O'REILLY®

Fourth Edition

Head First C#

A Learner's Guide to Real-World Programming with C# and .NET Core

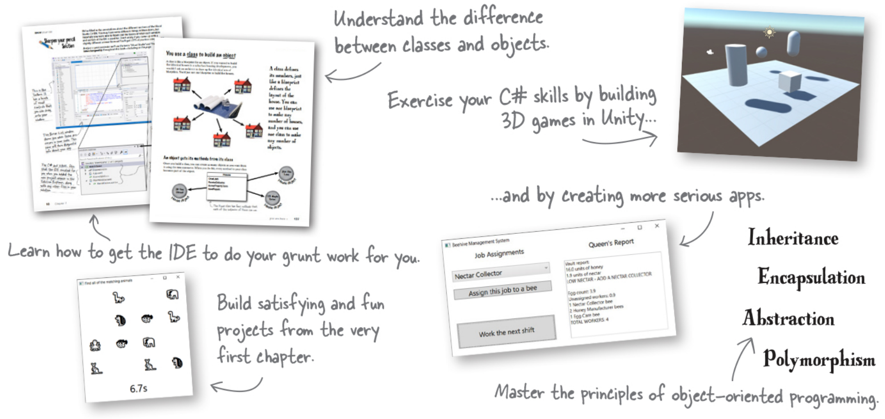
Andrew Stellman
& Jennifer Greene

A Brain-Friendly Guide

Head First C#

What will you learn from this book?

Dive into C# and create apps, user interfaces, games, and more using this fun and highly visual introduction to C#, .NET Core, and Visual Studio. With this completely updated guide, which covers C# 8.0 and Visual Studio 2019, beginning programmers like you will build a fully functional game in the opening chapter. Then you'll learn how to use classes and object-oriented programming, create 3D games in Unity, and query data with LINQ. And you'll do it all by solving puzzles, doing hands-on exercises, and building real-world applications. By the time you're done, you'll be a solid C# programmer—and you'll have a great time along the way!


Understand the difference between classes and objects.
Exercise your C# skills by building 3D games in Unity...
...and by creating more serious apps.
Learn how to get the IDE to do your grunt work for you.
Build satisfying and fun projects from the very first chapter.
Inheritance
Encapsulation
Abstraction
Polymorphism
Master the principles of object-oriented programming.
Queen's Report
Job Assignments
Nectar Collector
Assign this job to a bee
Work the next shift
6.7s

What's so special about this book?

Based on the latest research in cognitive science and learning theory, *Head First C#* uses a visually rich format to engage your mind rather than a text-heavy approach that puts you to sleep. Why waste your time struggling with new concepts? This multisensory learning experience is designed for the way your brain really works.

"Thank you so much! Your books have helped me to launch my career."

—*Ryan White*
Game Developer

"Andrew and Jennifer have written a concise, authoritative, and most of all, fun introduction to C# development."

—*Jon Galloway*
Senior Program Manager on the .NET Community Team at Microsoft

"If you want to learn C# in depth and have fun doing it, this is THE book for you."

—*Andy Parker*
Fledgling C# programmer

.NET

US $64.99 **CAN $85.99**
ISBN: 978-1-491-97670-8

O'REILLY®

Head First C#

Fourth Edition

WOULDN'T IT BE DREAMY IF THERE WAS A C# BOOK THAT WAS MORE FUN THAN MEMORIZING A DICTIONARY? IT'S PROBABLY NOTHING BUT A FANTASY...

Andrew Stellman
Jennifer Greene

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

Head First C#

Fourth Edition

by Andrew Stellman and Jennifer Greene

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

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

Series Creators:	Kathy Sierra, Bert Bates
Cover Designer:	Ellie Volckhausen
Brain Image on Spine:	Eric Freeman
Editors:	Nicole Taché, Amanda Quinn
Proofreader:	Rachel Head
Indexer:	Potomac Indexing, LLC
Illustrator:	Jose Marzan
Page Viewers:	Greta the miniature bull terrier and Samosa the Pomeranian

Printing History:

November 2007: First Edition.
May 2010: Second Edition.
August 2013: Third Edition.
December 2020: Fourth Edition

ISBN: 978-1-491-97670-8

[LSI] [2020-11-13]

Unity Lab #1

Explore C# with Unity

Welcome to your first **Head First C# Unity Lab**. Writing code is a skill, and like any other skill, getting better at it takes **practice and experimentation**. Unity will be a really valuable tool for that.

Unity is a cross-platform game development tool that you can use to make professional-quality games, simulations, and more. It's also a fun and satisfying way to get **practice with the C# tools and ideas** you'll learn throughout this book. We designed these short, targeted labs to **reinforce** the concepts and techniques you just learned to help you hone your C# skills.

These labs are optional, but valuable practice—**even if you aren't planning on using C# to build games**.

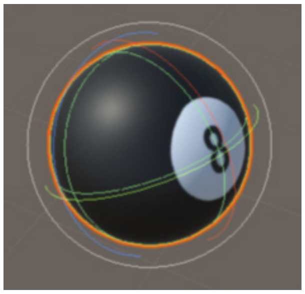
In this first lab, you'll get up and running with Unity. You'll get oriented with the Unity editor, and you'll start creating and manipulating 3D shapes.

Unity is a powerful tool for game design

Welcome to the world of Unity, a complete system for designing professional-quality games—both two-dimensional (2D) and three-dimensional (3D)—as well as simulations, tools, and projects. Unity includes many powerful things, including...

A cross-platform game engine

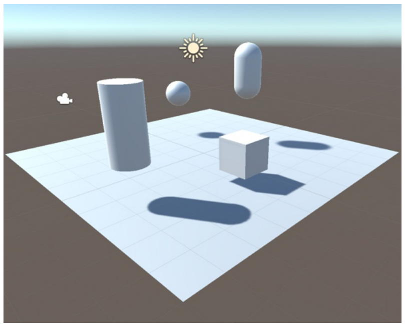
A **game engine** displays the graphics, keeps track of the 2D or 3D characters, detects when they hit each other, makes them act like real-world physical objects, and much, much more. Unity will do all of these things for the 3D games you build throughout this book.



A powerful 2D and 3D scene editor

You'll be spending a lot of time in the Unity editor. It lets you edit levels full of 2D or 3D objects, with tools that you can use to design complete worlds for your games. Unity games use C# to define their behavior, and the Unity editor integrates with Visual Studio to give you a seamless game development environment.

While these Unity Labs will concentrate on C# development in Unity, if you're a visual artist or designer, the Unity editor has many artist-friendly tools designed just for you. Check them out here: https://unity3d.com/unity/features/editor/art-and-design.



An ecosystem for game creation

Beyond being an enormously powerful tool for creating games, Unity also features an ecosystem to help you build and learn. The Learn Unity page (https://unity.com/learn) has valuable self-guided learning resources, and the Unity forums (https://forum.unity.com) help you connect with other game designers and ask questions. The Unity Asset Store (https://assetstore.unity.com) provides free and paid assets like characters, shapes, and effects that you can use in your Unity projects.

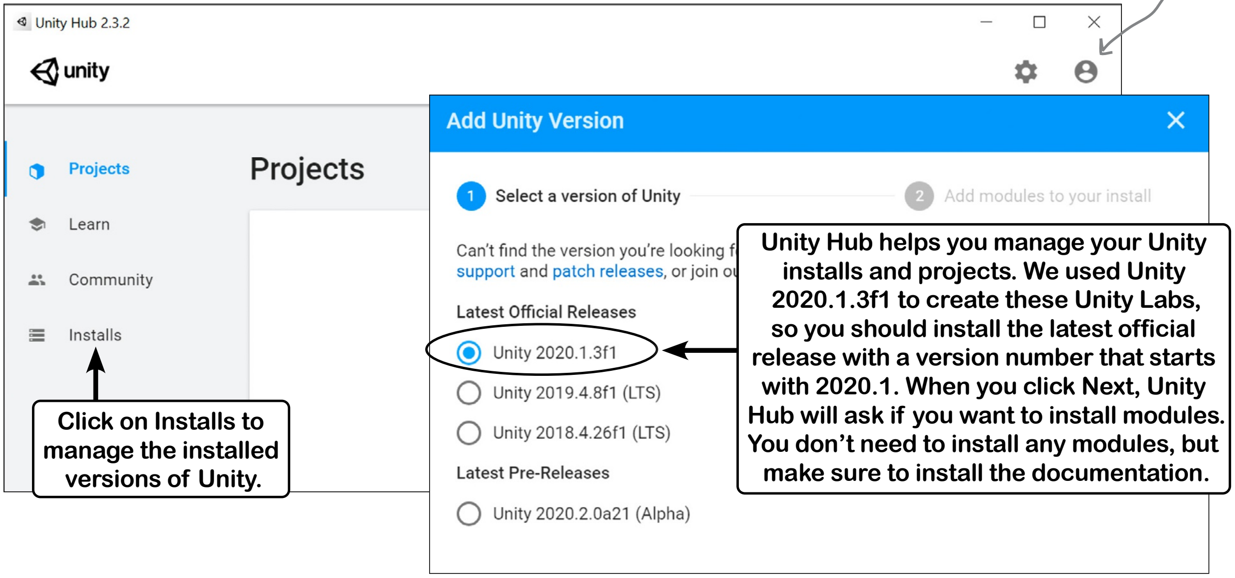
Our Unity Labs will focus on using Unity as a tool to explore C#, and practicing with the C# tools and ideas that you've learned throughout the book.

The *Head First C#* Unity Labs are laser-focused on a **developer-centric learning path**. The goal of these labs is to help you ramp up on Unity quickly, with the same focus on brain-friendly just-in-time learning you'll see throughout *Head First C#* to ***give you lots of targeted, effective practice with C# ideas and techniques***.

Download Unity Hub

All of the screenshots in this book were taken with the free Personal Edition of Unity. You'll need to enter your unity.com username and password into Unity Hub to activate your license.

Unity Hub is an application that helps you manage your Unity projects and your Unity installations, and it's the starting point for creating your new Unity project. Start by downloading Unity Hub from https://store.unity.com/download—then install it and run it.



Click on Installs to manage the installed versions of Unity.

Unity Hub helps you manage your Unity installs and projects. We used Unity 2020.1.3f1 to create these Unity Labs, so you should install the latest official release with a version number that starts with 2020.1. When you click Next, Unity Hub will ask if you want to install modules. You don't need to install any modules, but make sure to install the documentation.

Unity Hub lets you install multiple versions of Unity on the same computer, so you should install the same version that we used to build these labs. **Click Official Releases** and install the latest version that starts with ***Unity 2020.1***—that's the same version we used to take the screenshots in these labs. Once it's installed, make sure that it's set as the preferred version.

The Unity installer may prompt you to install a different version of Visual Studio. You can have multiple installations of Visual Studio on the same computer too, but if you already have one version of Visual Studio installed there's no need to make the Unity installer add another one.

You can learn more about installing Unity Hub on Windows, macOS, and Linux here: https://docs.unity3d.com/2020.1/Documentation/Manual/GettingStartedInstallingHub.html.

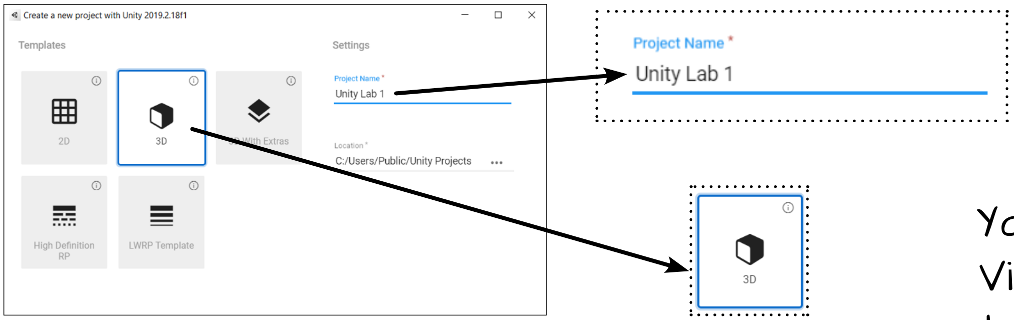
Unity Hub lets you have many Unity installs on the same computer. So even if there's a newer version of Unity available, you can use Unity Hub to install the version we used in the Unity Labs.

Unity Hub may look a little different.

*The screenshots in this book were taken with Unity 2020.1 (Personal Edition) and Unity Hub 2.3.2. You can use Unity Hub to install many different versions of Unity on the same computer, but you can only install the latest version of Unity Hub. The Unity development team is constantly improving Unity Hub and the Unity editor, so it's possible that what you see won't quite match what's shown on this page. We update these Unity Labs for newer printings of **Head First C#**. We'll add PDFs of updated labs to our GitHub page: https://github.com/head-first-csharp/fourth-edition.*

Use Unity Hub to create a new project

Click the NEW button on the Project page in Unity Hub to create a new Unity project. Name it ***Unity Lab 1***, make sure the 3D template is selected, and check that you're creating it in a sensible location (usually the Unity Projects folder underneath your home directory).


Create a new project with Unity 2019.2.18f1

Project Name *: Unity Lab 1

Location *: C:/Users/Public/Unity Projects

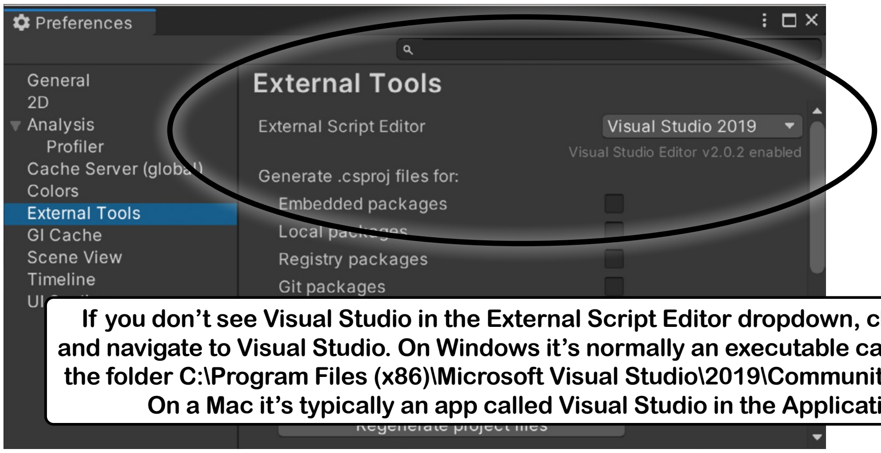
Click Create Project to create the new folder with the Unity project. When you create a new project, Unity generates a lot of files (just like Visual Studio does when it creates new projects for you). It could take Unity a minute or two to create all of the files for your new project.

You can use Visual Studio to debug the code in your Unity games. Just choose Visual Studio as the external script editor in Unity's preferences.

Make Visual Studio your Unity script editor

The Unity editor works hand-in-hand with the Visual Studio IDE to make it really easy to edit and debug the code for your games. So the first thing we'll do is make sure that Unity is hooked up to Visual Studio. **Choose Preferences from the Edit menu** (or from the Unity menu on a Mac) to open the Unity Preferences window. Click on External Tools on the left, and **choose Visual Studio** from the External Script Editor window.

In some older versions of Unity, you may see an ***Editor Attaching*** *checkbox—if so, make sure that it's checked (that will let you debug your Unity code in the IDE).*


External Tools

External Script Editor: Visual Studio 2019

Visual Studio Editor v2.0.2 enabled

Generate .csproj files for:

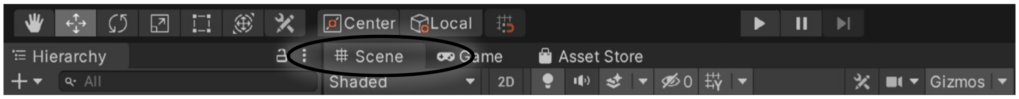
If you don't see Visual Studio in the External Script Editor dropdown, choose Browse... and navigate to Visual Studio. On Windows it's normally an executable called *devenv.exe* in the folder C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\Common7\IDE\. On a Mac it's typically an app called Visual Studio in the Applications folder.

OK! You're all ready to get started building your first Unity project.

Take control of the Unity layout

The Unity editor is like an IDE for all of the parts of your Unity project that aren't C#. You'll use it to work with scenes, edit 3D shapes, create materials, and so much more. Like in Visual Studio, the windows and panels in the Unity editor can be rearranged in many different layouts.

Find the Scene tab near the top of the window. Click on the tab and drag it to detach the window:

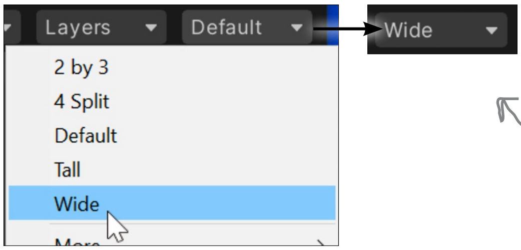


Try docking it inside or next to other panels, then drag it to the middle of the editor to make it a floating window.

The Scene view is your main interactive view of the world that you're creating. You use it to position 3D shapes, cameras, lights, and all of the other objects in your game.

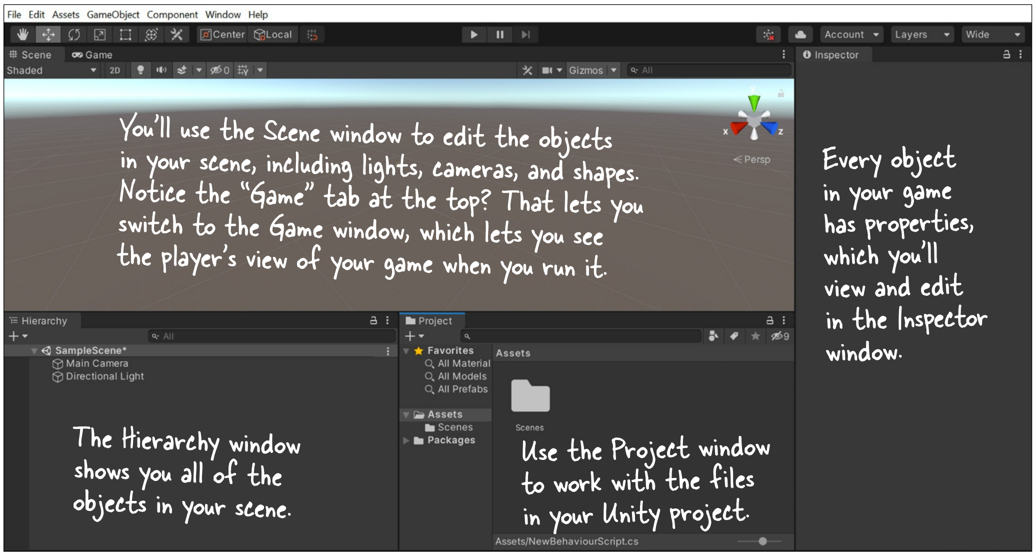
Choose the Wide layout to match our screenshots

We've chosen the Wide layout because it works well for the screenshots in these labs. Find the Layout dropdown and choose Wide so your Unity editor looks like ours.



Once you change the layout with the Layout dropdown on the right side of the toolbar, the dropdown may change its label to match the layout that you selected.

Here's what your Unity editor should look like in the Wide layout:



You'll use the Scene window to edit the objects in your scene, including lights, cameras, and shapes. Notice the "Game" tab at the top? That lets you switch to the Game window, which lets you see the player's view of your game when you run it.

Every object in your game has properties, which you'll view and edit in the Inspector window.

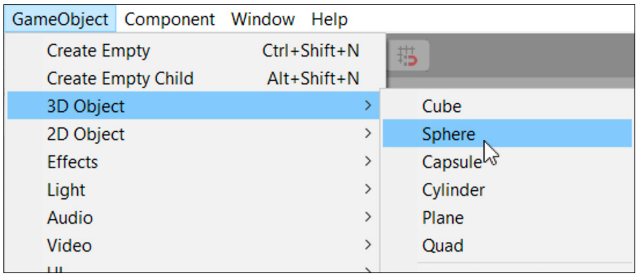
The Hierarchy window shows you all of the objects in your scene.

Use the Project window to work with the files in your Unity project.

Your scene is a 3D environment

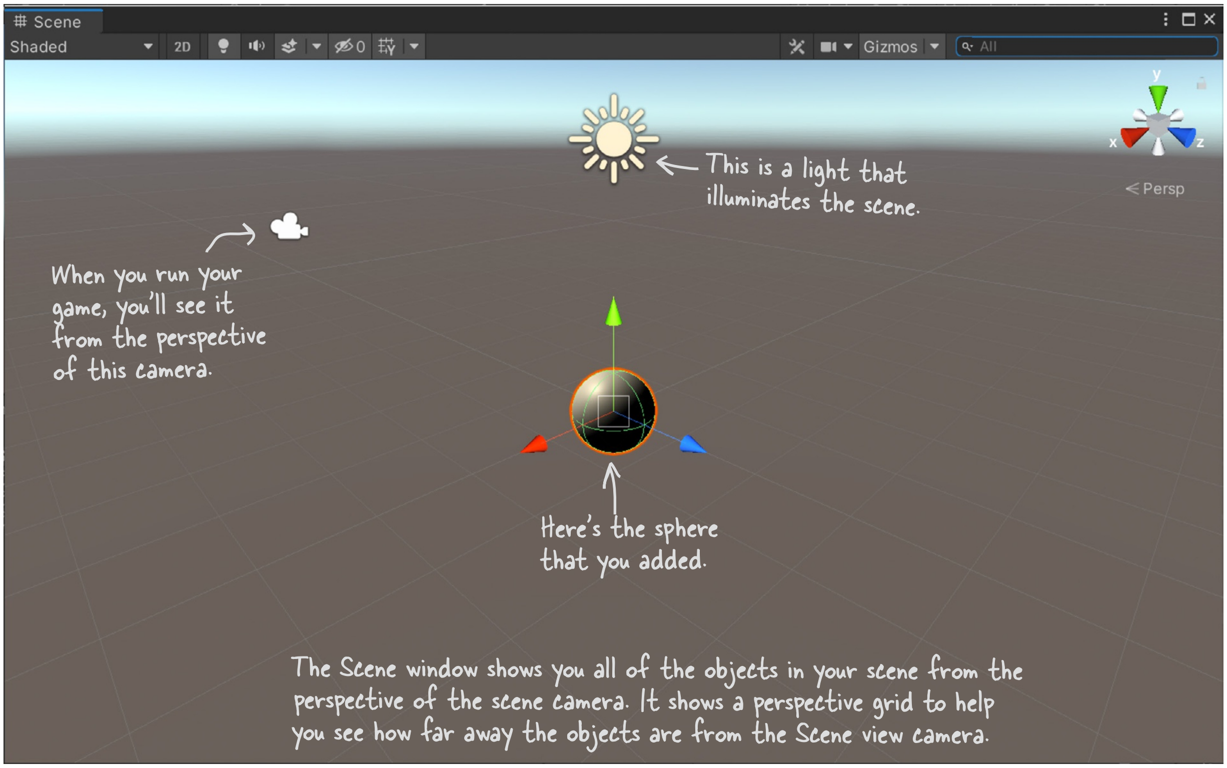
As soon as you start the editor, you're editing a **scene**. You can think of scenes as levels in your Unity games. Every game in Unity is made up of one or more scenes. Each scene contains a separate 3D environment, with its own set of lights, shapes, and other 3D objects. When you created your project, Unity added a scene called SampleScene, and stored it in a file called *SampleScene.unity*.

Add a sphere to your scene by choosing **GameObject >> 3D Object >> Sphere** from the menu:



These are called Unity's "primitive objects." We'll be using them a lot throughout these Untiy Labs.

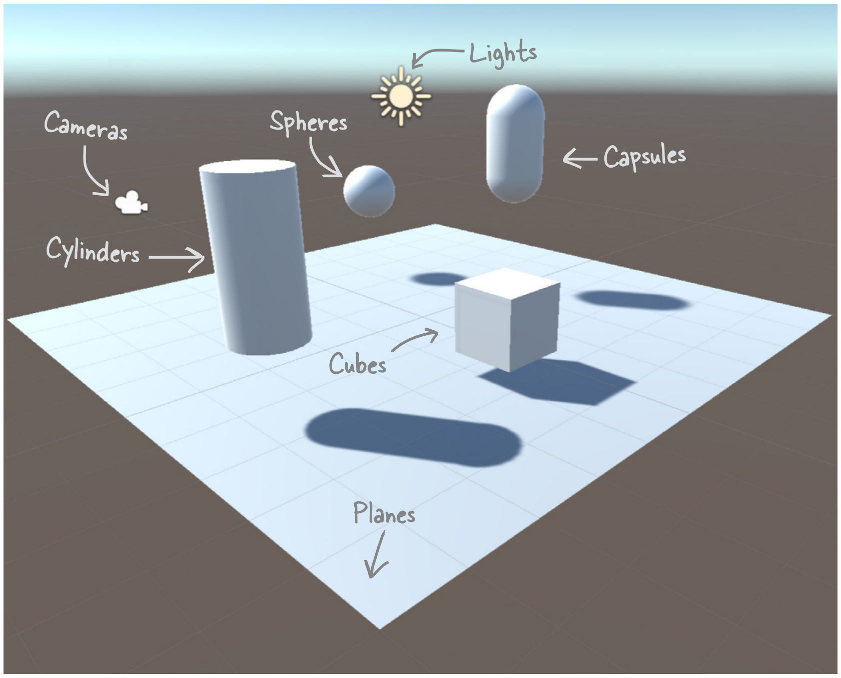
A sphere will appear in your Scene window. Everything you see in the Scene window is shown from the perspective of the **Scene view camera**, which "looks" at the scene and captures what it sees.


Scene
Shaded
2D
Gizmos
All
y
x
z
< Persp
This is a light that illuminates the scene.
When you run your game, you'll see it from the perspective of this camera.
Here's the sphere that you added.
The Scene window shows you all of the objects in your scene from the perspective of the scene camera. It shows a perspective grid to help you see how far away the objects are from the Scene view camera.

Unity games are made with GameObjects

When you added a sphere to your scene, you created a new **GameObject**. The GameObject is a fundamental concept in Unity. Every item, shape, character, light, camera, and special effect in your Unity game is a GameObject. Any scenery, characters, and props that you use in a game are represented by GameObjects.

In these Unity Labs, you'll build games out different kinds of GameObjects, including:



Each GameObject contains a number of **components** that provide its shape, set its position, and give it all of its behavior. For example:

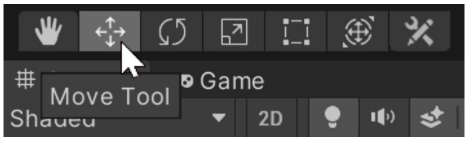
- *Transform components* determine the position and rotation of the GameObject.
- *Material components* change the way the GameObject is **rendered**—or how it's drawn by Unity—by changing the color, reflection, smoothness, and more.
- *Script components* use C# scripts to determine the GameObject's behavior.

ren-der, verb.
to represent or depict artistically.
Michelangelo ***rendered*** *his favorite model with more detail than he used in any of his other drawings.*

GameObjects are the fundamental objects in Unity, and components are the basic building blocks of their behavior. The Inspector window shows you details about each GameObject in your scene and its components.

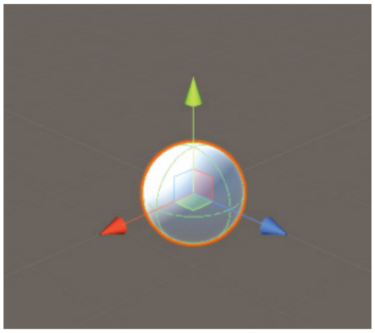
Use the Move Gizmo to move your GameObjects

The toolbar at the top of the Unity editor lets you choose Transform tools. If the Move tool isn't selected, press its button to select it.



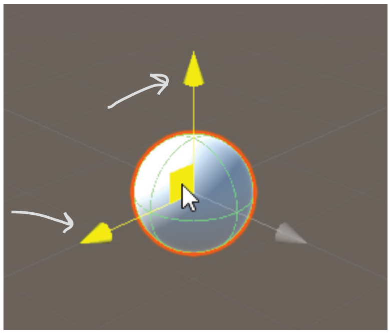
The buttons on the left side of the toolbar let you choose Transform Tools like the Move tool, which displays the Move Gizmo as arrows and a cube on top of the GameObject that's currently selected.

The Move tool lets you use the **Move Gizmo** to move GameObjects around the 3D space. You should see red, green, and blue arrows and a cube appear in the middle of the window. This is the Move Gizmo, which you can use to move the selected object around the scene.



Move your mouse cursor over the cube at the center of the Move Gizmo—notice how each of the faces of the cube lights up as you move your mouse cursor over it? Click on the upper-left face and drag the sphere around. You're moving the sphere in the X-Y plane.

When you click on the upper-left face of the cube in the middle of the Move Gizmo, its X and Y arrows light up and you can drag your sphere around the X-Y plane in your scene.



Move your sphere around the scene to get a feel for how the Move Gizmo works. Click and drag each of the three arrows to drag it along each plane individually. Try clicking on each of the faces of the cube in the Scene Gizmo to drag it around all three planes. Notice how the sphere gets smaller as it moves farther away from you—or really, the scene camera—and larger as it gets closer.

The Move Gizmo lets you move GameObjects along any axis or plane of the 3D space in your scene.

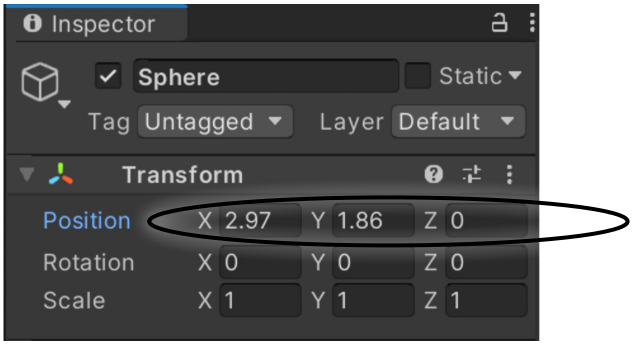
The Inspector shows your GameObject's components

As you move your sphere around the 3D space, watch the **Inspector window**, which is on the right side of the Unity editor if you're using the Wide layout. Look through the Inspector window—you'll see that your sphere has four components labeled Transform, Sphere (Mesh Filter), Mesh Renderer, and Sphere Collider.

Every GameObject has a set of components that provide the basic building blocks of its behavior, and every GameObject has a **Transform component** that drives its location, rotation, and scale.

You can see the Transform component in action as you use the Move Gizmo to drag the sphere around the X-Y plane. Watch the X and Y numbers in the Position row of the Transform component change as the sphere moves.

If you accidentally deselect a GameObject, just click on it again. If it's not visible in the scene, you can select it in the Hierarchy window, which shows all of the GameObjects in the scene. When you reset the layout to Wide, the Hierarchy window is in the lower-left corner of the Unity editor.



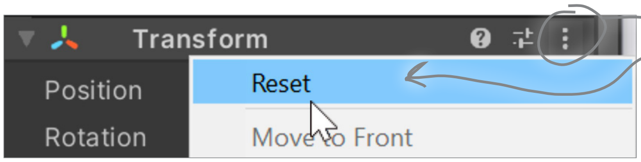
Inspector
Sphere — Static
Tag Untagged — Layer Default
Transform
Position X 2.97 Y 1.86 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Did you notice the grid in your 3D space? As you're dragging the sphere around, hold down the Control key. That causes the GameObject that you're moving to snap to the grid. You'll see the numbers in the Transform component move by whole numbers instead of small decimal increments.

Try clicking on each of the other two faces of the Move Gizmo cube and dragging to move the sphere in the X-Z and Y-Z planes. Then click on the red, green, and blue arrows and drag the sphere along just the X, Y, or Z axis. You'll see the X, Y, and Z values in the Transform component change as you move the sphere.

Now **hold down Shift** to turn the cube in the middle of the Gizmo into a square. Click and drag on that square to move the sphere in the plane that's parallel to the Scene view camera.

Once you're done experimenting with the Move Gizmo, use the sphere's Transform component context menu to reset the component to its default values. Click the **context menu button** (⋮) at the top of the Transform panel and choose Reset from the menu.



Use the context menu to reset a component. You can either click the three dots or right-click anywhere in the top line of the Transform panel in the Inspector window to bring up the context menu.

The position will reset back to [0, 0, 0].

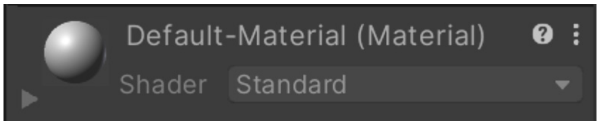
You can learn more about the tools and how to use them to position GameObjects in the Unity Manual. Click Help >> Unity Manual and search for the "Positioning GameObjects" page.

Save your scene often! Use File >> Save or Ctrl+S / ⌘S to save the scene right now.

Add a material to your Sphere GameObject

Unity uses **materials** to provide color, patterns, textures, and other visual effects. Your sphere looks pretty boring right now because it just has the default material, which causes the 3D object to be rendered in a plain, off-white color. Let's make it look like a billiard ball.

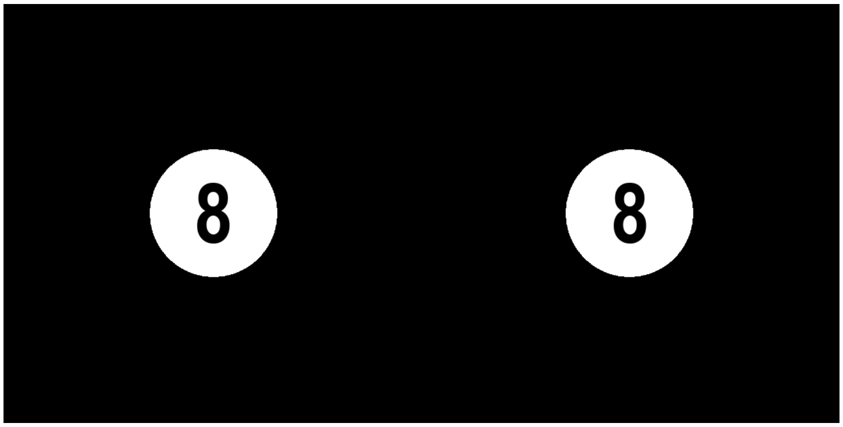
① **Select the sphere.**
When the sphere is selected, you can see its material as a component in the Inspector window:


Default-Material (Material)
Shader Standard

We'll make your sphere more interesting by adding a **texture**—that's just a simple image file that's wrapped around a 3D shape, almost like you printed the picture on a rubber sheet and stretched it around your object.

② **Go to our Billiard Ball Textures page on GitHub.**
Go to https://github.com/head-first-csharp/fourth-edition and click on the *Billiard Ball Textures* link to browse a folder of texture files for a complete set of billiard balls.

③ **Download the texture for the 8 ball.**
Click on the file *8 Ball Texture.png* to view the texture for an 8 ball. It's an ordinary 1200 × 600 PNG image file that you can open in your favorite image viewer.

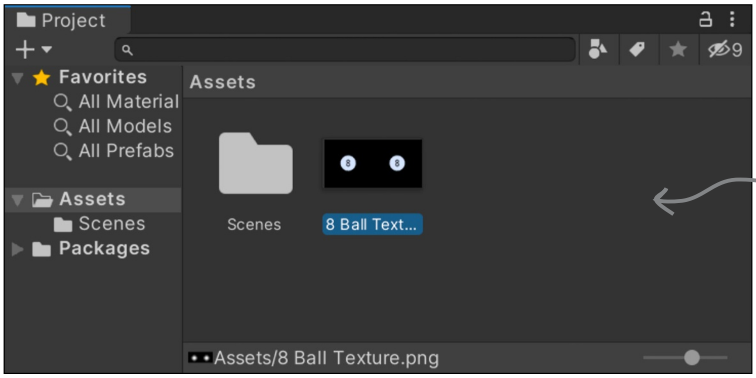


We designed this image file so that it looks like an 8 ball when Unity "wraps" it around a sphere.

Download the file into a folder on your computer.

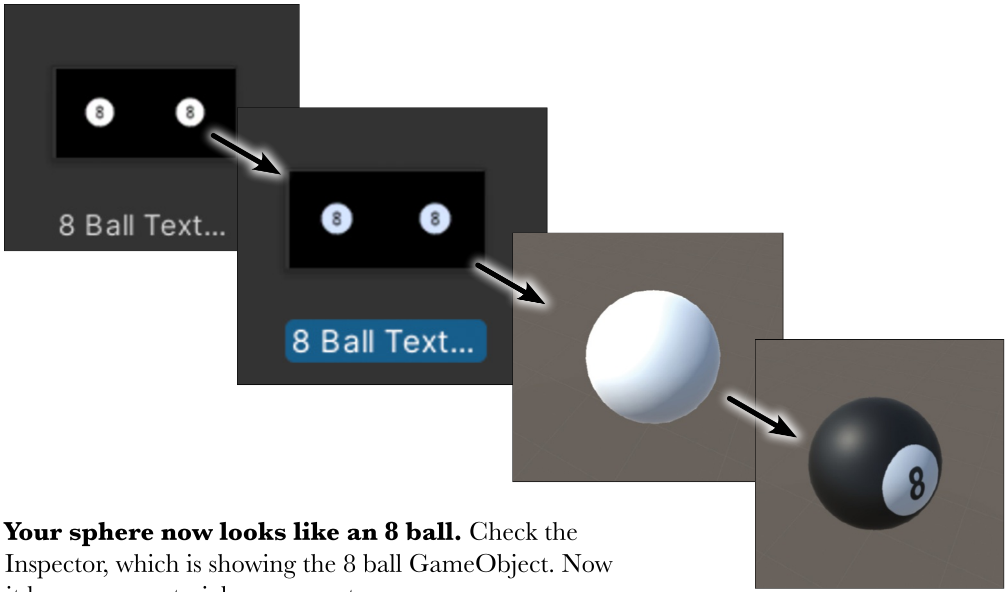
(You might need to right-click on the Download button to save the file, or click Download to open it and then save it, depending on your browser.)

④ **Import the 8 Ball Texture image into your Unity project.**
Right-click on the Assets folder in the Project window, choose **Import New Asset...** and import the texture file. You should now see it when you click on the Assets folder in the Project window.

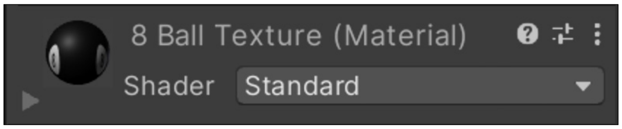


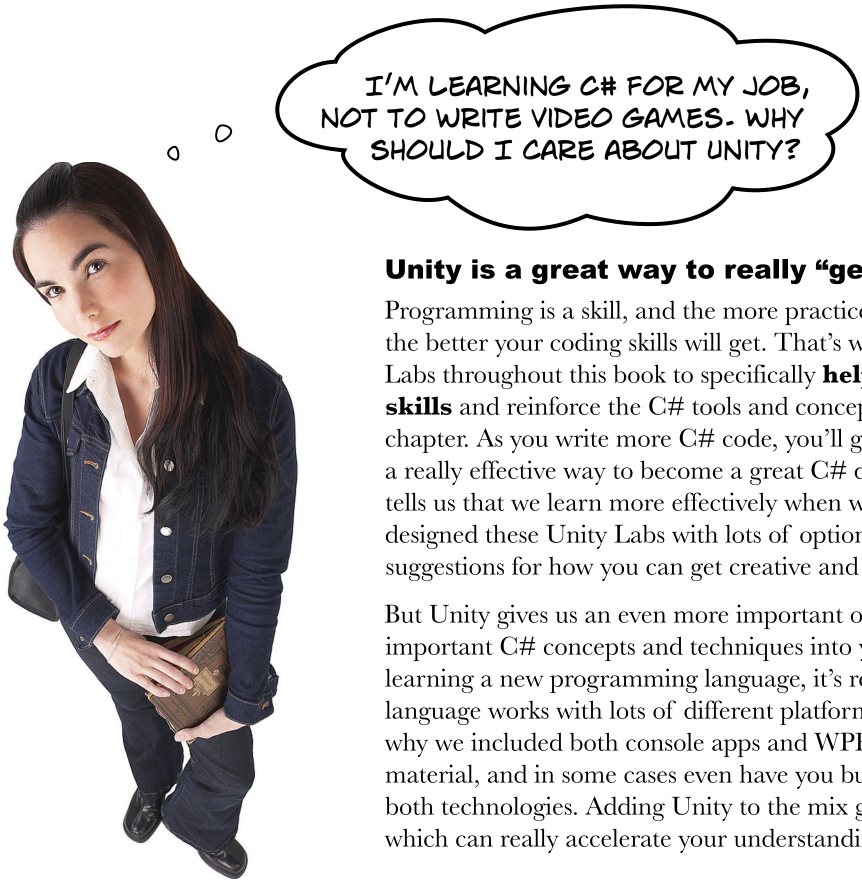
You right-clicked inside the Assets folder in the Project window to import the new asset, so Unity imported the texture into that folder.

⑤ **Add the texture to your sphere.**
Now you just need to take that texture and "wrap" it around your sphere. Click on 8 Ball Texture in the Project window to select it. Once it's selected, **drag it onto your sphere**.



Your sphere now looks like an 8 ball. Check the Inspector, which is showing the 8 ball GameObject. Now it has a new material component:


8 Ball Texture (Material)
Shader Standard


I'M LEARNING C# FOR MY JOB, NOT TO WRITE VIDEO GAMES. WHY SHOULD I CARE ABOUT UNITY?

Unity is a great way to really "get" C#.

Programming is a skill, and the more practice you get writing C# code, the better your coding skills will get. That's why we designed the Unity Labs throughout this book to specifically **help you practice your C# skills** and reinforce the C# tools and concepts that you learn in each chapter. As you write more C# code, you'll get better at it, and that's a really effective way to become a great C# developer. Neuroscience tells us that we learn more effectively when we experiment, so we designed these Unity Labs with lots of options for experimentation, and suggestions for how you can get creative and keep going with each lab.

But Unity gives us an even more important opportunity to help get important C# concepts and techniques into your brain. When you're learning a new programming language, it's really helpful to see how that language works with lots of different platforms and technologies. That's why we included both console apps and WPF apps in the main chapter material, and in some cases even have you build the same project using both technologies. Adding Unity to the mix gives you a third perspective, which can really accelerate your understanding of C#.

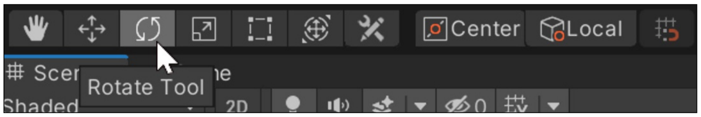
The **GitHub for Unity extension** (https://unity.github.com) lets you save your Unity projects in GitHub. Here's how:

- **To install GitHub for Unity:** Go to https://assetstore.unity.com and add GitHub for Unity to your assets. Go back to Unity, **choose Package Manager** from the Window menu, select "GitHub for Unity" from "My Assets," and import it. You'll need to import GitHub into each new Unity project.
- **To push your changes to a GitHub repo:** Choose GitHub from the Window menu. Each Unity project is stored in a separate repository in your GitHub account, so **click the Initialize button** to initialize a new *local* repo (you'll be prompted to log into GitHub), then **click the Publish button** to create a new repo in your GitHub account for your project. Any time you want to push your changes to GitHub, **go to the Changes tab** in the GitHub window, **click All**, enter a **commit summary** (any text will do), and **click Commit at** the bottom of the GitHub window. Then click **Push (1)** at the top of the GitHub window to push your changes back to GitHub.

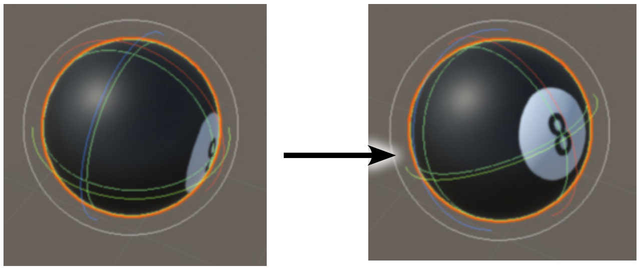
You can also back up and share your Unity projects with **Unity Collaborate**, which lets you publish your projects to their cloud storage. Your Unity Personal account comes with 1 GB of cloud storage for free, which is enough for all of the Unity Lab projects in this book. Unity will even keep track of your project history (which doesn't count against your storage limit). To publish your project, click the **Collab (** ✓ Collab ▾ **) button** on the toolbar, then click Publish. Use the same button to publish any updates. To see your published projects, log into https://unity3d.com and use the account icon to view your account, then click the Projects link from your account overview page to see your projects.

Rotate your sphere

Click the **Rotate tool** in the toolbar. You can use the Q, W, E, R, T, and Y keys to quickly switch between the Transform tools—press E and W to toggle between the Rotate tool and Move tool.



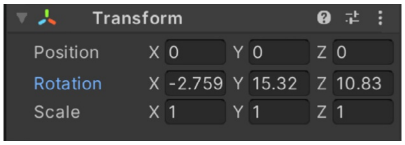
❶ **Click on the sphere.** Unity will display a wireframe sphere Rotate Gizmo with red, blue, and green circles. Click the red circle and drag it to rotate the sphere around the X axis.



It's easy to reset your windows and scene camera.

If you change your Scene view so you can't see your sphere anymore, or if you drag your windows out of position, just use the layout dropdown in the upper-right corner to **reset the Unity editor to the Wide layout**. It will reset the window layout and move the Scene view camera back to its default position.

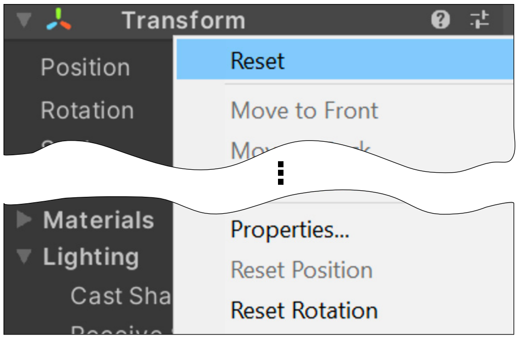
❷ **Click and drag the green and blue circles to rotate around the Y and Z axes.** The outer white circle rotates the sphere along the axis coming out of the Scene view camera. Watch the Rotation numbers change in the Inspector window.



Transform

Position	X 0	Y 0	Z 0
Rotation	X -2.759	Y 15.32	Z 10.83
Scale	X 1	Y 1	Z 1

❸ **Open the context menu of the Transform panel in the Inspector window.** Click Reset, just like you did before. It will reset everything in the Transform component back to default values—in this case, it will change your sphere's rotation back to [0, 0, 0].



Click the three dots (or right-click anywhere in the header of the Transform panel) to bring up the context menu. The Reset option at the top of the menu resets the component to its default values.

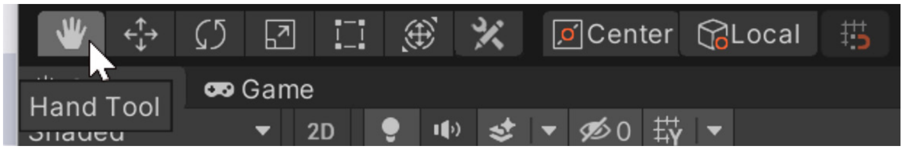
Use these options from further down in the context menu to reset the position and rotation of a GameObject.

Use File >> Save or Ctrl+S / ⌘S to save the scene right now. Save early, save often!

Move the Scene view camera with the Hand tool and Scene Gizmo

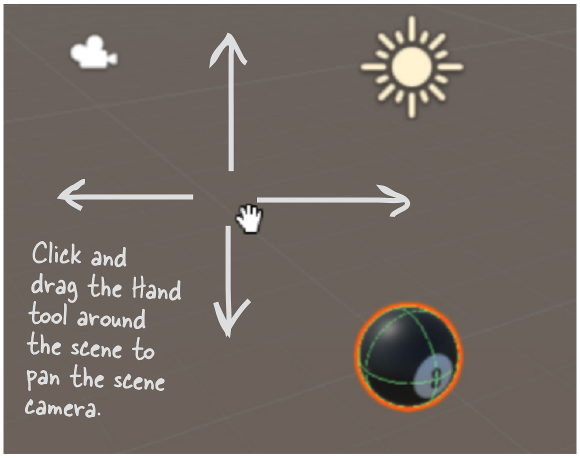
Use the mouse scroll wheel or scroll feature on your trackpad to zoom in and out, and toggle between the Move and Rotate Gizmos. Notice that the sphere changes size, but the Gizmos don't. The Scene window in the editor shows you the view from a virtual **camera**, and the scroll feature zooms that camera in and out.

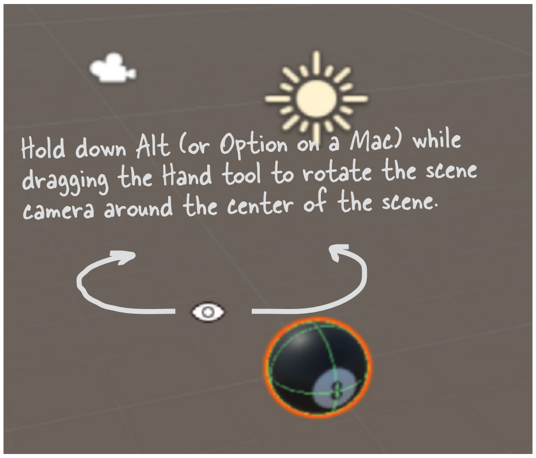
Press Q to select the **Hand tool**, or choose it from the toolbar. Your cursor will change to a hand.



Hold down Alt (or Option on a Mac) while dragging and the Hand tool turns into an eye and rotates the view around the center of the window

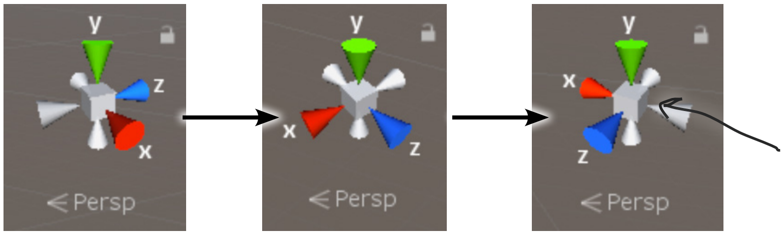
The Hand tool pans around the scene by changing the position and rotation of the scene camera. When the Hand tool is selected, you can click anywhere in the scene to pan.


Click and drag the Hand tool around the scene to pan the scene camera.


Hold down Alt (or Option on a Mac) while dragging the Hand tool to rotate the scene camera around the center of the scene.

When the Hand tool is selected, you can ***pan*** the scene camera by **clicking and dragging**, and you can ***rotate*** it by holding **down Alt (or Option) and dragging**. Use the **mouse scroll wheel** to zoom. Holding down the **right mouse button** lets you ***fly through the scene*** using the W-A-S-D keys.

When you rotate the scene camera, keep an eye on the **Scene Gizmo** in the upper-right corner of the Scene window. The Scene Gizmo always displays the camera's orientation—check it out as you use the Hand tool to move the Scene view camera. Click on the X, Y, and Z cones to snap the camera to an axis.



Click any of the cones in the Scene Gizmo to snap the camera to an axis. Drag them around to rotate the camera.

The Unity Manual has great tips on navigating scenes: https://docs.unity3d.com/Manual/SceneViewNavigation.html.

there are no Dumb Questions

Q: I'm still not clear on exactly what a component is. What does it do, and how is it different from a GameObject?

A: A GameObject doesn't actually do much on its own. All a GameObject really does is serve as a *container* for components. When you used the GameObject menu to add a Sphere to your scene, Unity created a new GameObject and added all of the components that make up a sphere, including a Transform component to give it position, rotation, and scale, a default Material to give it its plain white color, and a few other components to give it its shape, and help your game figure out when it bumps into other objects. These components are what make it a sphere.

Q: So does that mean I can just add any component to a GameObject and it gets that behavior?

A: Yes, exactly. When Unity created your scene, it added two GameObjects, one called Main Camera and another called Directional Light. If you click on Main Camera in the Hierarchy window, you'll see that it has three components: a Transform, a Camera, and an Audio Listener. If you think about it, that's all a camera actually needs to do: be somewhere, and pick up visuals and audio. The Directional Light GameObject just has two components: a Transform and a Light, which casts light on other GameObjects in the scene.

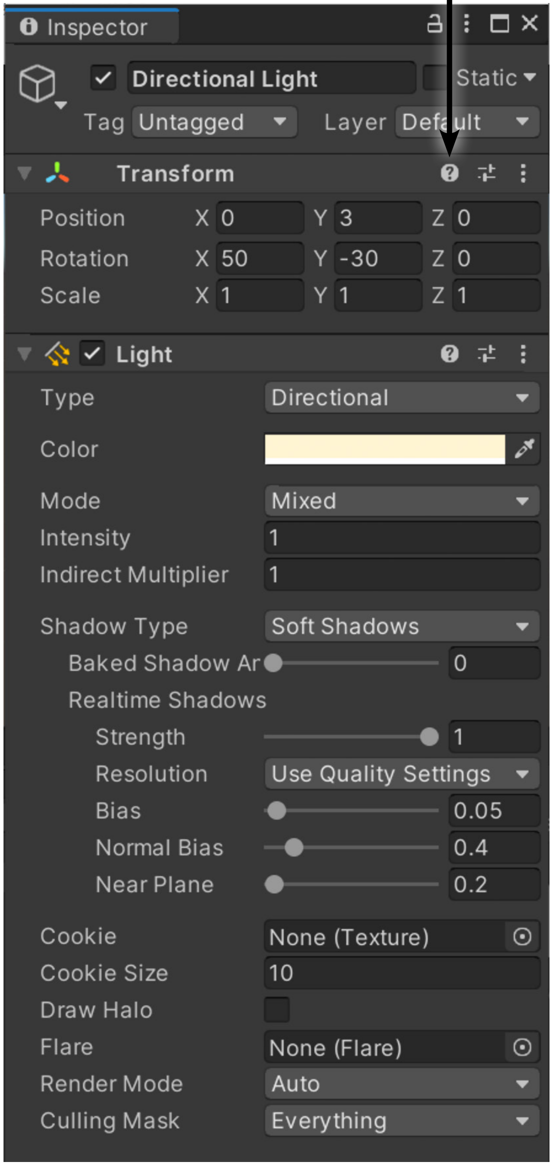
Q: If I add a Light component to any GameObject, does it become a light?

A: Yes! A light is just a GameObject with a Light component. If you click on the Add Component button at the bottom of the Inspector and add a Light component to your ball, it will start emitting light. If you add another GameObject to the scene, it will reflect that light.

Q: It sounds like you're being careful with the way you talk about light. Is there a reason you talk about emitting and reflecting light? Why don't you just say that it glows?

A: Because there's a difference between a GameObject that emits light and one that glows. If you add a Light component to your ball, it will start emitting light—but it won't look any different, because the Light only affects other GameObjects in the scene that reflect its light. If you want your GameObject to glow, you'll need to change its material or use another component that affects how it's rendered.

You can click on the Help icon for any component to bring up the Unity Manual page for it.


Inspector
Directional Light ✓ — Static
Tag: Untagged — Layer: Default

Transform
Position X 0 Y 3 Z 0
Rotation X 50 Y -30 Z 0
Scale X 1 Y 1 Z 1

Light ✓
Type: Directional
Color:
Mode: Mixed
Intensity: 1
Indirect Multiplier: 1
Shadow Type: Soft Shadows
Baked Shadow Ar: 0
Realtime Shadows
Strength: 1
Resolution: Use Quality Settings
Bias: 0.05
Normal Bias: 0.4
Near Plane: 0.2
Cookie: None (Texture)
Cookie Size: 10
Draw Halo:
Flare: None (Flare)
Render Mode: Auto
Culling Mask: Everything

When you click on the Directional Light GameObject in the Hierarchy window, the Inspector shows you its components. It just has two: a Transform component that provides its position and rotation and a Light component that actually casts the light.

Get creative!

We built these Unity Labs to give you a **platform to experiment on your own with C#** because that's the single most effective way for you to become a great C# developer. At the end of every Unity Lab, we'll include a few suggestions for things that you can try on your own. Take some time to experiment with everything you just learned before moving on to the next chapter:

★ Add a few more spheres to your scene. Try using some of the other billiard ball maps. You can download them all from the same location where you downloaded *8 Ball Texture.png* from.

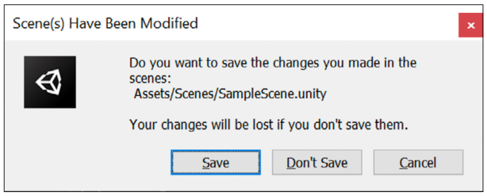
★ Try adding other shapes by choosing Cube, Cylinder, or Capsule from the GameObject >> 3D Object menu.

★ Experiment with using different images as textures. See what happens to photos of people or scenery when you use them to create textures and add them to different shapes.

★ Can you create an interesting 3D scene out of shapes, textures, and lights?

The more C# code you write, the better you'll get at it. That's the most effective way for you to become a great C# developer. We designed these Unity Labs to give you a platform for practice and experimentation.

When you're ready to move on to the next chapter, make sure you save your project, because you'll come back to it in the next lab.. Unity will prompt you to save when you quit.


Scene(s) Have Been Modified

Do you want to save the changes you made in the scenes:
Assets/Scenes/SampleScene.unity

Your changes will be lost if you don't save them.

BULLET POINTS

- The **Scene view** is your main interactive view of the world that you're creating.
- The **Move Gizmo** lets you move objects around your scene. The **Scale Gizmo** lets you modify your GameObjects' scale.
- The **Scene Gizmo** always displays the camera's orientation.
- Unity uses **materials** to provide color, patterns, textures, and other visual effects.
- Some materials use **textures**, or image files wrapped around shapes.
- Your game's scenery, characters, props, cameras, and lights are all built from **GameObjects**.
- GameObjects are the fundamental objects in Unity, and **components** are the basic building blocks of their behavior.
- Every GameObject has a **Transform component** that provides its position, rotation, and scale.
- The **Project window** gives you a folder-based view of your project's assets, including C# scripts and textures.
- The **Hierarchy window** shows all of the GameObjects in the scene.
- **GitHub for Unity** (https://unity.github.com) makes it easy to save your Unity projects in GitHub.
- **Unity Collaborate** also lets you back up projects to free cloud storage that comes with a Unity Personal account.