

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 to write games in C#**.

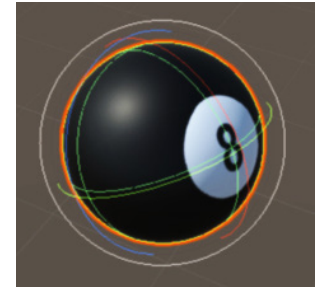
In this first lab, you'll get started with Unity. You'll get oriented with the Unity editor, and you'll start creating and manipulating 3D shapes. That will lay down a foundation to write code in the next lab.

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.

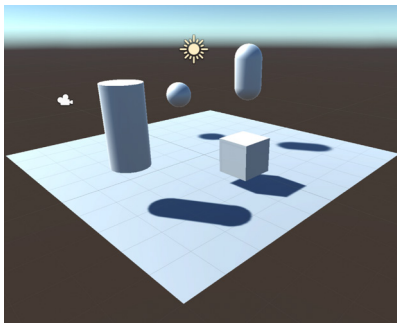


A powerful 2D and 3D scene editor

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://unity.com/solutions/artist-designers>



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.

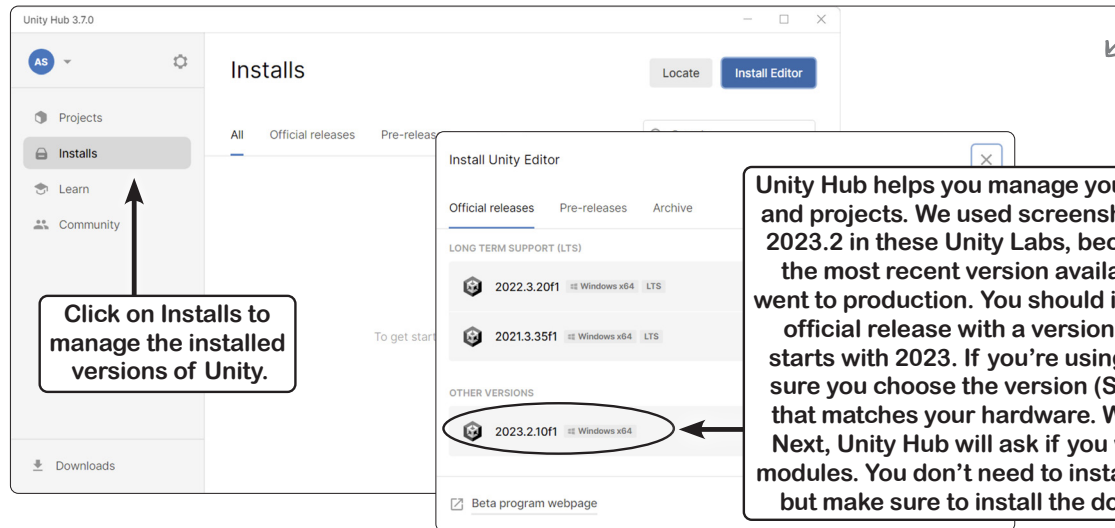
Our Unity Labs will focus on using Unity as a tool to explore C# and practicing with the C# tools and ideas that you've learned throughout the book.

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

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://unity.com/developer-tools>—then install it and run it.

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.



Click on Installs to manage the installed versions of Unity.

Unity Hub helps you manage your Unity installs and projects. We used screenshots from Unity 2023.2 in these Unity Labs, because that was the most recent version available when we went to production. You should install the latest official release with a version number that starts with 2023. If you're using a Mac, make sure you choose the version (Silicon or Intel) that matches your hardware. 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 the Install Editor button** and install the latest official release that starts with **Unity 2023**—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 and macOS here:
<https://docs.unity3d.com/Manual/GettingStartedInstallingUnity.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.



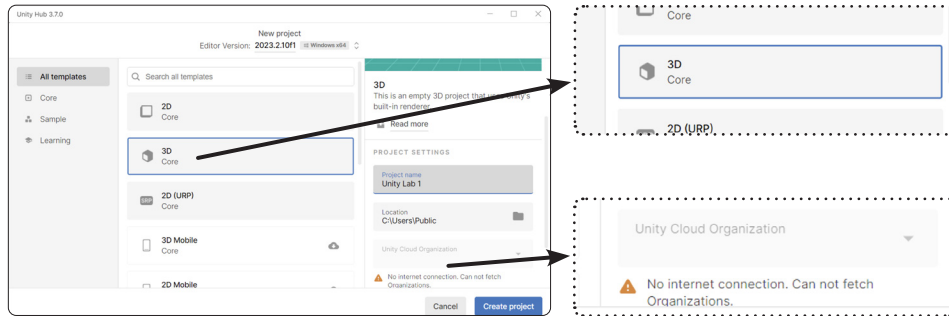
Unity Hub may look a little different.

The screenshots in this book were taken with **Unity 2023.2 in dark mode** and **Unity Hub 3.7.0 in light mode**. 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. The next version, **Unity 6**, is in early release as we go to print. Watch for PDFs of updated labs on our GitHub page: <https://github.com/head-first-csharp/fifth-edition>

At the time we're writing this, Unity is planning on moving away from year-based versions and calling their next version Unity 6.

Use Unity Hub to create a new project

Click the **New project** 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).

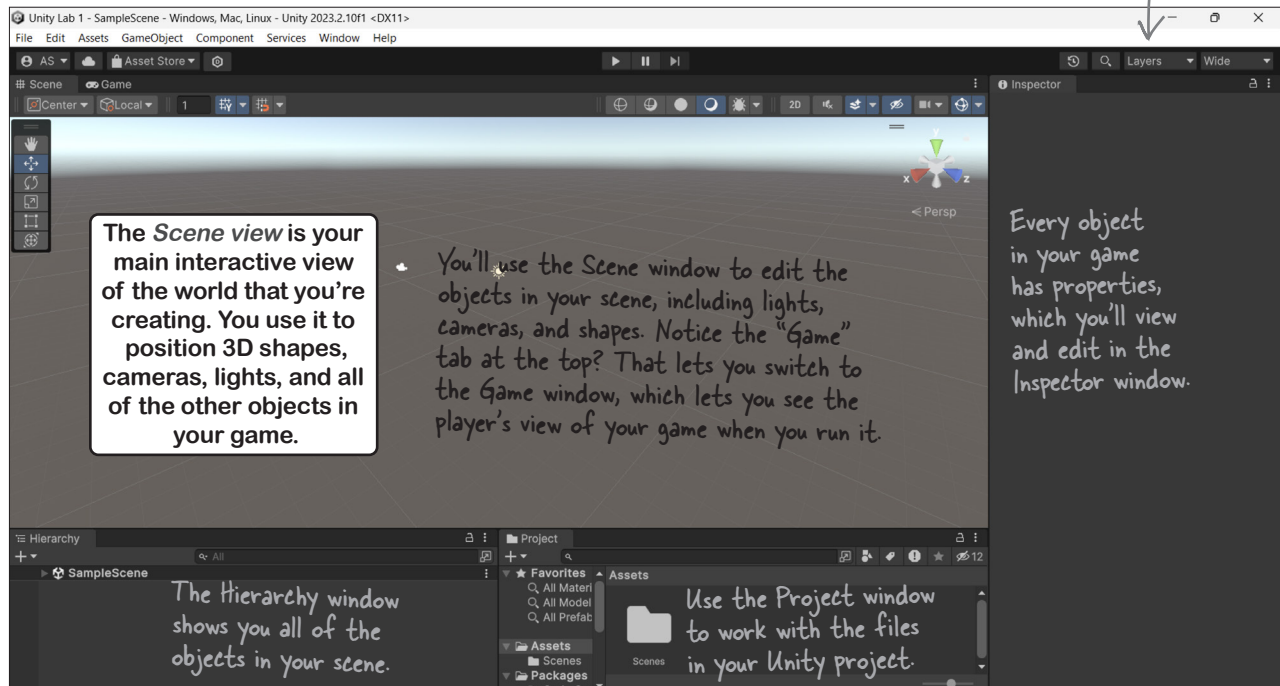


Click Create Project to create the new folder with the Unity project. When you create a new project, Unity generates a lot of files (just like Visual Studio does when it creates new projects for you). It could take Unity a minute or two to create all of the files for your new project.

Work with your project in the Unity editor

Once your project is created, it will load in the **Unity editor**, a powerful tool that you'll use to create 3D environments. Here are some important parts of the Unity editor:

You can use this dropdown to change the layout of the Unity editor.



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

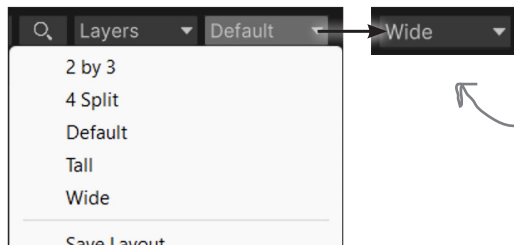
Take control of the Unity layout

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.

When you started up Unity, did you notice that your screen looked a little different from our screenshot? Just like in Visual Studio, the windows and panels in the Unity editor can be rearranged in many different layouts. We chose a layout that works well for screenshots in a book. We also chose dark mode, which we think is easier to read when these pages are printed.

Choose the Wide layout to match our screenshots

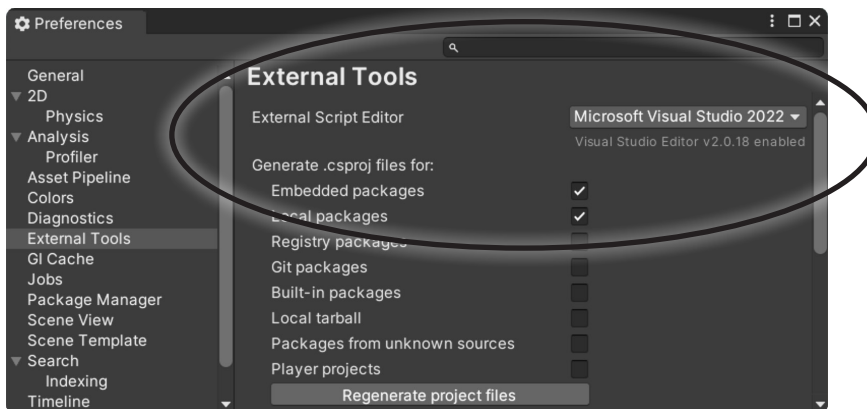
We've chosen the Wide layout because it works well for the screenshots in these labs. Find the Layout dropdown in the toolbar and choose Wide so your Unity editor looks like ours.



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

Set up Unity to work with Visual Studio

The goal of these Unity Labs is to give you an **exciting and fun way to explore C#**. The Unity editor works with Visual Studio and VSCode to make it easy to edit and debug the code for your games. Open the **Unity Preferences Window** (on Windows choose Preferences from the Edit menu; on a Mac choose Settings from the Unity menu). Click on External Tools on the left, click the External Script Editor dropdown, and **choose Visual Studio 2022** (or **Visual Studio Code** if you're using VSCode) from the list of options.



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\Microsoft Visual Studio\2022\Community\Common7\IDE\.

On a Mac it's typically an app called Visual Studio in the Applications folder.

You can download PDFs of all of these Unity Labs and print them out if that makes it easier for you to follow along.

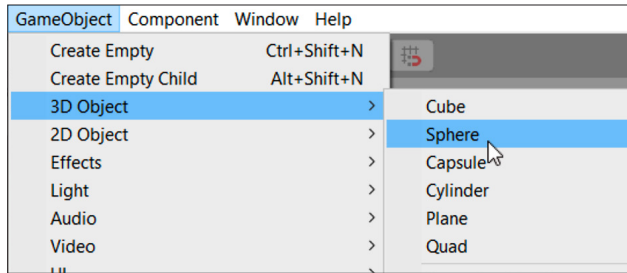
You can use Visual Studio to debug the code in your Unity games. Just choose Visual Studio as the external script editor in Unity's preferences.

You'll write code and do some debugging with Visual Studio or VSCode in the next Unity Lab. This lab is about getting used to the way Unity works so you're ready to do that.

Your scene is a 3D environment

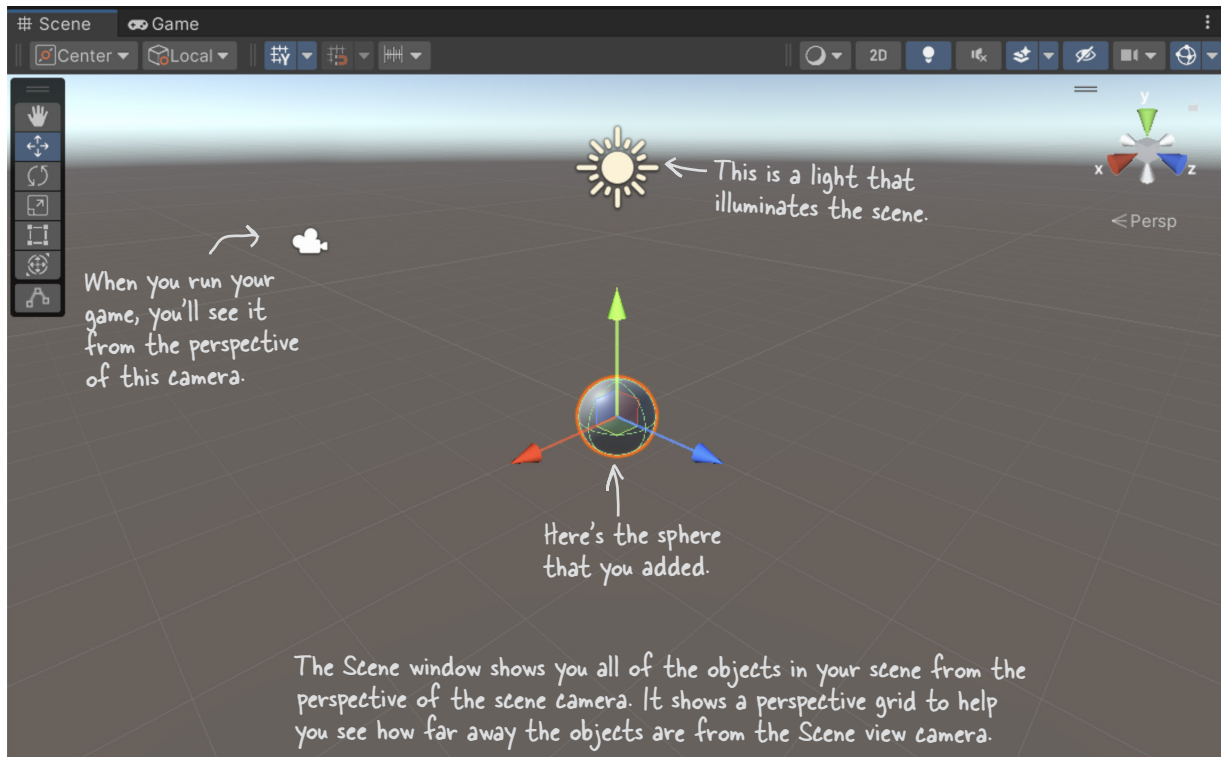
As soon as you start the editor, you're editing a **scene**. You can think of scenes as levels in your Unity games. Every game in Unity is made up of one or more scenes. Each scene contains a separate 3D environment, with its own set of lights, shapes, and other 3D objects. When you created your project,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

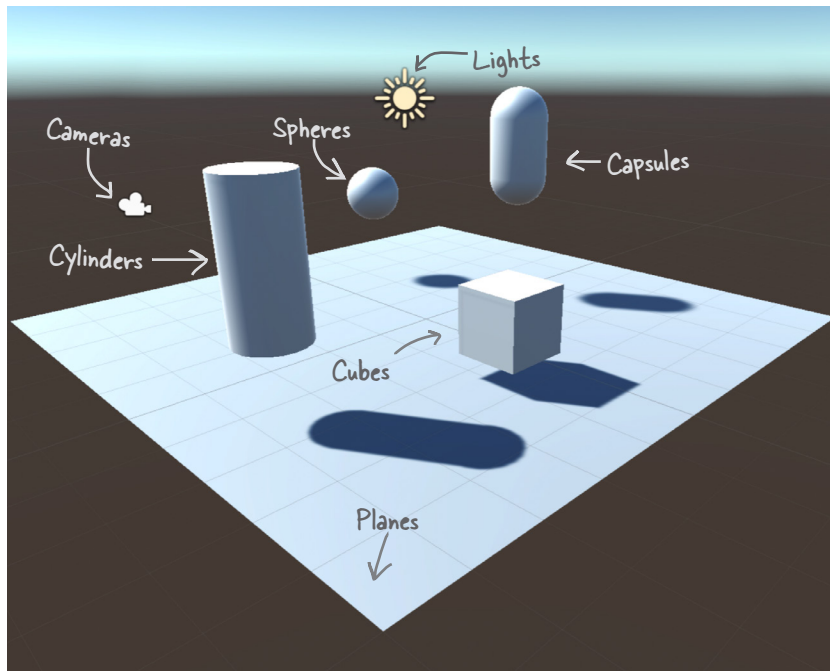
A sphere will appear in your Scene window. Everything you see in the Scene window is shown from the perspective of the **Scene view camera**, which "looks" at the scene and captures what it sees.



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 with different kinds of **GameObjects**, including:



Each **GameObject** contains several **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.

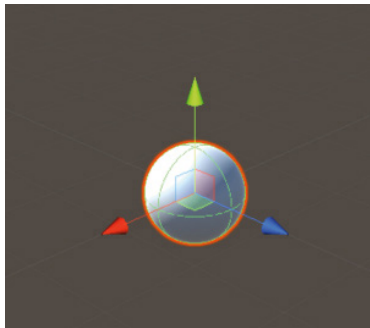
Use the Move Gizmo to move your GameObjects

The Tools panel lets you choose **Transform tools**. If the Move Tool isn't selected, click on the sphere that you just added, then click the Move Tool in the **Tools overlay** to select it.



The *Tools overlay* lets you choose tools to manipulate GameObjects. You'll use the *Move Tool* to move your sphere around the scene. In the Wide view, the Tools overlay is vertical. You can right-click the two lines at the top to change its orientation so it's horizontal, or you can drag it to the toolbar or the side of the window to dock it.

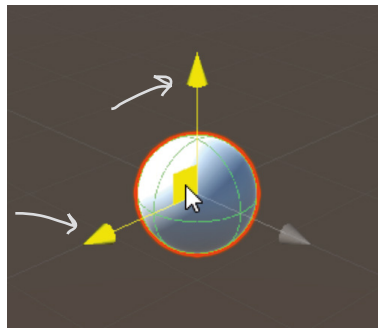
The **Move Tool** lets you use the **Move Gizmo** to move GameObjects around the 3D space. You should see red, green, and blue arrows and a cube appear in the middle of the window. This is the Move Gizmo, which you can use to move the selected object around the scene.



Using the Move Tool displays the Move Gizmo as arrows and a cube on top of the GameObject that's currently selected. When you click the sphere and then choose the Move Tool, you'll see the Move Gizmo appear on the sphere. Click anywhere else in the scene to deselect the sphere and the Move Gizmo goes away.

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.



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



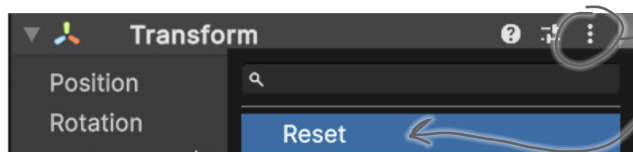
If you accidentally deselect a GameObject, just click on it again. If it's not visible in the scene, you can select it in the *Hierarchy window*, which shows all of the GameObjects in the scene. When you reset the layout to Wide, the Hierarchy window is in the lower-left corner of the Unity editor.

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.

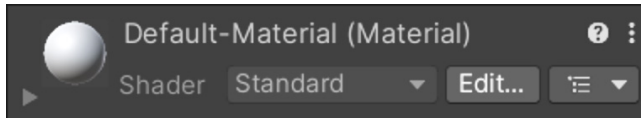
Save your scene often! Use File >> Save or Ctrl+S / ⌘S to save the scene right now.

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:



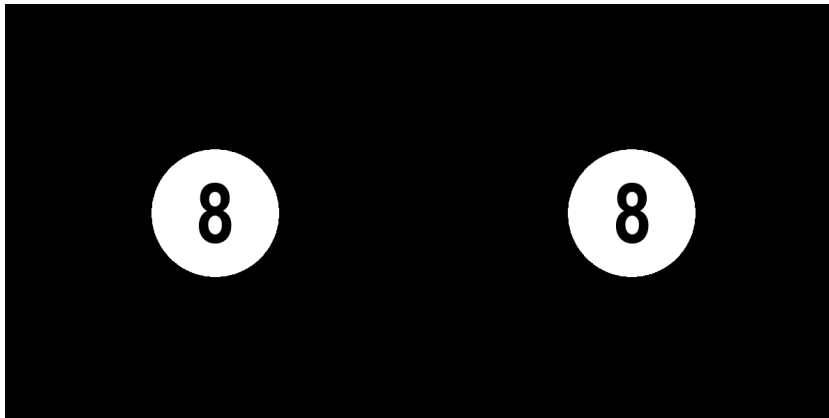
We'll make your sphere more interesting by adding a **texture**—that's just a simple image file that's wrapped around a 3D shape, almost like you printed the picture on a rubber sheet and stretched it around your object.

② Go to our Billiard Ball Textures page on GitHub.

Go to <https://github.com/head-first-csharp/fifth-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.



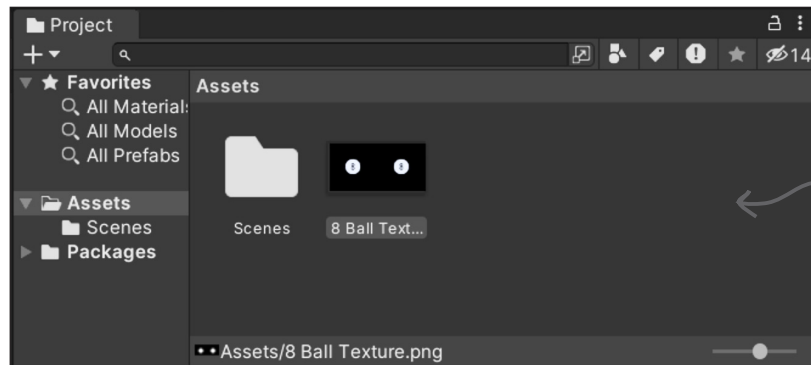
← We designed this image file so that it looks like an 8 ball when Unity “wraps” it around a sphere.

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.

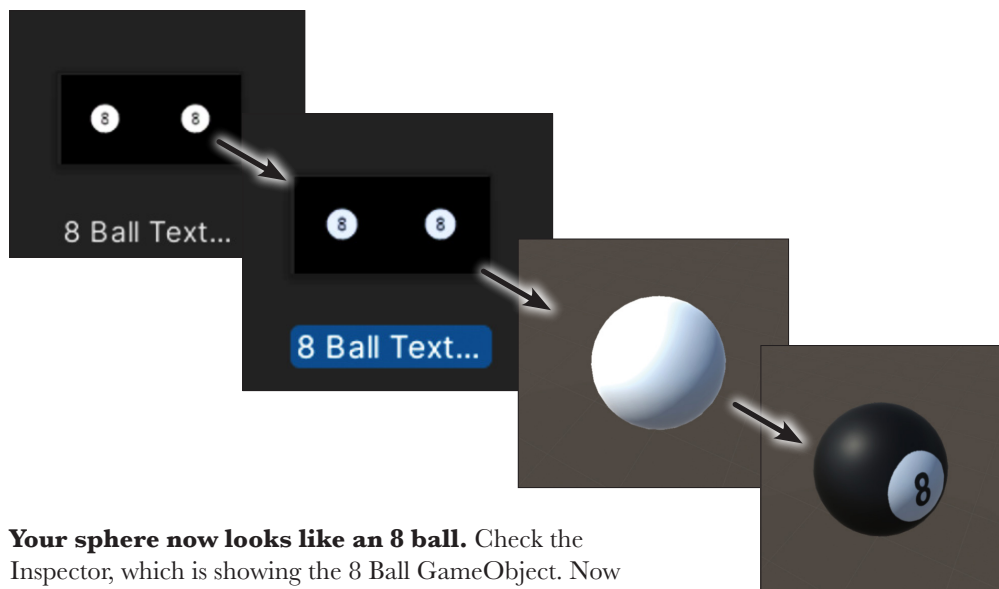
Right-click on the Assets folder in the Project window, choose **Import New Asset** and import the texture file. You should now see it when you click on the Assets folder in the Project window.



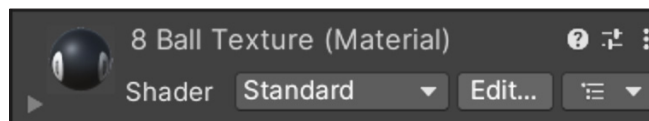
You right-clicked inside the Assets folder in the Project window to import the new asset, so Unity imported the texture into that folder.

⑤ Add the texture to your sphere.

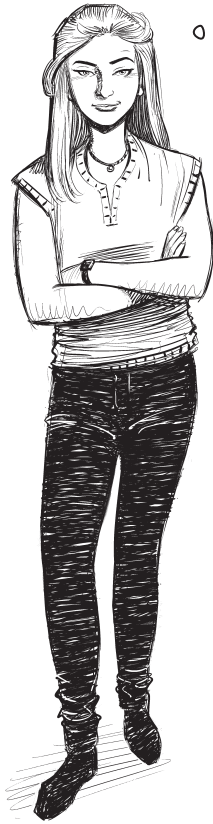
Now you just need to take that texture and “wrap” it around your sphere. Click on 8 Ball Texture in the Project window to select it. Once it’s selected, **drag it onto your sphere**.



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:



Check your Assets window again. Unity created a new Materials folder in it and added a material called 8 Ball Texture.



I'm learning C# for my job, not to write video games. Why should I care about Unity?

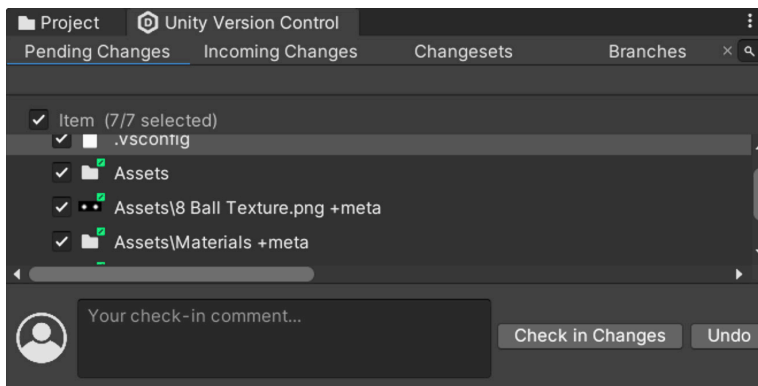
Unity is a great way to really “get” C#.


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 MAUI 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 accelerate your understanding of C#.

Do you want to make sure your Unity projects are always backed up? Try Unity Version Control.

Unity Version Control is a version control system that lets you back up your projects to cloud storage that comes free with your Unity account—and it's **built right into the Unity editor**, which makes it easy for you to use.

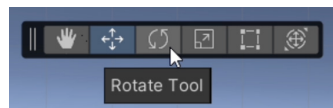


Click the  **Unity Version Control** button in the toolbar to open the Unity Version Control window. The first time you use it, you'll get an option to log in or sign up. When you sign in with your Unity ID, you'll get to a web page where you can sign into your Unity account, then sign up for the free Unity VCS level and join your default organization. Then you can check in changes any time you want.

Go to the Head First C# GitHub page for a free PDF that gives you step-by-step instructions for setting Unity Version Control: <https://github.com/head-first-csharp/fifth-edition>

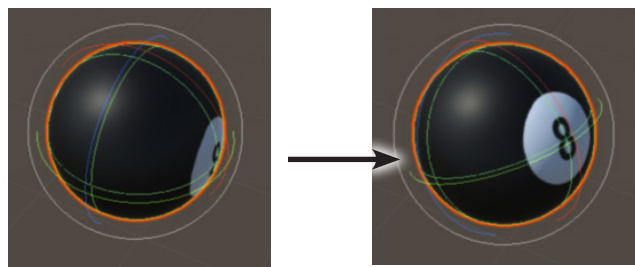
Rotate your sphere

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.

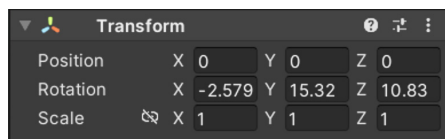


We switched the Tools overlay to a horizontal view by right-clicking on the two lines and choosing Horizontal. Try it out.

- 1 **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.



- 2 **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.

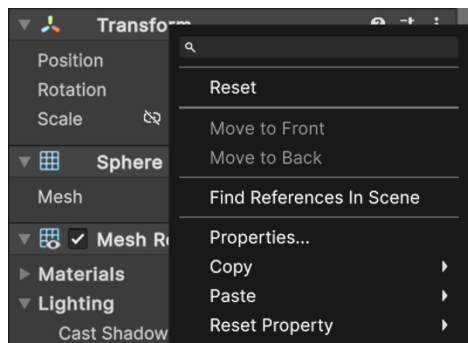


Relax

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

- 3 **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.

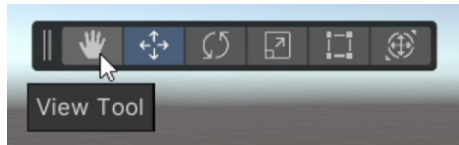
Use these options from farther down in the context menu to reset the position and rotation of a GameObject.

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

Move the Scene view camera with the View Tool and Scene Gizmo

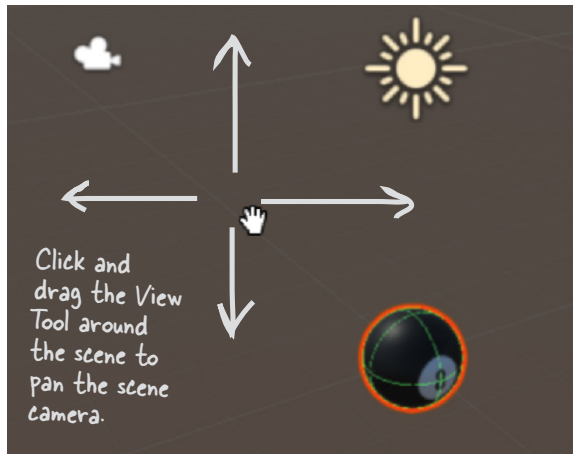
Use the mouse scroll wheel or scroll feature on your trackpad to zoom in and out, and toggle between the Move and Rotate Gizmos. Notice that the sphere changes size, but the Gizmos don't. The Scene window in the editor shows you the view from a virtual **camera**, and the scroll feature zooms that camera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

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



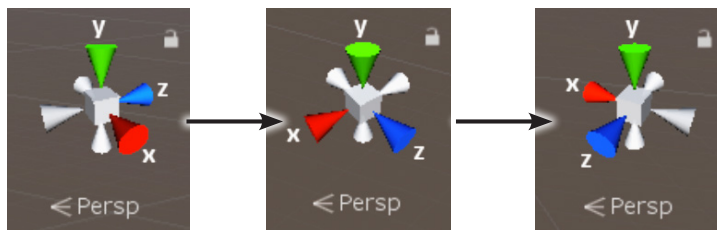
Hold down **ALT** (Windows) or **Option** (Mac) while dragging and the View Tool turns into an eye and rotates the view around the center of the window.

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



When the View 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 View 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/SceneView.Navigation.html>

Take a minute and look at this page—it's got some really useful stuff. ↗

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.

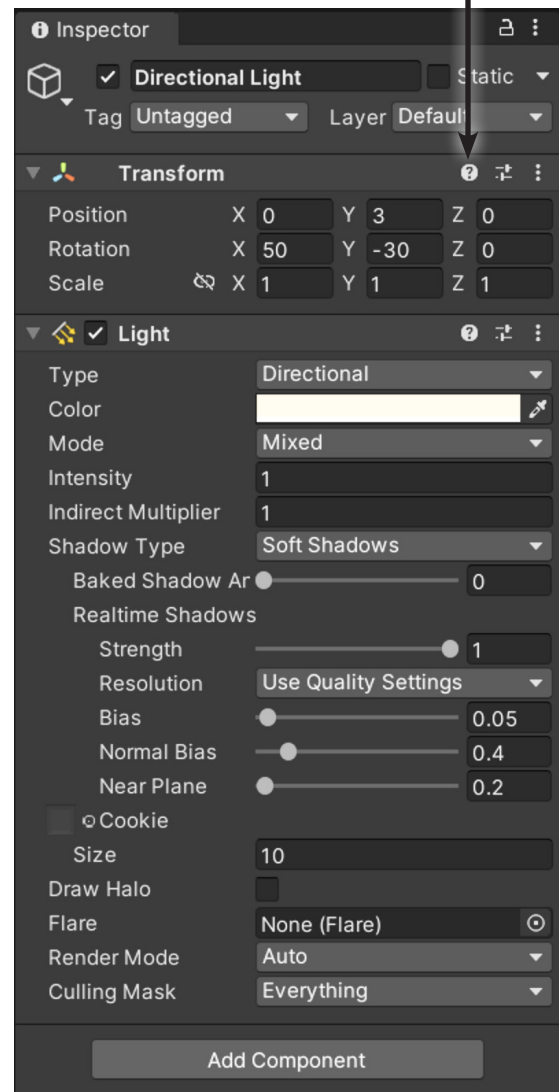
Q: If I add a Light component to any GameObject, does it become a light?

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.



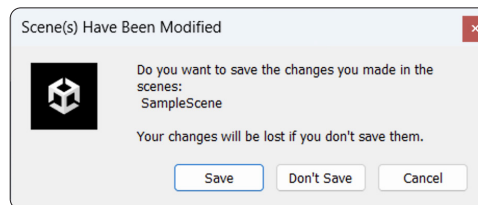
When you click on the Directional Light GameObject in the Hierarchy window, the Inspector shows you its components. It just has two: a Transform component that provides its position and rotation and a Light component that actually casts the light. What do you think you'll use the Add Component button for?

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. This lab **lays down the foundation** to start writing Unity code—which you'll do in the next lab. At the end of each Unity Lab, we'll include suggestions for things that you can try on your own. Take some time to experiment with everything you just learned before moving 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*.
- ★ 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?

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.



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.

Bullet Points

- The **Scene view** is your main interactive view of the world that you're creating.
- When you select an object and use the **Move Tool**, Unity displays the **Move Gizmo** that lets you move objects around your scene.
- The **View Tool** lets you pan around the scene. 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. **Components** are the building blocks for 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.
- **Unity Version Control System (VCS)** is an easy way to back up projects to free cloud storage that comes with a Unity Personal account. Download a PDF to help you set up version control in Visual Studio, VSCode, and Unity for free from our GitHub page: <https://github.com/head-first-csharp/fifth-edition>