

locking, 345, 350

Preview mode, 345

recording in, 350

timelines, 344

clips, 342, 348

blending on a track,

351–352

sequencing, 349

scripts and, 352–353

tracks, 342, 346–347

toolbar, 18

Touch variable, 399

tracking touches, 399–400

tracking touches, 399–400

tracks, 342, 346–347

blending clips on, 351–352

muting, 350

Trails module, 295–296

transforms, 30–31, 38

accessing, 154

hazards of, 35–36

Rect, 241, 242–243

transitions, 337–338

translation, 32–33, 36

trees

painting, 70–71

placing on terrain, 71

triggers, 167–169

Captain Blaster, 267

Triggers module, 294

troubleshooting

axis mismatches, 397

conditionals, 134

dark scenes, 408

falling through the world, 77,
108

methods, 147

missing sprites, 215

particle systems, 362–363

U

UI (user interface)

anchors, 242, 243–244

basic principles, 239

buttons, 248–249

On Click(), 250

creating, 250–251

properties, 249

canvas, 240, 245

adding to a scene, 240

performance and, 240

previewing changes, 244

render modes, 252–254

Captain Blaster, 268–269

design, 239

images, 246, 247, 248

properties, 246

sorting elements, 252

text, 248

properties, 248

Unity

2D Scene view, 206–207

Animation window, 306–308

cameras, 19

Flythrough mode, 19–20

Game view, 16–17

Hand tool, 19

Hierarchy view, 12

Inspector view, 13–14

installing, 1–2

interface, 7–9

layouts, 8

LTS version, 5

New Project dialog, 6–7

Order in Layer setting, 215

- package files, 66
- player settings, 410, 412–C23.01180
 - cross-platform, 410–411
 - per-platform, 410–412
- Project view, 9–11
- projects, organization, 11
- Scene view, 14–15
 - navigating, 18
- snap controls, 20
- supported operating systems and hardware, 5
- Tile Palette window, 225
- toolbar, 18
- updates, 2
- views, 7–8
 - duplicating, 9
- Unity editor**
 - Console window, 126–127
 - installing, 3–5
 - Projects section, 5–6

- Unity Hub, downloading and installing, 2–3**
- Unity Remote, 394**
- unwrap, 49**
- updates, Unity, 2**
- updating, prefabs, 199–200**

V

- value curve, 285**
- variables, 128**
 - access modifiers, 129
 - C#, 128
 - creating, 128
 - parameters, 143
 - private, 129
 - public, 129
 - modifying, 130

- scope, 129
- Touch, 399
- Velocity over Lifetime module, 287–288**
- vertices, 42**
- views, 7–8**
 - Animator, 331–332
 - duplicating, 9
- visual scripting, 141**

W-X-Y-Z

- WalkForwardStraight animation, 326–327**
- WalkForwardTurnRight animation, 327–329**
- world coordinates, 28–29**
- World Space, 254**
- writing, methods, 144–146**