O'REILLY®

Unity Game Development Cookbook

Essentials for Every Game

Grayscale Edition

For Sale in the Indian Subcontinent & Select Countries Only*

*Refer Back Cover

Paris Buttfield-Addison,
Jon Manning & Tim Nugent

SPD®

Unity Game Development Cookbook

Essentials for Every Game

Paris Buttfield-Addison, Jon Manning,
and Tim Nugent

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

SPD®

SHROFF PUBLISHERS & DISTRIBUTORS PVT. LTD.
Mumbai Bangalore Kolkata New Delhi

Unity Game Development Cookbook

by Paris Buttfield-Addison, Jon Manning, and Tim Nugent

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles *(http://safari.oreilly.com)*. For more information, contact our corporate/institutional sales department: (800) 998-9938 or *corporate@oreilly.com*.

Acquisition Editor: Rachel Roumeliotis
Editor: Jeff Bleiel
Production Editor: Deborah Baker
Copyeditor: Rachel Monaghan
Proofreader: Kim Cofer
Indexer: Judy McConville
Interior Designer: David Futato
Cover Designer: Karen Montgomery
Illustrator: Rebecca Demarest

Printing History:
March 2019: First Edition

Revision History for the First Edition: 2019-03-12: First Release

See http://oreilly.com/catalog/errata.csp?isbn=9781491999158 for release details.

First Indian Reprint: April 2019

ISBN: 978-93-5213-818-0

While the publisher and the authors have used good faith efforts to ensure that the information and instructions contained in this work are accurate, the publisher and the authors disclaim all responsibility for errors or omissions, including without limitation responsibility for damages resulting from the use of or reliance on this work. Use of the information and instructions contained in this work is at your own risk. If any code samples or other technology this work contains or describes is subject to open source licenses or the intellectual property rights of others, it is your responsibility to ensure that your use thereof complies with such licenses and/or rights.

Published by **Shroff Publishers & Distributors Pvt. Ltd.** B-103, Railway Commercial Complex, Sector 3, Sanpada (E), Navi Mumbai 400705 • TEL: (91 22) 4158 4158 • FAX: (91 22) 4158 4141
E-mail: spdorders@shroffpublishers.com • Web: www.shroffpublishers.com
CIN : U22200MH1992PTC067760 Printed at Jasmine Art Printers Pvt. Ltd., Navi Mumbai.

Table of Contents

Preface. vii

1. Working with Unity. 1
   1.1 Getting Around in Unity 1
   1.2 Working with Game Objects 10
   1.3 Working with Components 11
   1.4 Working with Prefabs 15
   1.5 Working with Scenes 16
   1.6 Managing Assets 17
   1.7 Building Unity Projects 18
   1.8 Accessing Preferences 19

2. Scripting. 21
   2.1 Working with MonoBehaviours 21
   2.2 Using Callbacks 26
   2.3 Creating Frame Rate–Independent Behavior 28
   2.4 Accessing Components 29
   2.5 Finding Objects 31
   2.6 Using Coroutines 32
   2.7 Singletons 35
   2.8 Loading a Level (Scene) 38
   2.9 Storing Data on Disk 42
   2.10 Saving and Loading the Game State 42
   2.11 Managing Objects Using an Object Pool 58
   2.12 Storing Data in Assets Using ScriptableObject 66

3. Input. 69
   3.1 Working with Keyboard Input 69

3.2 Working with Mouse Input 70
3.3 Locking and Hiding the Mouse Cursor 71
3.4 Working with Gamepads 72
3.5 Customizing Unity's Input System 74
3.6 Responding to Pointer Events from the Event System 78

4. Math. 83
4.1 Storing Coordinates of Varying Dimensions Using Vectors 84
4.2 Rotating in 3D Space 89
4.3 Performing Transformations in 3D Space with Matrices 91
4.4 Working with Angles 95
4.5 Finding the Distance to a Target 97
4.6 Finding the Angle to a Target 98

5. 2D Graphics. 101
5.1 Importing Sprites 101
5.2 Adding a Sprite to the Scene 103
5.3 Creating a Sprite Animation 104
5.4 Creating a Sprite with 2D Physics 105
5.5 Customizing Sprite Collision Shapes 105
5.6 Using a Composite Collider 108
5.7 Using the Sprite Packer 110
5.8 Applying Forces to 2D Objects 110
5.9 Creating a Conveyor Belt 112
5.10 Using a Custom Material for Sprites 114
5.11 Managing Sprite Sorting 116
5.12 Using Sorting Groups 117
5.13 Creating a 2.5D Scene 119

6. 3D Graphics. 121
6.1 Creating a Simple Material 121
6.2 Controlling a Material's Property Through a Script 123
6.3 Creating an Unlit Material 125
6.4 Setting Up a Material Using Textures 125
6.5 Making a Material Use a Shader 128
6.6 Setting Up a Bloom Effect Using Post-Processing 128
6.7 Using High-Dynamic-Range Colors 133
6.8 Setting Up a Project to Use a Scriptable Render Pipeline 136
6.9 Creating a Shader Using the Shader Graph 138
6.10 Creating a Glowing Effect Using the Shader Graph 140
6.11 Exposing Properties from a Shader Graph 142
6.12 Animating a Shader over Time 144

6.13 Controlling the Speed of an Animated Shader 146
6.14 Using a Subgraph to Reuse Graph Components 147
6.15 Implementing a Dissolve Effect Using a Shader Graph 149
6.16 Using Baked Lighting and Real-Time Lighting 152
6.17 Using Baked Emission Sources 154
6.18 Making Static Objects Cast Shadows on Dynamic Objects 155
6.19 Using Light Probes to Influence Lighting 156
6.20 Using Reflection Probes 159
6.21 Faking a Dynamic Emissive Object 163
6.22 Loading Textures from Disk 164
6.23 Rendering to a Texture 166
6.24 Saving Screenshots 169

7. Physics and Character Control. 171
7.1 Understanding FixedUpdate 171
7.2 Implementing Mouselook 172
7.3 Controlling a 3D Character 175
7.4 Interacting with Switches and Objects 178
7.5 Picking Up and Putting Down Objects 181
7.6 Detecting When an Object Is Touching Another Object 187
7.7 Detecting When an Object Is in a Trigger Area 188
7.8 Implementing Moving Platforms 190
7.9 Implementing Platform Riding 193
7.10 Responding to Being Pushed by Objects 197

8. Animation and Movement. 201
8.1 Animating an Object 202
8.2 Basic Character Movement 206
8.3 Inverse Kinematics 211
8.4 Masked Movement 215
8.5 Blended Movement 220
8.6 Navigation and Animating in Sync 224
8.7 Cinematic Camera Tracking 228
8.8 Automatically Switching Cameras 231
8.9 Keeping Multiple Objects in View 232
8.10 Dollying a Camera 233

9. Gameplay. 237
9.1 Managing Quests 237
9.2 Managing Hitpoints 253
9.3 Creating a Top-Down Camera 256
9.4 Dragging a Box to Select Objects 259

9.5 Creating a Menu Structure 263
9.6 Creating a Wheeled Vehicle 267
9.7 Keeping a Car from Tipping Over 271
9.8 Creating Speed Boosts 272
9.9 Creating a Camera That Orbits Around Its Target 274
9.10 Creating Orbiting Cameras That Won't Clip Through Walls 277
9.11 Detecting When the Player Has Completed a Lap 278

10. Behavior and AI. 289

10.1 Defining a Path That AI Entities and the Player Can Follow 289
10.2 Letting Entities in Your Game Follow a Path 291
10.3 Enemies Detecting When They Can See the Player 293
10.4 Finding a Good Distribution of Random Points (Poisson Disc) 299
10.5 Enemies Detecting Where They Can Take Cover 304
10.6 Building and Using a State Machine 310

11. Sound and Music. 315

11.1 Playing Sounds 315
11.2 Setting Up a Mixer 318
11.3 Using Audio Effects 320
11.4 Using Send and Receive Effects 322
11.5 Ducking 325
11.6 Using Multiple Audio Zones 326
11.7 Playing Audio Through Code 328
11.8 Using a Sound Manager 329

12. User Interface. 335

12.1 Working with UI Controls 336
12.2 Theming Controls 340
12.3 Animating the UI 344
12.4 Creating a List of Items 347
12.5 Fading Out List Items 352
12.6 Creating Onscreen Position Indicators 354
12.7 Custom Editors 359
12.8 Property Drawers 364
12.9 Attribute Drawers 369
12.10 Asset Processing 372
12.11 Scripted Importers 374
12.12 Wizards 378

Index. 381

Preface

Welcome to the *Unity Game Development Cookbook*! We're incredibly excited about this book, and really proud of our work on it. We hope you find it a useful companion to your game development using the Unity engine.

Unity is an amazing platform, and the more you use it the more you'll come up with new ideas on how you can use it, why you could use it, and so on! It's all very exciting. Trust us, we've been there.

As Secret Lab, we use Unity to build most of our video games. We're best known for building children's games in Australia, as well as our contributions to the BAFTA- and IGF-winning *Night in the Woods* (*http://nightinthewoods.com*). We discovered Unity more than a decade ago, and we haven't stopped using it since.

In this book, we've assembled a selection of recipes to solve common problems that we found ourselves solving over and over as we build video games using Unity. This book won't cover every single tiny thing you want to know about Unity, but it hopefully covers the majority of things you'd be doing day-to-day in video game development.

These recipes are about things we wish we had someone to ask about back when we were first building our own games. We really hope they help you!

It's a really exciting time to be using Unity. The game development world is getting bigger and more interesting every day, and the tools and technologies we use are better than ever.

Go out and build great games!

> If you want to get more resources for this book, and our other books, check out our website (*http://www.secretlab.com.au*).

Resources Used in This Book

If you want to download all our code from the recipes throughout this book, head to *https://github.com/thesecretlab/unity-game-development-cookbook-1e*. We also have a dedicated page for this book (*http://secretlab.com.au/books/unity-cookbook-1e*), where we keep both the source code as well as other downloads and resources.

Audience and Approach

This book assumes that you know a little of the C# programming language, or a similar language like C++, Java, Go, Swift, Python, or other common languages, but don't know anything about Unity. This book is intended to be a companion to get your games up and running faster.

> If you like our style, and want to read more Unity material from us, you can learn Unity from scratch in *Mobile Game Development with Unity*, also available from the fine folks at O'Reilly.

Our screenshots are taken with macOS, but everything we talk about is generally applicable to Unity on Windows, macOS, or Linux.

Organization of This Book

This book is arranged into 12 chapters:

- Chapter 1 introduces the fundamental concepts you need to know in order to use Unity: game objects, components, scenes, and how to work with the Unity editor itself.
- Chapter 2 covers the scripting interface of Unity, which is the way you write the actual code that defines the way your game works. After we establish the basics, we'll dive into some practical examples, including how to write a save and load system, how to efficiently work with objects, and how to store your data in a way that both your code and Unity can work with easily.
- In Chapter 3, you'll learn how to get input from your user, in the form of keyboard input, the mouse, and from game pads. We'll also discuss how to set up game objects that react to input events, like the user clicking on them.
- Chapter 4 covers the fundamentals of some of the mathematical concepts that are useful to know about when development games, such as vectors, matrices and quaternions. We'll also look at some practical uses of these concepts, like detecting if an object is in front of the player or not.

- Chapter 5 discusses the 2D graphics and physics systems that are built in to Unity. You'll learn how to display sprites, sort them, and make them bounce off each other.
- Chapter 6 covers the material and shading system of Unity: how materials and shaders work, how to build a shader in Unity, and how to get the best results in your scenes.
- In Chapter 7, you'll learn how to make the 3D physics system of Unity do what you need it to in order to support common gameplay tasks, like picking up and throwing objects, and creating a moving platform that the player can ride on.
- Chapter 8 introduces the animation system used in Unity, and covers topics like how to set up a character to blend between different animation states, and how to integrate player movement with character animation. We'll also introduce the camera system in Unity, and discuss how to set up cameras that move to follow targets.
- Chapter 9 is all about creating the gameplay that players interact with. There's a wide range of common gameplay tasks in here, like managing the state of a quest that you've given the player, tracking if a racing car is taking too much of a shortcut, and managing how damage gets dealt between objects in your game.
- In Chapter 10, you'll learn how to add brains to your characters, including the ability to detect when the player can be seen, the ability to navigate a space and avoid obstacles, and the ability to figure out a place where they *can't* be seen.
- Chapter 11 introduces the audio systems in Unity. We'll cover the basics of playing sounds, and then move on to more advanced features, like routing audio to multiple groups, and automatically ducking the music when a character is speaking.
- Chapter 12, our final chapter, covers the tools for building an interface for your players to look at and interact with. We'll cap the whole book off with a section on how to build your *own* tools in Unity, by extending the editor.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`

Shows commands or other text that should be typed literally by the user.

`Constant width italic`

Shows text that should be replaced with user-supplied values or by values determined by context.

This element signifies a tip or suggestion.

This element signifies a general note.

This element indicates a warning or caution.

Using Code Examples

Supplemental material (code examples, exercises, errata, etc.) is available for download from our website (*https://www.secretlab.com.au/books/unity-cookbook-1e*).

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Unity Game Development Cookbook* by Paris Buttfield-Addison, Jon Manning, and Tim Nugent (O'Reilly). Copyright 2019 Secret Lab, 978-1-491-99915-8."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

O'Reilly Online Learning Platform

For almost 40 years, *O'Reilly Media* has provided technology and business training, knowledge, and insight to help companies succeed.

Our unique network of experts and innovators share their knowledge and expertise through books, articles, conferences, and our online learning platform. O'Reilly's online learning platform gives you on-demand access to live training courses, in-depth learning paths, interactive coding environments, and a vast collection of text and video from O'Reilly and 200+ other publishers. For more information, please visit *http://oreilly.com*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *http://bit.ly/unity-game-dev-cb*.

To comment or ask technical questions about this book, send email to *bookquestions@oreilly.com*.

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

Acknowledgments

Jon thanks his mother, father, and the rest of his crazily extended family for their tremendous support.

Paris thanks his amazing mother, without whom he wouldn't be doing much of anything at all really, let alone with enthuasiam. She's about to submit her doctorate for assessment, and he's sure she will be absolutely amazing at it.

Tim thanks his parents and family for putting up with his rather lackluster approach to life.

We'd all like to thank Rachel Roumeliotis, whose skill and advice were invaluable in completing this book, as well as every other book we've written together. We thoroughly look forward to our coffee catchups every few months/year or so!

Thank you to our editor on this book, Jeff Bleiel. You were calm, clear, enthusiastic, and wonderful to work with. We hope to work on more books with you in the future!

Likewise, all the O'Reilly Media staff we've interacted with over the course of writing the book have been the absolute gurus of their fields. O'Reilly Media has a vast array of incredibly talented people on staff, on contract, and generally surrounding them. It's actually terrifying how good they are at their jobs.

A huge thank you to Tony Gray and the Apple University Consortium (AUC) for the monumental boost they gave us and others listed on this page. We wouldn't be writing this book if it weren't for them. And now you're writing books, too, Tony—sorry about that!

Thanks also to Neal Goldstein, who deserves full credit and/or blame for getting us into the whole book-writing racket. We're thankful for the support of the goons at MacLab (who know who they are and continue to stand watch for Admiral Dolphin's inevitable apotheosis), as well as Professor Christopher Lueg, Associate Professor Leonie Ellis, and the staff at the University of Tasmania for putting up with us.

Thanks also to Mars G. for generally being amazing, as well as Dave J. for serving the coffee. If you ever visit Hobart, Tasmania, Australia, make sure you get a coffee at Yellow Bernard. It's the best.

Thanks also to our tech reviewers for their thoroughness and professionalism. The reviews were brilliantly helpful.

Finally, thank you very much for buying our book—we appreciate it! And if you have any feedback, please let us know. You can email us at *lab@secretlab.com.au* and find us on Twitter at @thesecretlab (*https://twitter.com/thesecretlab*).

CHAPTER 1

Working with Unity

To make things in Unity, it's important to understand how Unity works and how to work with it as a piece of software. In this chapter, we'll take a tour through the interface and features of Unity, so that you'll be ready to start building things in the editor.

> If you're new to Unity, we recommend reading all the recipes in this chapter, as well as in Chapter 2, before trying other recipes in this book.

This chapter also introduces you to many of the terms used in the Unity ecosystem, which will help you make sense of the rest of this book.

1.1 Getting Around in Unity

Problem

You want to learn how to navigate the Unity editor, and understand what each component does, how to use it, and how to customize it.

Solution

Unity's user interface is organized along *windows* of content. Each different view can do different things. In this recipe, we'll present some of the most important ones and talk about how to arrange the editor to suit your needs.

When you launch Unity and create a new project, you'll be taken to the editor's main window (Figure 1-1).

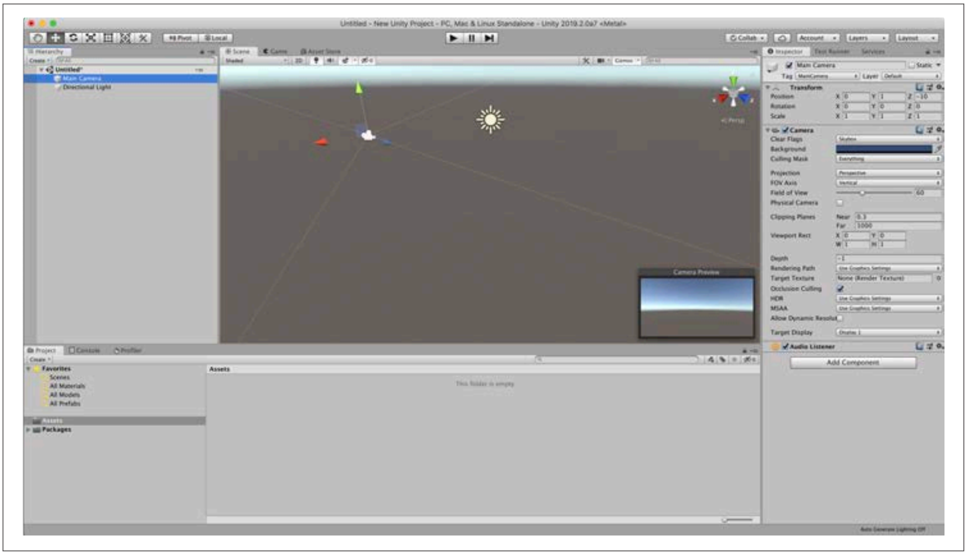


Figure 1-1. The main window of the Unity editor

When it's first launched, Unity uses the layout that you see in Figure 1-1. However, you can resize any window by clicking and dragging on its edges. You can also reposition the window by clicking and dragging its tab to somewhere else. If you drag a tab to the edge of another window, it will be placed at that side of the window. If you drag a tab onto the middle of the window, it will be added to the list of tabs at the top of the window.

There are several features that are of particular importance to anyone working in Unity. Let's take a look at each of them!

Toolbar

The toolbar contains controls that affect Unity as a whole (Figure 1-2). It's always at the top of the editor window, and can't be moved.

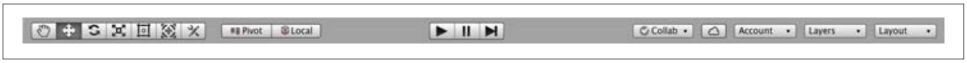


Figure 1-2. The toolbar

The toolbar, from left to right, contains the following controls:

Tools palette
: This palette controls the behavior of the transform controls that appear when an object is selected. Only one mode can be selected at a time; they are:

Hand tool
Clicking and dragging in the Scene view will move your view of the scene.

Move tool
Objects that are selected can be moved.

Rotate tool
Objects that are selected can be rotated around their pivot point or center.

Scale tool
Objects that are selected can be scaled around their pivot point or center.

Rectangle tool
Objects that are selected have a rectangle drawn around them, and can be scaled and repositioned. This tool is largely used for 2D objects like sprites and user interface elements.

Transform tool
This tool combines the Move, Rotate, and Scale tools. Selected objects can be moved, rotated, and scaled.

Custom tool
Any custom tools that are defined by the code in your game will appear in here.

Pivot/Center toggle
This toggle sets whether the transform controls are placed at the local pivot point of the object or in the center of the object's volume. (This can be different for certain 3D models; for example, models of human characters typically place the pivot point at the character's feet.)

Local/Global toggle
This toggle sets whether the Transform tool operates in global space or local space. For example, in local space, dragging the blue "forward" arrow of the Move tool moves an object forward based on its own orientation, while in global mode, dragging the blue "forward" arrow ignores the object's orientation.

Play button
This button starts Play mode, which enters your game. You can click the button again to end Play mode and return to editing.

> You can edit the scene while in Play mode, but any changes you make to the scene will be lost when you end the game. Don't forget to check if you're playing the game or not before doing a large amount of work!

Pause button

This button pauses the game. If you're in Play mode, the game will pause immediately. If you're not in Play mode, you can still click this button; if you then click the Play button, the game will pause immediately after the first frame.

Step button

This button advances one frame, while keeping the game paused.

> To be more specific, stepping forward "one frame" means that Unity will advance the game's clock by the *fixed timestep* and then re-render the game. By default, the fixed timestep is 0.02 seconds; you can configure this in the project's Time settings (from the Edit menu, choose Project Settings → Time).

Collab menu

This menu provides controls for working with Unity Collaborate, Unity's version control service.

> Unity Collaborate is outside the scope of this book, but the Unity manual (*https://bit.ly/2GMVmDv*) provides a good introduction.

Services button

This button opens the Services view, which allows you to work with Unity's web-based services like Cloud Build, Unity Analytics, and more. For more information, see Unity's Services page (*https://unity3d.com/services*).

Account button

This button allows you to configure your Unity account.

Layers button

With this button you can choose which layers are currently visible or selectable.

Layout button

This button allows you to save and restore a predefined layout of windows. This is useful when you have a layout that works well for tasks like animation, or level layout, and you want to switch between different tasks without having to fiddle with precise window placement.

Scene

The Scene view allows you to view, select, and modify the objects in a scene (Figure 1-3). In the Scene view, you can left-click on any object to select it; when an object is selected, you can move, rotate, or scale it by using the transform controls, depending on which tool you have selected in the toolbar (see Figure 1-4).

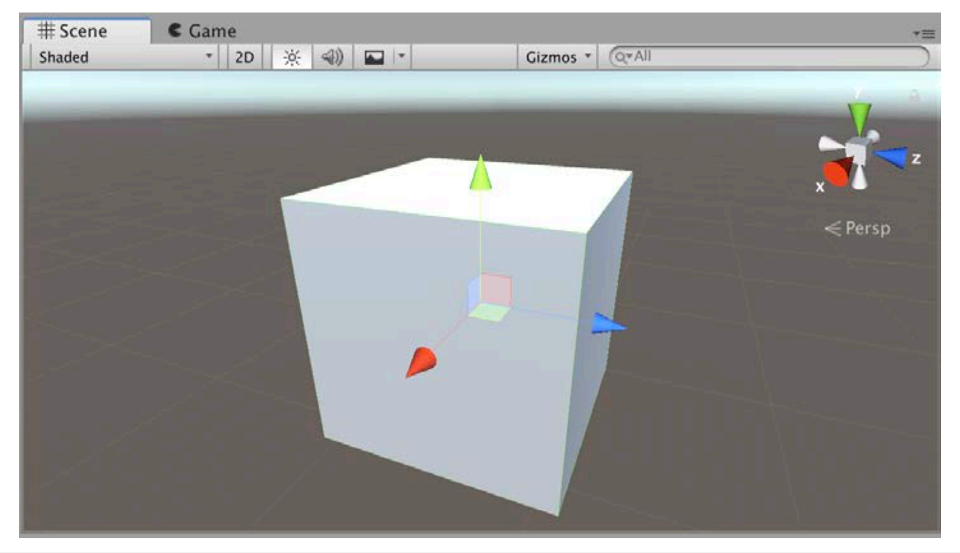


Figure 1-3. The Scene view

To use the transform controls on a selected object, click and drag the arrows (when using the Move tool), circles (the Rotate tool), or boxes (Scale tool) attached to it. Hold down the Shift key to snap the movement, rotate, and scale to a predefined increment. To move around the Scene view, select the Hand tool from the palette at the top left of the view, and click and drag. You can also hold the Alt key (Option on a Mac) to make clicking and dragging rotate the view; Alt-Control (Option-Command) and drag will pan the view around. You can use the scroll wheel on your mouse, or a two-finger gesture on a trackpad, to zoom in and out.

> You can also quickly move the Scene view to focus on the currently selected object. To do this, press the F key.

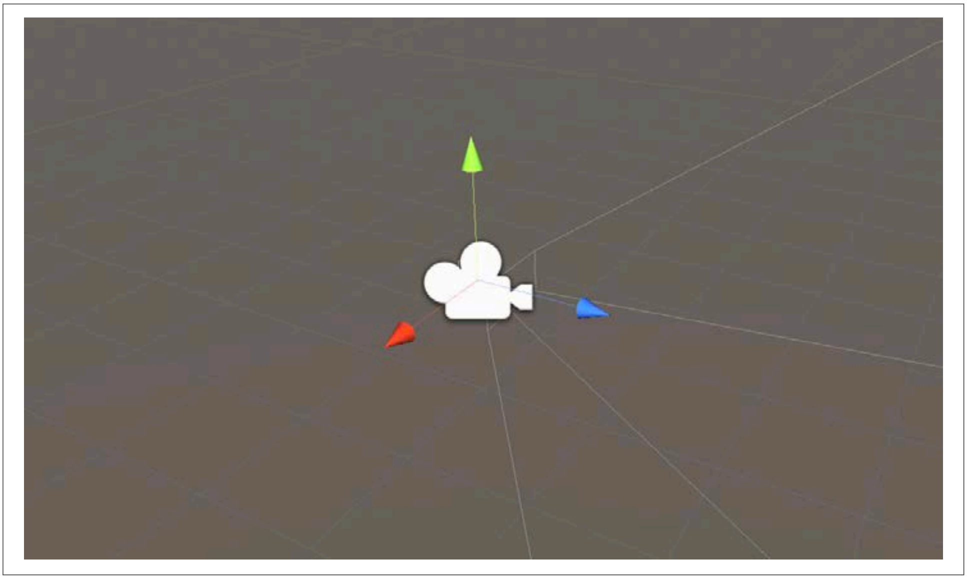


Figure 1-4. The Transform tool, shown with object selected. In this image a Camera has been selected, and the Transform tool is in Move mode; clicking and dragging the arrows will move the selected object in that direction.

Game

The Game view displays the view through the camera, and shows what the player would see if the game were running outside of Unity (Figure 1-5). The Game view itself isn't interactive unless the editor is in Play mode.

There are a few controls at the top of the Game view that let you control how the game is presented:

Display menu
Lets you control which display's contents to show in the view. (In Unity, a camera can be configured to send its results to the main screen, or to an external screen.)

Resolution menu
Lets you specify an aspect ratio or fixed resolution for the game display.

Scale slider
Lets you zoom in on the rendered view of the game.

Maximize On Play toggle
Button, when selected, makes the Game view fill the entire editor window when the game enters Play mode.

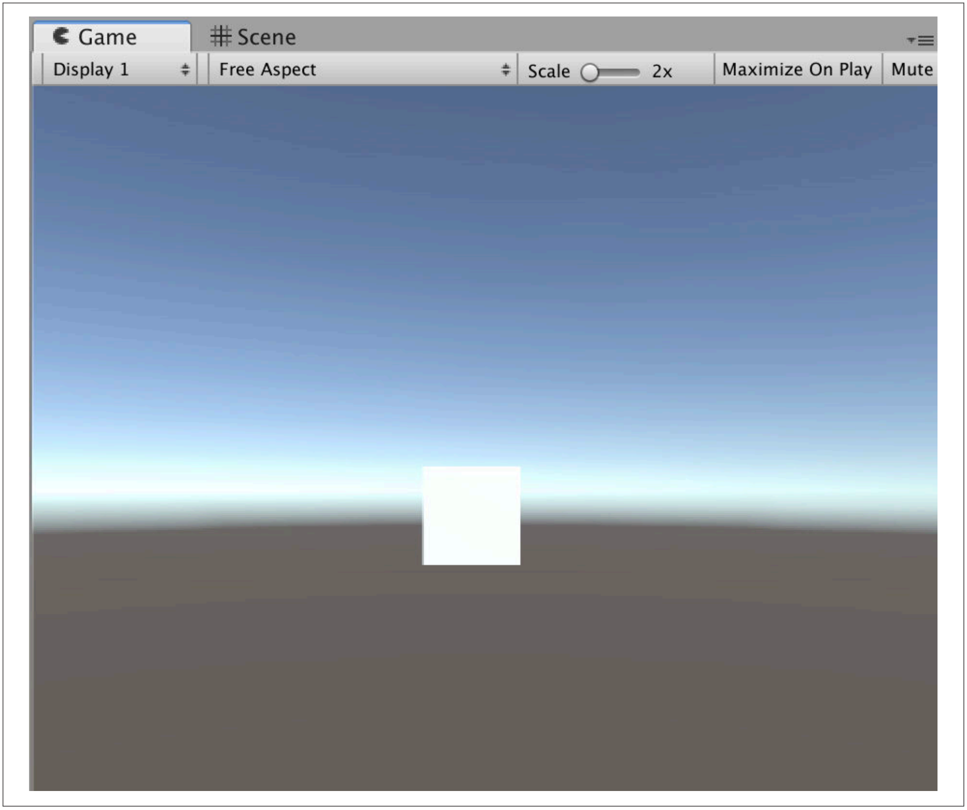


Figure 1-5. The Game view

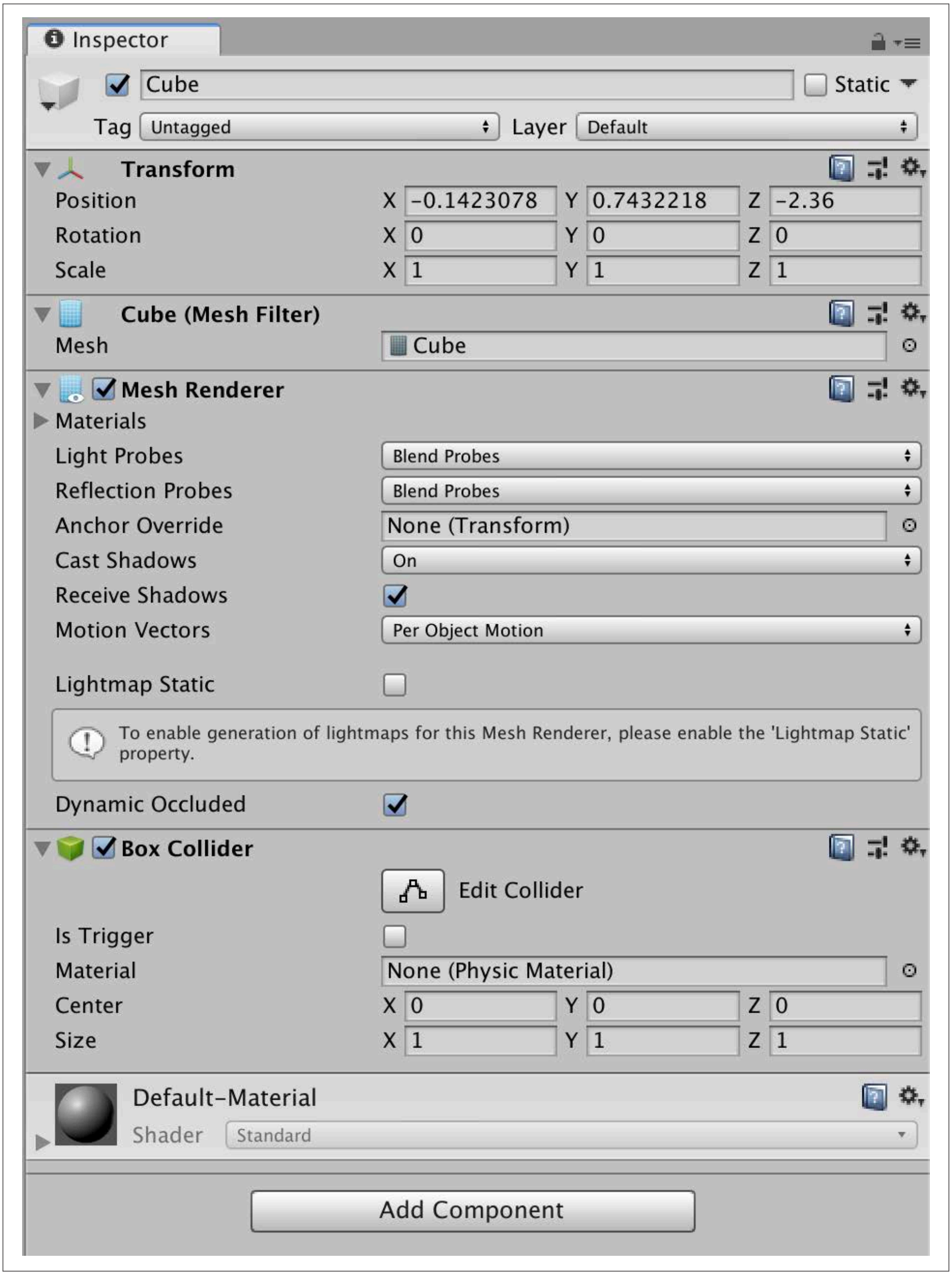
Mute Audio toggle
: Button disables all audio playback from the game. (This is useful when you don't want the game's audio to play over the top of your music while you're working, for example.)

Stats toggle
: Button controls whether performance statistics will be displayed in an overlay panel.

Gizmos button
: Lets you control whether the *gizmos*—icons that represent certain objects in the scene, like cameras—appear in the Game view, like they do in Scene view.

Inspector

The Inspector shows information about the objects currently selected (Figure 1-6). From here, you can control every *component* attached to the game object.



☑ Cube □ Static

Tag Untagged Layer Default

Transform

Position X -0.1423078 Y 0.7432218 Z -2.36

Rotation X 0 Y 0 Z 0

Scale X 1 Y 1 Z 1

Cube (Mesh Filter)

Mesh Cube

☑ Mesh Renderer

Materials

Light Probes Blend Probes

Reflection Probes Blend Probes

Anchor Override None (Transform)

Cast Shadows On

Receive Shadows ☑

Motion Vectors Per Object Motion

Lightmap Static □

To enable generation of lightmaps for this Mesh Renderer, please enable the 'Lightmap Static' property.

Dynamic Occluded ☑

☑ Box Collider

Is Trigger □

Material None (Physic Material)

Center X 0 Y 0 Z 0

Size X 1 Y 1 Z 1

Default-Material

Shader Standard

Figure 1-6. The Inspector

Components are a huge part of how Unity works, which means we'll discuss them in Recipe 1.3, instead of here.

At the top of the Inspector, you can set the name of the currently selected object. You can also set an icon for the object by clicking the icon to the left of the object's name and choosing a symbol to use. This is useful for game objects that are invisible.

By default, the Inspector will change its contents when the current selection changes. If you don't want it to do this, you can click the Lock icon at the top-right corner of the Inspector, and it will stay on the object that it's currently displaying.

Hierarchy

The Hierarchy shows the list of objects in the current scene (Figure 1-7). From here, you can browse through the list of objects in your scene. If you select an item in the Hierarchy, it becomes selected in the Scene view, and vice versa. You can also drag and drop one object on another to make it a child of that second object.

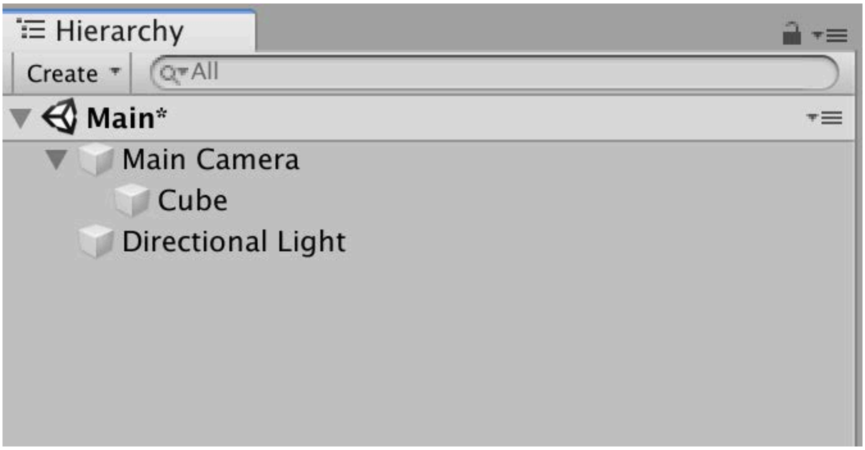


Figure 1-7. The Hierarchy view

Project

The Project view shows the contents of the project's *Assets* folder (Figure 1-8). Any files that are in the folder will be visible here; you can also move and rename files.

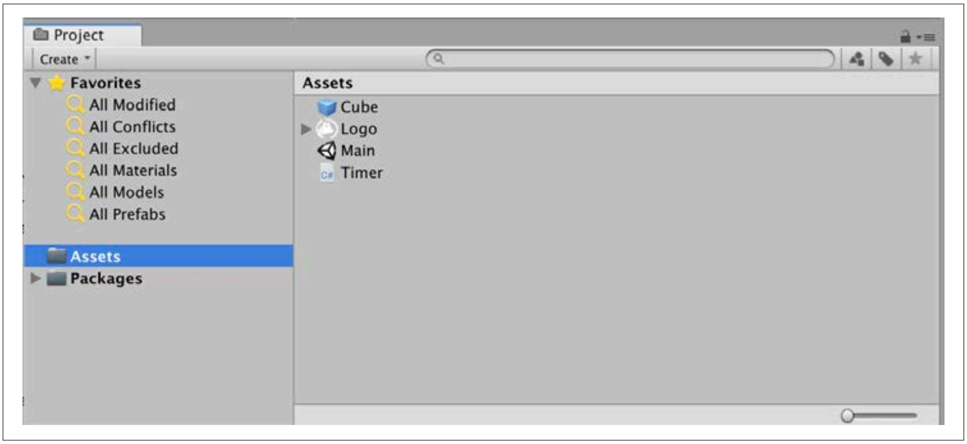


Figure 1-8. The Project view

When you rename a file, do it inside the Project view, and not outside of Unity. Unity tracks the additional metadata about your assets, such as import settings for textures, by creating a new file with the extension *.meta* next to each asset file. For example, an asset file called *Hero.png* would have a file called *Hero.png.meta* placed next to it. If you rename, move, or delete the asset file inside Unity, Unity will also update the *.meta* file, but if you rename, move, or delete the asset file outside of Unity, it won't know what to do, and it will have to recreate the *.meta* file from scratch (which means that any references to the file are lost, as well as any import settings).

Discussion

You can right-click the menu at the top bar of any window to close it. If you've closed a window and you want to get it back, you can usually find it in the Window menu. If you're really stuck, open the Window menu and choose Layouts → Revert Factory Settings.

1.2 Working with Game Objects

Problem

You want to create and modify game objects, which populate your game's scenes.

Solution

To create a new, empty game object, open the GameObject menu, and choose Create Empty. A new, empty game object will be added to the scene. It won't be visible in the scene, because it won't have any components that render anything. To learn more about components, see Recipe 1.3.

> You can also press Command-Shift-N (Control-Shift-N on a PC) to create a new, empty game object.

You can rename a game object by selecting it, and changing its name in the Inspector.

Game objects can be the *child* of other game objects. When a game object moves, rotates, or scales, its children are affected as well. This means that you can create hierarchies of game objects that work together as a system; for example, a car could have each of its four wheels as child objects, which means that they automatically stay in the correct position as the car moves around. To make one game object the child of another, drag and drop it onto another object in the Hierarchy (see "Hierarchy" on page 9).

You can also reorder an object by dragging and dropping it in the Hierarchy. As a shortcut, you can make an object move to the end of its siblings by pressing Command-Equals (Control-Equals on a PC), and move to the start of its siblings by pressing Command-Minus (Control-Minus on a PC).

Discussion

You can also create new game objects by dragging and dropping assets into the Scene view. For example, if you drag and drop a 3D model asset into the Scene view, Unity will create a new game object that contains the necessary components for rendering that model.

As a shortcut, you can quickly create an empty game object as a child of the currently selected object by opening the GameObject menu and choosing Create Empty Child. You can also press Option-Shift-N (Alt-Shift-N on a PC).

1.3 Working with Components

Problem

You want to add and modify components, which control the appearance and behavior of your game objects.

Solution

On its own, a game object is just an empty container. It's the components that make a game object actually do anything useful.

To get started in thinking about components, we'll create a new game object that comes with some useful components built in: we'll add a cube to the scene!

To do this, follow these steps:

1. Open the GameObject menu, and choose 3D Object → Cube. A new cube will be added to the scene (Figure 1-9).
2. Select the new cube in the Hierarchy or the Scene view. The Inspector will update to show the list of components attached to it (Figure 1-10).

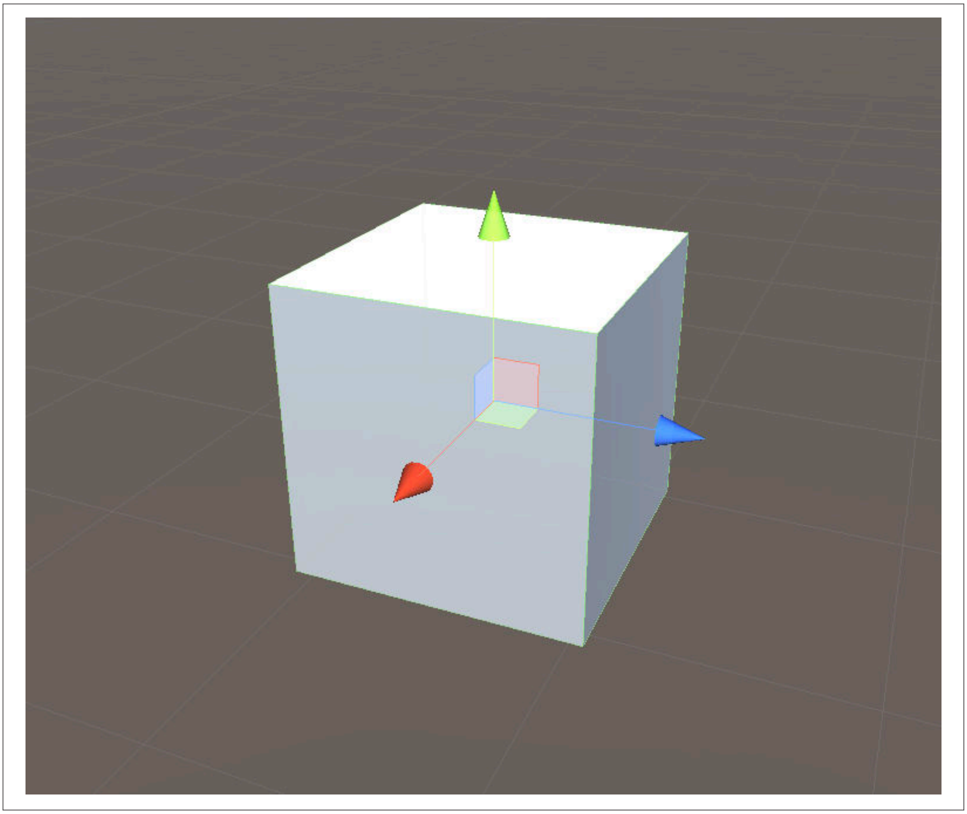
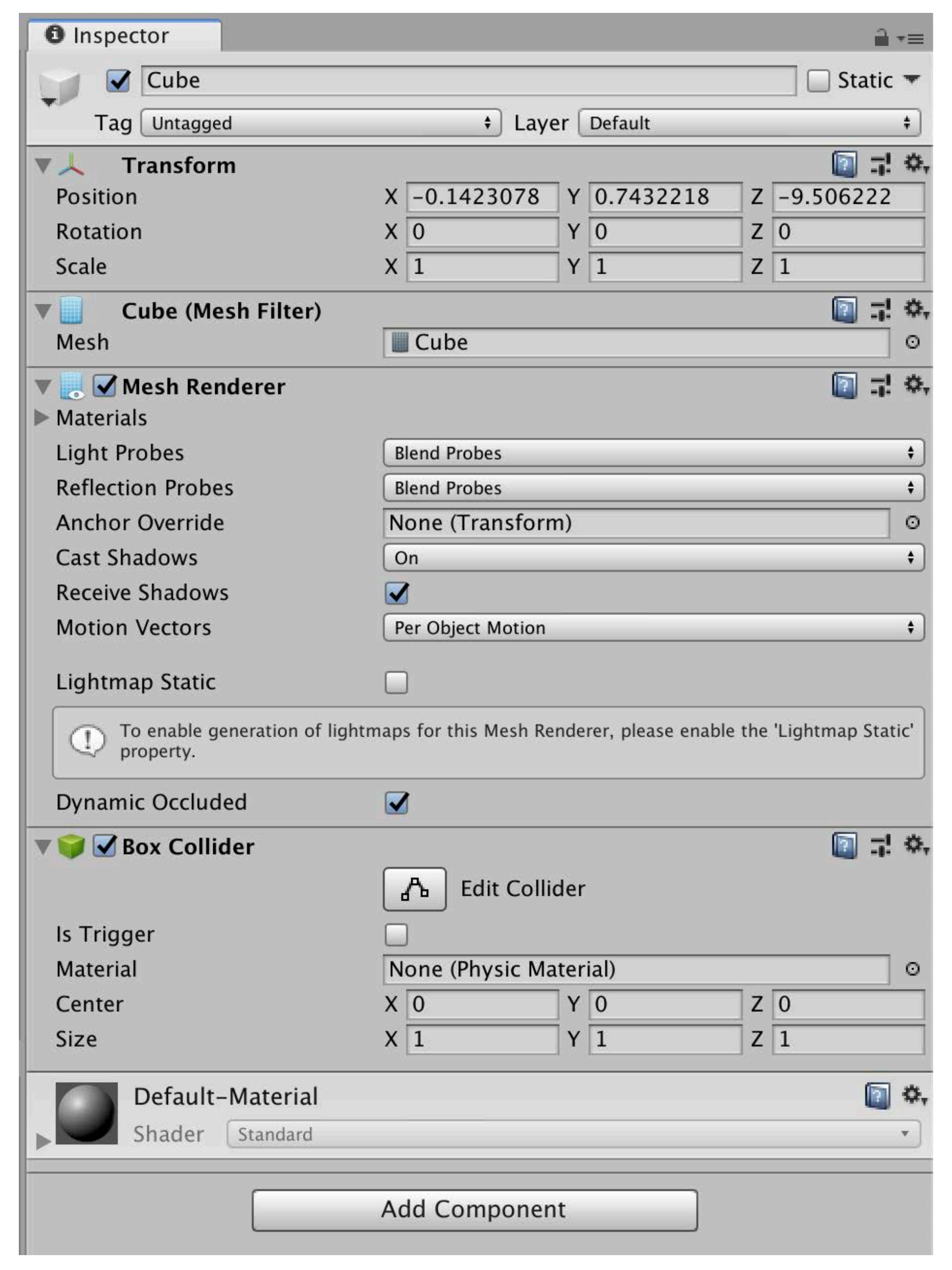


Figure 1-9. A new cube, freshly added to the scene



Inspector

☑ Cube □ Static

Tag Untagged Layer Default

Transform

Position	X	-0.1423078	Y	0.7432218	Z	-9.506222
Rotation	X	0	Y	0	Z	0
Scale	X	1	Y	1	Z	1

Cube (Mesh Filter)

Mesh Cube

☑ Mesh Renderer

Materials

Light Probes	Blend Probes
Reflection Probes	Blend Probes
Anchor Override	None (Transform)
Cast Shadows	On
Receive Shadows	☑
Motion Vectors	Per Object Motion
Lightmap Static	□

To enable generation of lightmaps for this Mesh Renderer, please enable the 'Lightmap Static' property.

Dynamic Occluded ☑

☑ Box Collider

Edit Collider

Is Trigger	□					
Material	None (Physic Material)					
Center	X	0	Y	0	Z	0
Size	X	1	Y	1	Z	1

Default-Material

Shader Standard

Add Component

Figure 1-10. The Inspector for the new cube

Every game object has at least one component: a Transform. The Transform component stores the position, rotation, and scale of an object, and is also responsible for keeping track of the parent of an object. You can't remove the Transform component.

On the cube, you'll find several additional components. Each of them do something different:

- The MeshFilter loads a mesh from a file on disk for the MeshRenderer to use. (For the cube object we're adding in this recipe, the asset is supplied by Unity; most of the game objects in your games will use assets you add to the project.)
- The MeshRenderer draws the mesh on the screen, using a Material asset to determine its appearance.
- The BoxCollider defines the physical shape of the object in the world.

Components let you configure how they work by exposing properties that are listed in the Inspector. For example, the MeshFilter component has a single property: the Mesh that it should be using. This is an example of an *object field*—it's a reference to another object that's part of your project. In the case of the MeshFilter, the field can take any Mesh object; these are found in the assets you add to your project. The type of object that an object field can use is determined by the component; you can't drop any other kind of object besides a Mesh into this field, for example.

Object fields don't have to refer to assets—they can also refer to other objects in the scene, too.

To add a component yourself, you can either use the Component menu or click the Add Component button at the bottom of the Inspector. Both of these options will let you specify what kind of component you want to add.

Discussion

You can remove a component by clicking on the Gear icon at the component's top-right corner, and choosing Remove Component.

To copy a component to another game object, click the Gear icon, and choose Copy Component. Next, go to the object you want to copy the component to, and click the Gear icon on any existing component (if it doesn't have any, use the Transform tool). Click Paste Component As New, and the component you copied will be pasted.

Scripts (which we discuss in Chapter 2) are components, too, and work in the exact same way as any other component you might attach to a game object.

1.4 Working with Prefabs

Problem

You want to store a game object in a file, so that you can reuse multiple copies of it.

Solution

Normally, game objects you add to your scenes are stored entirely within the scene. If you want to define an object ahead of time, and then make multiple copies of it, you can store it as a *prefab*. A prefab is an asset that stores a game object; you can *instantiate* the prefab, which creates a copy of it in your scenes.

To make a prefab, first create the original object in the scene. For example, create a new cube by opening the GameObject menu and choosing 3D Object → Cube. The object will appear in both the scene and the Hierarchy.

Next, drag and drop the object from the Hierarchy into the Project view. A new file will be created (Figure 1-11): this is the prefab! You'll also notice that the cube's entry in the Hierarchy view has turned blue, which indicates that it's an instance of a prefab. You can now safely delete the original cube from the scene.

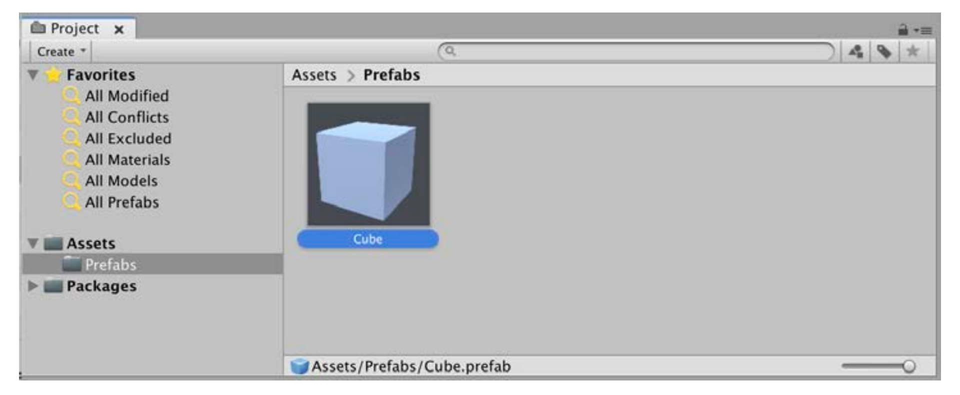
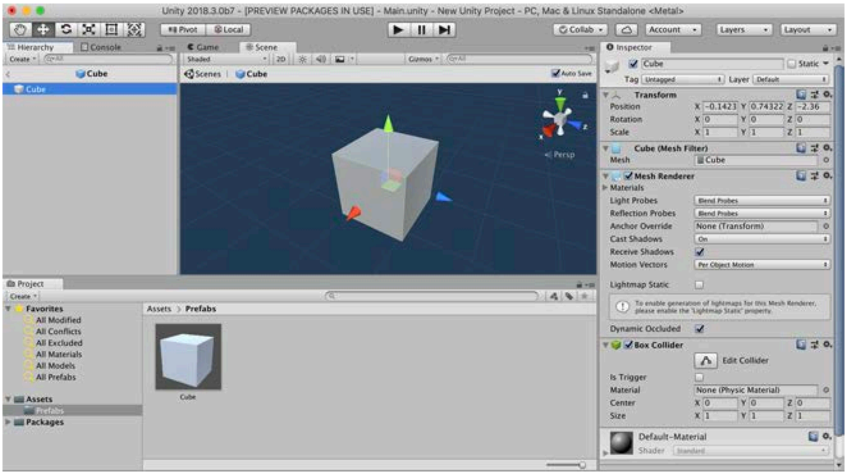


Figure 1-11. The new prefab, created from the cube

You can create an *instance* of a prefab by dragging and dropping the prefab into the scene. An instance is a copy of the game object and components stored in the prefab.

Edit a prefab by selecting the file in the Project view, clicking Open Prefab in the Inspector, and making changes to it (Figure 1-12).



☑ Cube □ Static
Tag Untagged Layer Default

Transform
Position X -0.1423 Y 0.74322 Z -2.36
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Cube (Mesh Filter)
Mesh Cube

☑ Mesh Renderer
Materials
Light Probes Blend Probes
Reflection Probes Blend Probes
Anchor Override None (Transform)
Cast Shadows On
Receive Shadows ☑
Motion Vectors Per Object Motion
Lightmap Static □
To enable generation of lightmaps for this Mesh Renderer, please enable the 'Lightmap Static' property.
Dynamic Occluded ☑

☑ Box Collider
Edit Collider
Is Trigger □
Material None (Physic Material)
Center X 0 Y 0 Z 0
Size X 1 Y 1 Z 1

Default-Material
Shader Standard

Figure 1-12. Editing the cube prefab

When you're done, click the back arrow button in the Hierarchy. All instances of the prefab across your entire project will be updated to include your changes.

Discussion

Instantiated copies of prefabs are linked to their original; if you make changes to the prefab, the changes you make apply to any instances of it. The reverse is not true by default: if you make changes to an instance, they won't apply to the original. For example, if you add an instance of the cube prefab to the scene and modify its `Scale` property, the changes will apply only to that instance. Additionally, the property that you change will be highlighted in bold, and with a blue line, to make it easier to see.

However, if you *do* want to apply the changes, right-click the property that you've changed, and click "Apply to Prefab." If you want to apply all of your changes, open the Overrides menu at the top of the Inspector and click Apply All.

> From the Overrides menu, you can also see a summary of the changes you've made to an instance.

1.5 Working with Scenes

Problem

You want to create and edit scenes, which are the containers for your game's objects.

Solution

When you create a new project, Unity will create a new, empty scene for you. When you press Command-S (Control-S on a PC), Unity will save the scene file to disk; if this is the first time that the scene has been saved, Unity will ask you where to save it. You can create more scenes by opening the File menu and choosing New Scene; don't forget to save your new scene to store it on disk.

Discussion

You can use scenes for a variety of purposes. For example, your main menu can be stored as a scene, as well as each of your game's levels. During gameplay, you can load new scenes via code. We'll discuss this in Recipe 2.8.

1.6 Managing Assets

Problem

You want to add assets to your project and configure how Unity imports them.

Solution

To add a file to your project, simply drag and drop it from the Finder (Explorer on a PC) into the Project view. Unity will import it and make it available to your project.

Once Unity has imported it, you can select the file and configure *how* Unity imports it by looking at the Inspector. Different file formats have different options; for example, you can configure an image to be imported as a sprite, in which case Unity will generate an additional sprite data asset for use in the sprite system (discussed in Chapter 5), or as one of multiple different types of textures.

Discussion

Unity supports a wide variety of file formats:

- 3D objects (Autodesk FBX, Collada): Unity can also import Maya, Cinema 4D, 3ds Max, Cheetah3D, Modo, Lightwave, Blender, and SketchUp files, if the corresponding software is installed.
- (Audio: WAV, MP3, Ogg, AIFF): Unity also supports a variety of tracker module formats—specifically, Impulse Tracker (*.it*), Scream Tracker (*.s3m*), Extended Module File Format (*.xm*), and Module File Format (*.mod*).
- 2D textures (Adobe Photoshop, BMP, PNG, JPG, BMP, TGA)
- Text (*.txt*, *.json*)

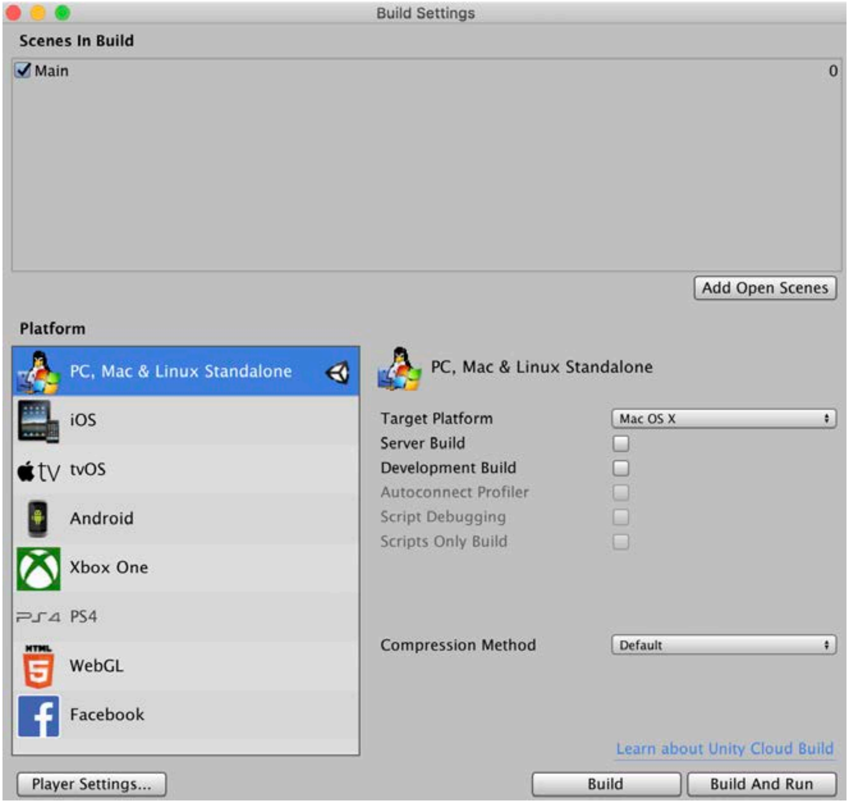
1.7 Building Unity Projects

Problem

You want to configure Unity to build your game, so that you can distribute the game to players.

Solution

To build your game, open the Build Settings view by opening the File menu and choosing Build Settings (Figure 1-13).



Build Settings

Scenes In Build

☑ Main 0

PC, Mac & Linux Standalone

Target Platform: Mac OS X

Server Build ☐

Development Build ☐

Autoconnect Profiler ☐

Script Debugging ☐

Scripts Only Build ☐

Compression Method: Default

Figure 1-13. The Build Settings view

When you build your game, you specify which scenes should be included. If you haven't already saved your current scene, you should do so now by opening the File menu, and choosing Save, or by pressing Command-S (Control-S on a PC). You can drag and drop the scenes you want to include into the Scenes In Build list, or you can click the Add Open Scenes button to add all scenes that you have open right now.

Next, you need to select which platform you want to build for. Unity supports a wide range of platforms, ranging from desktop PCs to mobile phones to consoles and more. Only one platform can be selected at a time; the current platform will be marked with a Unity logo next to it. To build for a different platform, select it and click the Switch Platform button.

When you're ready, click the Build button. Unity will ask you where you want to save the build; once that's done, Unity will start building.

Discussion

To build for certain platforms, you need to download the appropriate platform support module. If you don't have the necessary module, you won't be able to build; to get the module, click the platform and click the Open Download Page button. You'll be taken to the appropriate web page for downloading the module.

> Certain platforms, like consoles, require a special license; see the Unity Platform Installation page (*https://unity3d.com/platform-installation*) for more information.

1.8 Accessing Preferences

Problem

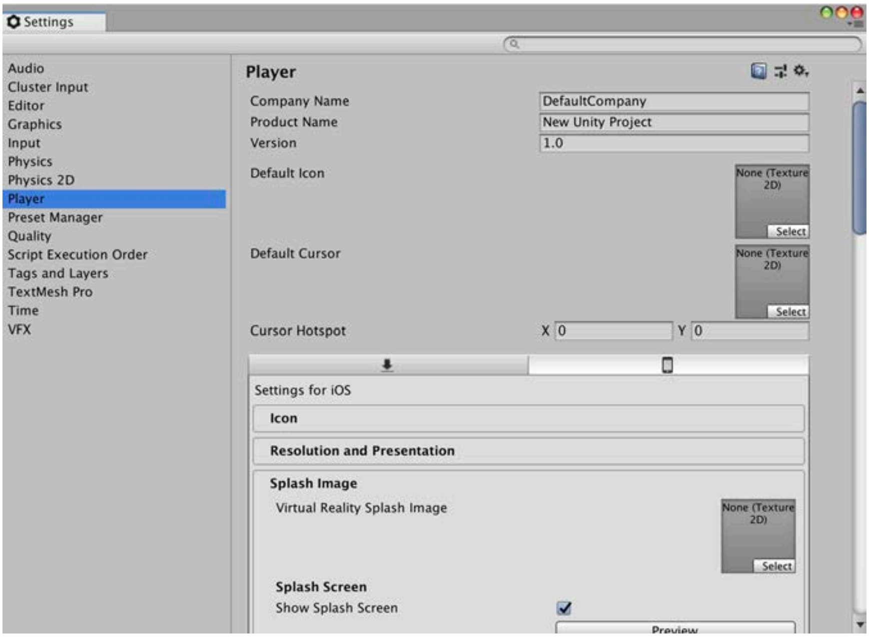
You want to access the settings that control Unity's behavior, both for the entire application and for your current project.

Solution

To open the per-project settings, open the Edit menu, and choose Project Settings. The Project Settings view will appear (Figure 1-14).

In the Project Settings view you can configure a wide range of possible settings for your game, including the maximum resolution, build name, graphical quality settings, input, and more.

For settings that affect the entire Unity application itself, access the Preferences view.



Settings

Audio
Cluster Input
Editor
Graphics
Input
Physics
Physics 2D
Player
Preset Manager
Quality
Script Execution Order
Tags and Layers
TextMesh Pro
Time
VFX

Player

Company Name: DefaultCompany
Product Name: New Unity Project
Version: 1.0
Default Icon: None (Texture 2D) Select
Default Cursor: None (Texture 2D) Select
Cursor Hotspot: X 0 Y 0

Settings for iOS

Icon

Resolution and Presentation

Splash Image

Virtual Reality Splash Image: None (Texture 2D) Select

Splash Screen

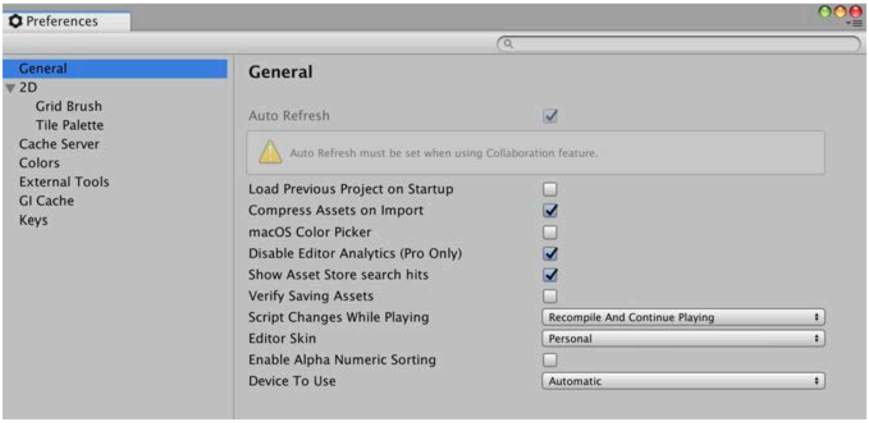
Show Splash Screen: ☑

Preview

Figure 1-14. The Project Settings view

Discussion

The Preferences view (Figure 1-15) allows you to configure things like colors, hotkeys, and other settings that apply to all projects. Access these settings on a PC by opening the Edit menu and choosing Preferences; on a Mac, open the Unity menu and choose Preferences.



Preferences

General
▼ 2D
Grid Brush
Tile Palette
Cache Server
Colors
External Tools
GI Cache
Keys

General

Auto Refresh: ☑

⚠ Auto Refresh must be set when using Collaboration feature.

Load Previous Project on Startup: □
Compress Assets on Import: ☑
macOS Color Picker: □
Disable Editor Analytics (Pro Only): ☑
Show Asset Store search hits: ☑
Verify Saving Assets: □
Script Changes While Playing: Recompile And Continue Playing
Editor Skin: Personal
Enable Alpha Numeric Sorting: □
Device To Use: Automatic

Figure 1-15. The Unity Preferences view

O'REILLY®

Unity Game Development Cookbook

Discover how to use the Unity game engine to its full potential for both 3D and 2D game development—from the basics of scripting to useful tricks in gameplay, behavior, and animation. With this problem-solving cookbook, you'll get started in two ways:

First, you'll learn about the Unity game engine through brief recipes that teach specific features of the software and scripting systems.

Second, you'll apply a collection of snippets to address common gameplay scenarios, such as properly keeping score.

Using our cookbook format, we pinpoint the problem, set out the solution, and discuss how to solve your problem in the best and most straightforward way possible. This book is ideal for beginning to intermediate Unity developers.

You'll find solutions for:

- **2D and 3D graphics**
- **Math, physics, and character control**
- **Animation and movement**
- **Behavior and AI**
- **Sound and music**
- **Input and gameplay**
- **Scripting and user interface**

Dr. Paris Buttfield-Addison is a game designer, computing researcher, and legal nerd, and the cofounder of game development studio Secret Lab.

Dr. Jon Manning is a software engineering expert in Swift, C#, and Objective-C. As cofounder of Secret Lab, he created the popular Yarn Spinner framework and helped on the BAFTA-winning game *Night in the Woods*.

Dr. Tim Nugent is a mobile app developer, game designer, and computing researcher.

"Want to understand Unity? The *Unity Game Development Cookbook* is concise, easily understood, and filled with delicious morsels of game-making truth."

—**Liam Esler**
Game developer for *Baldur's Gate II: Enhanced Edition*, *Icewind Dale: Enhanced Edition*, and *Pillars of Eternity*

"*Unity Game Development Cookbook* is an invaluable resource for Unity developers. Jon, Tim, and Paris leverage their considerable experience to present easy to follow solutions for a wide range of problems. Whether you're a complete beginner or an experienced developer, there's something in here for everyone."

—**Tom Brophy**
Lead Programmer, Mountains
(Creators of *Florence*)

DEVELOPMENT / MOBILE

ISBN : 978-93-5213-818-0

9 789352 138180

MRP: ₹ 1,375.00

Twitter: @oreillymedia
facebook.com/oreilly

SPD®

SHROFF PUBLISHERS & DISTRIBUTORS PVT. LTD.

First Edition/2019/Paperback/English