Appendix: Setting Up Your Development Environment

Learning to program requires that you actually DO lots of programming – not just read about it, hear about it, or watch someone else do it, but actually do it yourself. That means that you'll need to set up a programming environment so you can learn all the stuff in this book by actually slinging code. For the topics covered in this book, you'll need to install Visual Studio Community 2019 and the Unity game engine. Doing this will let you use Visual Studio to write C# console applications and will also let you write C# scripts in Unity as you develop Unity games. We'll discuss how to install each below.

A.1. Visual Studio Community 2019

Microsoft's commercial Integrated Development Environments (IDEs) are called Visual Studio, and they come in various capability (and cost) levels. Because there are lots of people who want to program as a hobby rather than as a commercial endeavor, Microsoft also distributes free IDEs. We used the free Visual Studio Community 2019 IDE to develop all the code in this book. Although the source code provided on the accompanying web site can be used in any IDE, if you have Visual Studio Community 2019 you can just open up and use the projects as is.

Here's how you can get this IDE installed:

1. Download Visual Studio Community 2019 at http://www.visualstudio.com/.
2. Run the install file you downloaded.

Be sure to check the following items in the Workloads area of the installer:

Windows: .NET desktop development
Mobile & Gaming: Game development with Unity

You should also check the following in the Individual components area of the installer:

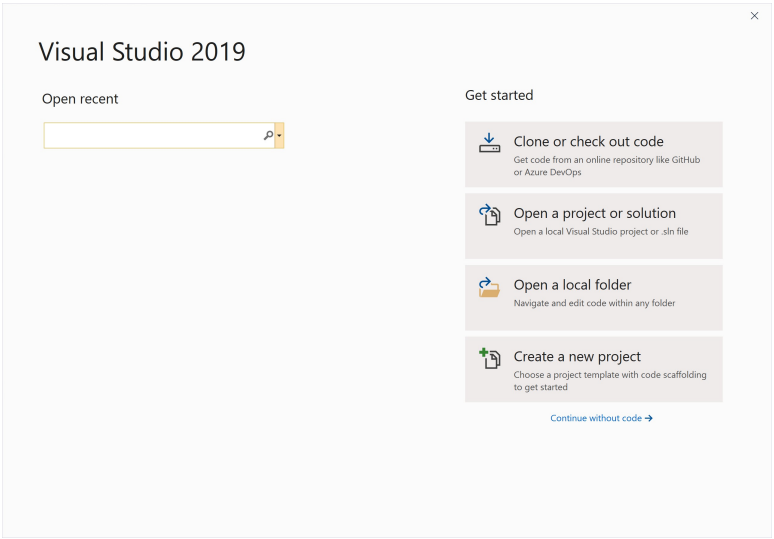
Code tools: Help Viewer

You can certainly select other Workloads and Individual components if you'd like, but we just installed the three items above. After selecting those items, click the Install button and wait patiently while everything is installed.

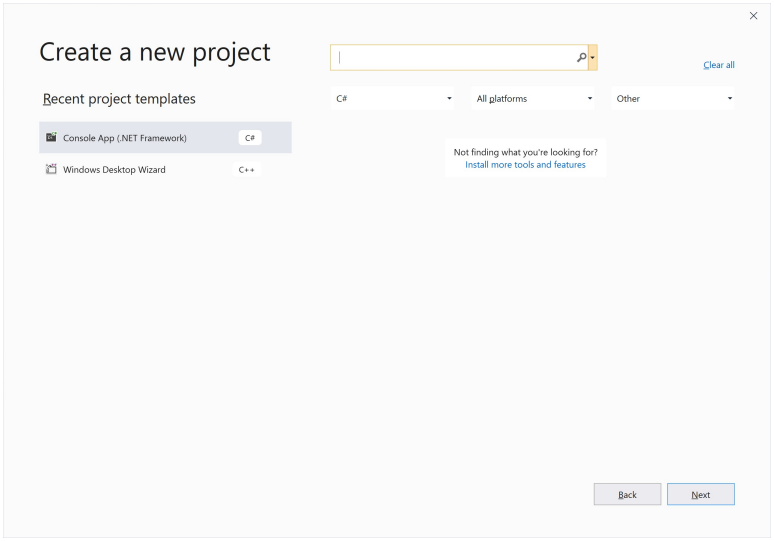
At this point, you should have a working version of the IDE installed. Let's check to make sure you do.

We put a shortcut on our desktop to make launching the IDE easier; for us, the IDE is installed at C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\Common7\IDE\devenv.exe. Of course, you can also use Windows to search on Visual Studio and click the Visual Studio 2019 result instead if you prefer.

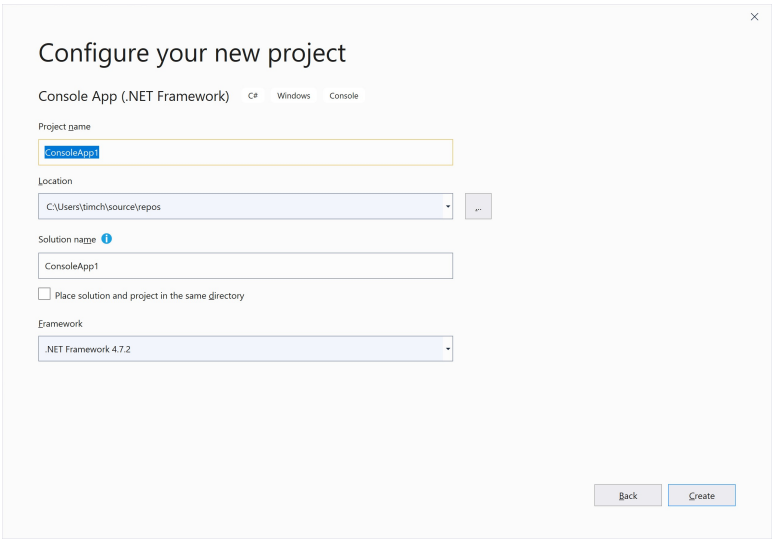
During the first startup of Visual Studio, be sure to change Development Settings to Visual C# before clicking the Start Visual Studio button on the bottom right. When Visual Studio starts up, you should end up with something like the window below. Yours may not match exactly, but it should be similar.



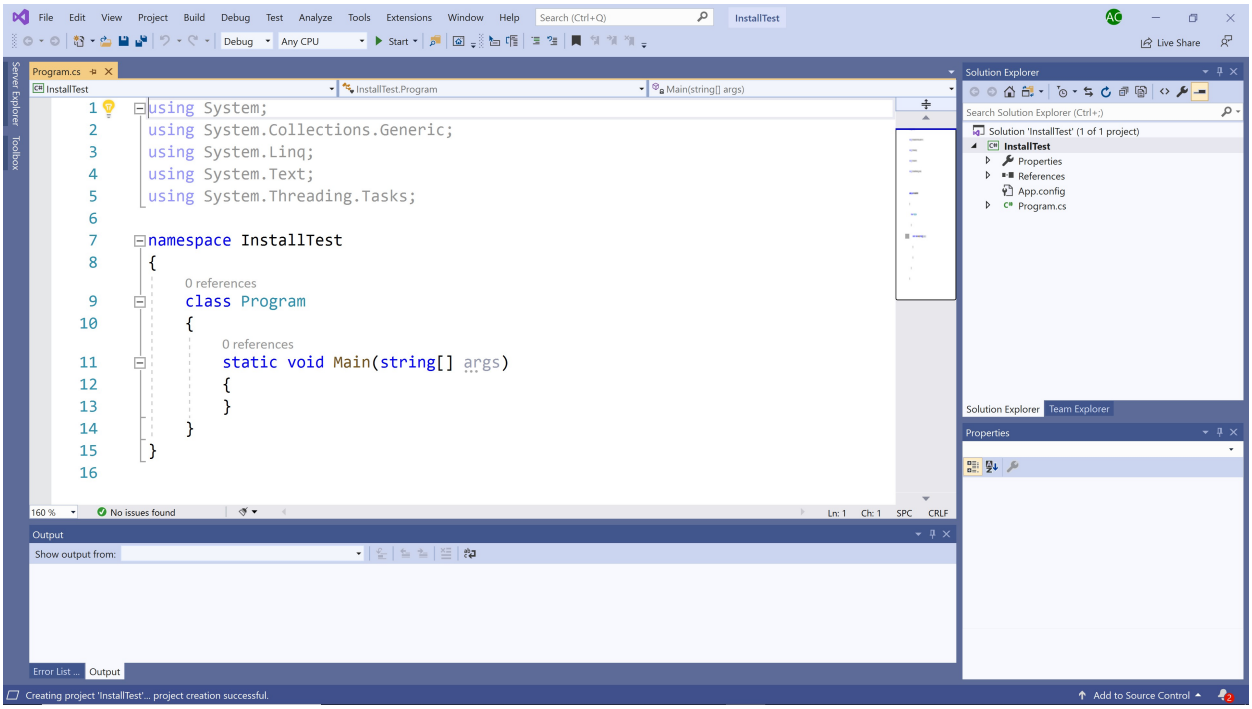
We're going to compile and run a simple program to make sure the IDE installed correctly. Click the Create a new project box on the lower right, which will bring you to a screen like the one below. Select the Console App (.NET Framework) C# button on the upper left, then click the Next button on the lower right.



You'll end up at a screen like the one shown below. Change the Project name from ConsoleApp1 to InstallTest in the text box at the top left of the screen, change the Location to put the project wherever you want it to be saved, then click the Create button at the bottom right.



After a moment of two, you'll come back to the main screen of the IDE, which should look like this (we zoomed in on the Code window in the upper left using Ctrl + middle mouse wheel):



If you have the Toolbox pane displayed on the left-hand side, you can simply click the Auto Hide "pin" just to left of the X in the upper right corner of the Toolbox pane to make it hide itself on the left.

When we told the IDE we wanted a new console app project, it automatically gave us a template for that kind of application. Compile the application by pressing F6 or by selecting Build > Build

Solution from the top menu bar[1]. Once you see a "Build succeeded" message in the bar at the bottom of the IDE, you can actually run the application. Do that by pressing Ctrl+F5; when you do, you should get the Command Prompt window below. Just press any key to return to the IDE.

Your Command Prompt window probably has the default black background with light grey text on it rather than a white background with black text. Although that works fine on a computer screen, it looks horrible in an electronic or printed book! If that's the only difference between your output and the output above, you're good to go. If you decide you want to change your defaults, open a Command Prompt window, right click near the top of the window, select Defaults, click the Colors tab, and change away. We also changed the Font, so feel free to customize that window as much as you want.

Exit the IDE. You certainly don't need to save the project if you don't want to – you'll get plenty of practice creating and saving projects – but feel free to do so if you want to.

If everything worked as described here, Visual Studio Community 2019 is installed and you're ready to move on to setting up the documentation. If your IDE isn't working, you should try working through the install process again or get online with Microsoft to ask for help.

When we're developing code, it's really useful to have the help documentation available directly within the IDE. The default for help documentation is to simply access help (through the IDE) online. If your computer always has connectivity to the Internet, this should work fine. If you're sometimes disconnected while you're programming, though, you'll probably want to set up your environment to use local help instead.

Start up the IDE, then click Help > Add and Remove Help Content. The IDE starts up the Microsoft Help Viewer, which lets you add local content. At a minimum, you should add the following topics:

.NET API Reference (already set to be added by default)

---

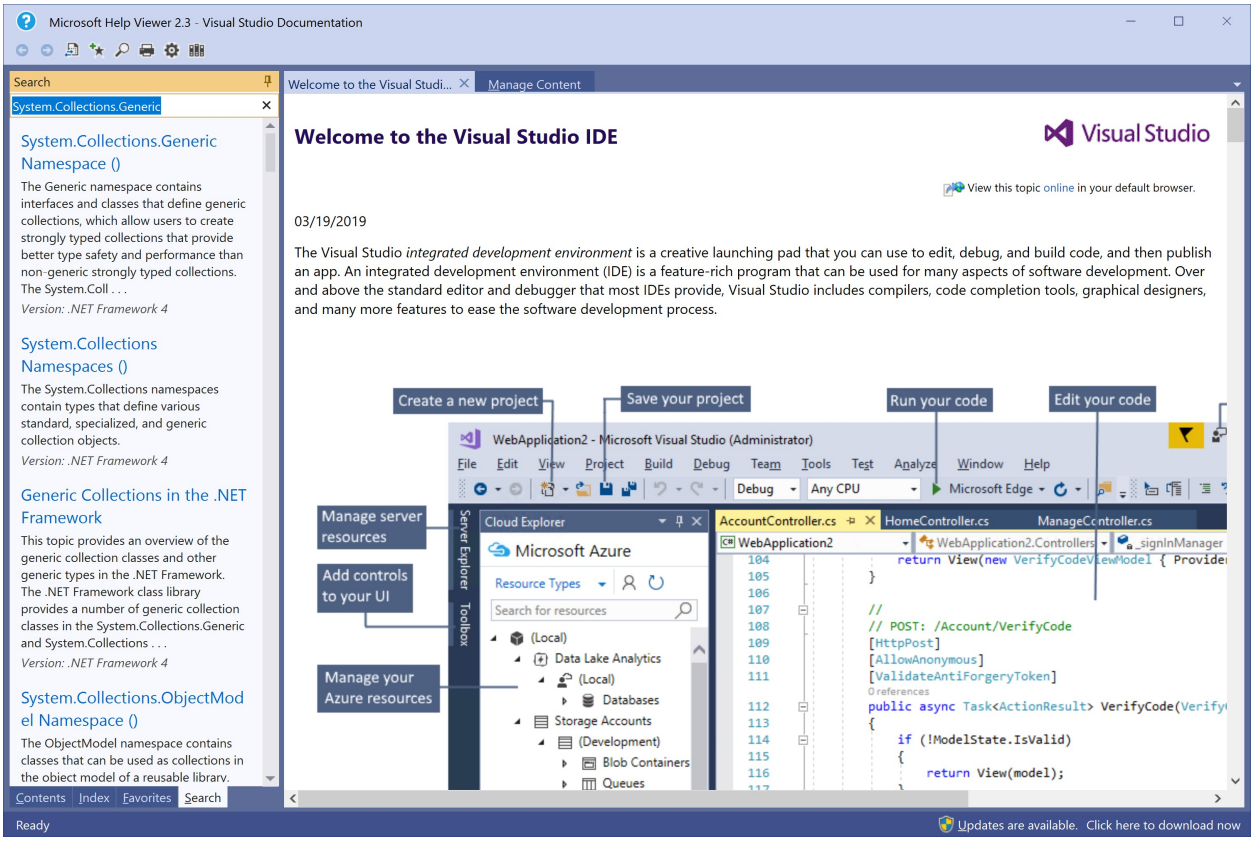
[1] We've actually seen the keyboard shortcut for building the solution as F7 and Ctrl+Shift+B also. If you select Build > Build Solution from the top menu bar, the keyboard shortcut for that action is shown to the right. You can also customize the keyboard shortcuts in Visual Studio if you'd like; see https://msdn.microsoft.com/en-us/library/5zwses53.aspx

Visual C# (already set to be added by default)

Click the Update button near the lower right of the Help Viewer and wait patiently while the update packages are downloaded and installed.

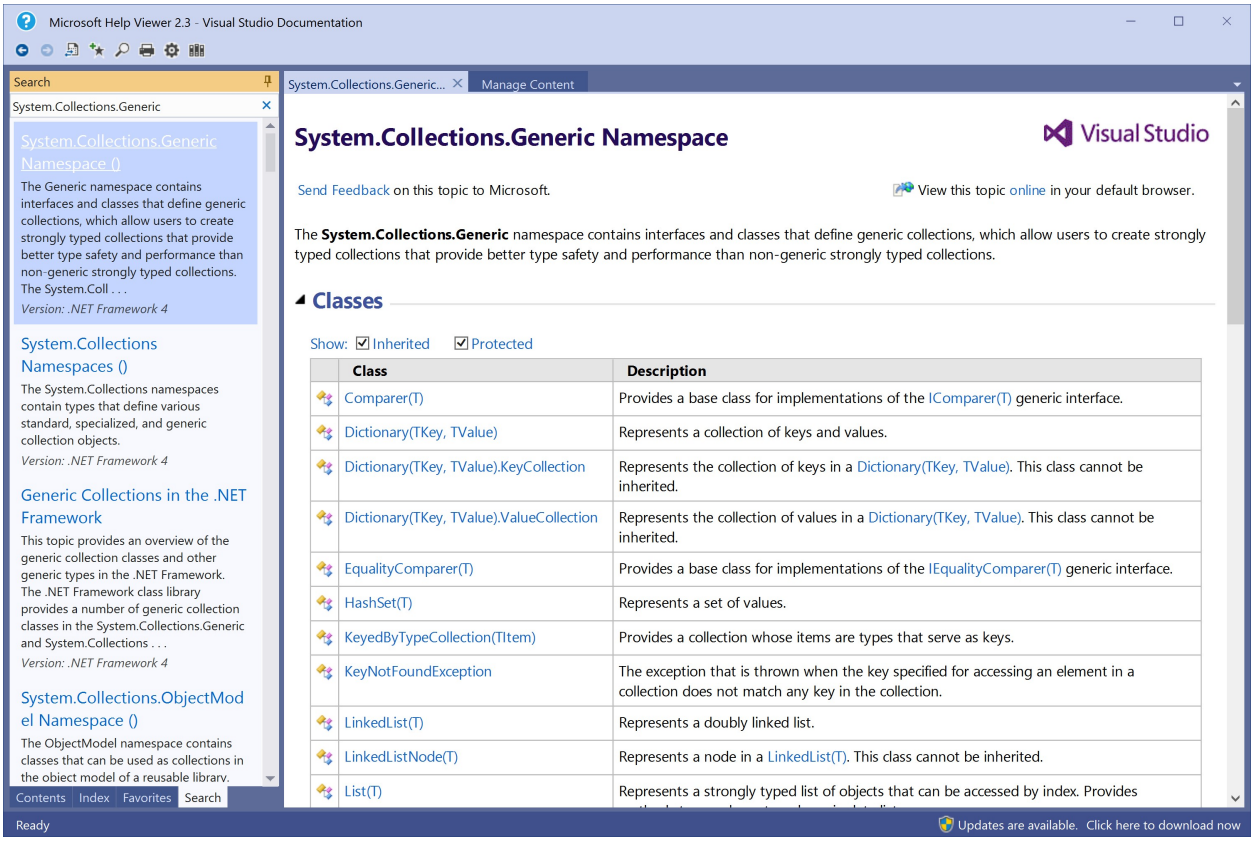
Finally, you can read the help documentation through a browser (which will access the online documentation) or through the Microsoft Help Viewer (which will access the local help content). Both approaches provide access to the same help content, they just look slightly different – and, of course, if you're offline you can only access the local help. If you prefer using a browser, you don't need to make any changes because that's the default. If you prefer using the help viewer (like we do), select Help > Set Help Preference > Launch in Help Viewer.

Let's make sure the local documentation installed properly. Click Help > View Help. Enter System.Collections.Generic in the search box and press Enter. You should get a page of search results like the one shown below.



Your page will look different if you're using online help, but you should see a set of results somewhere on the page.

Click on the link that say System.Collections.Generic Namespace () and you should get a window of documentation like the one shown below.



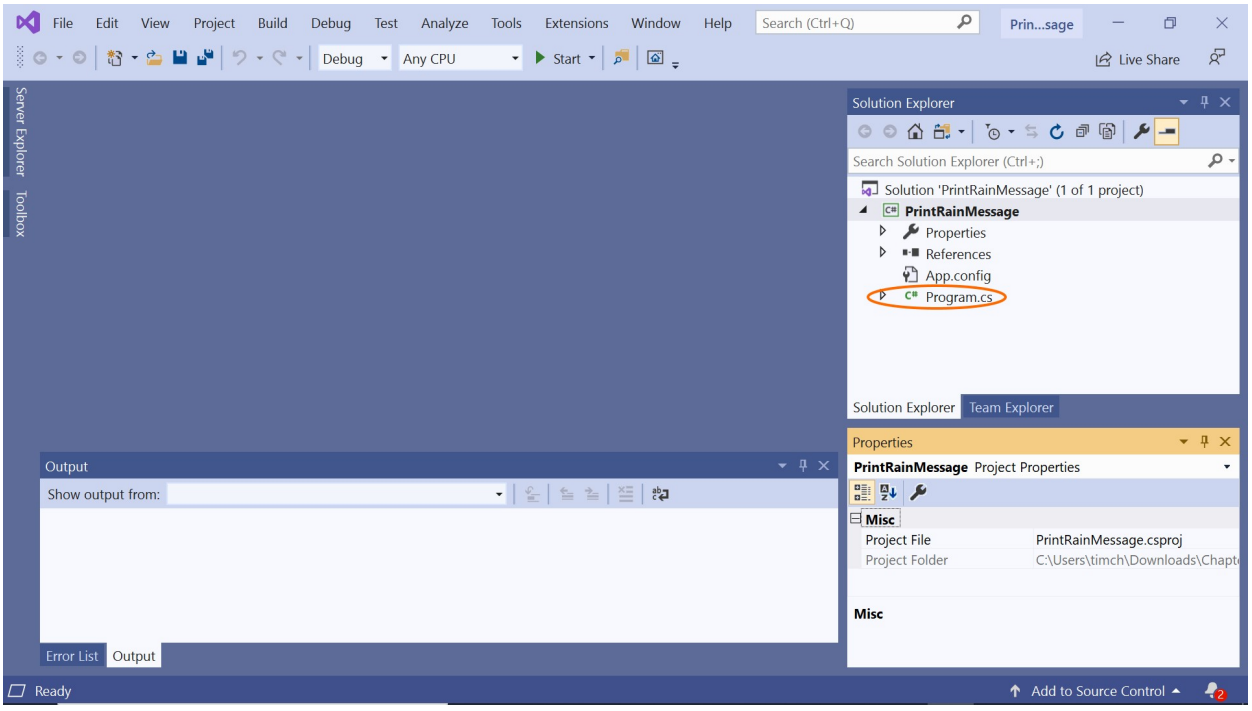
You should also know that you can always look at the C# documentation online at https://msdn.microsoft.com/en-us/library/; just click on the magnifying glass near the upper right of the page and use the search box to look for a specific C# topic.

Opening Downloaded Visual Studio Projects

When you download a Visual Studio project, you can open it in Visual Studio by double clicking the solution file. The solution file has a .sln file extension, but in case you're not showing file extensions in Windows Explorer it's the file marked "Microsoft Visual St...", "Visual Studio Solu...", or something similar as the Type. Let's see how that works.

Download the Chapter 2 code from the Burning Teddy web site and extract the zip file somewhere on your computer.

Navigate into the folder called PrintRainMessage (it will be wherever you extracted the Chapter 2 code zip file) and double click the solution file. When the project opens, you might think something is wrong because the code pane doesn't show any code. Simply click Program.cs in the Solution Explorer to see the code in that file.



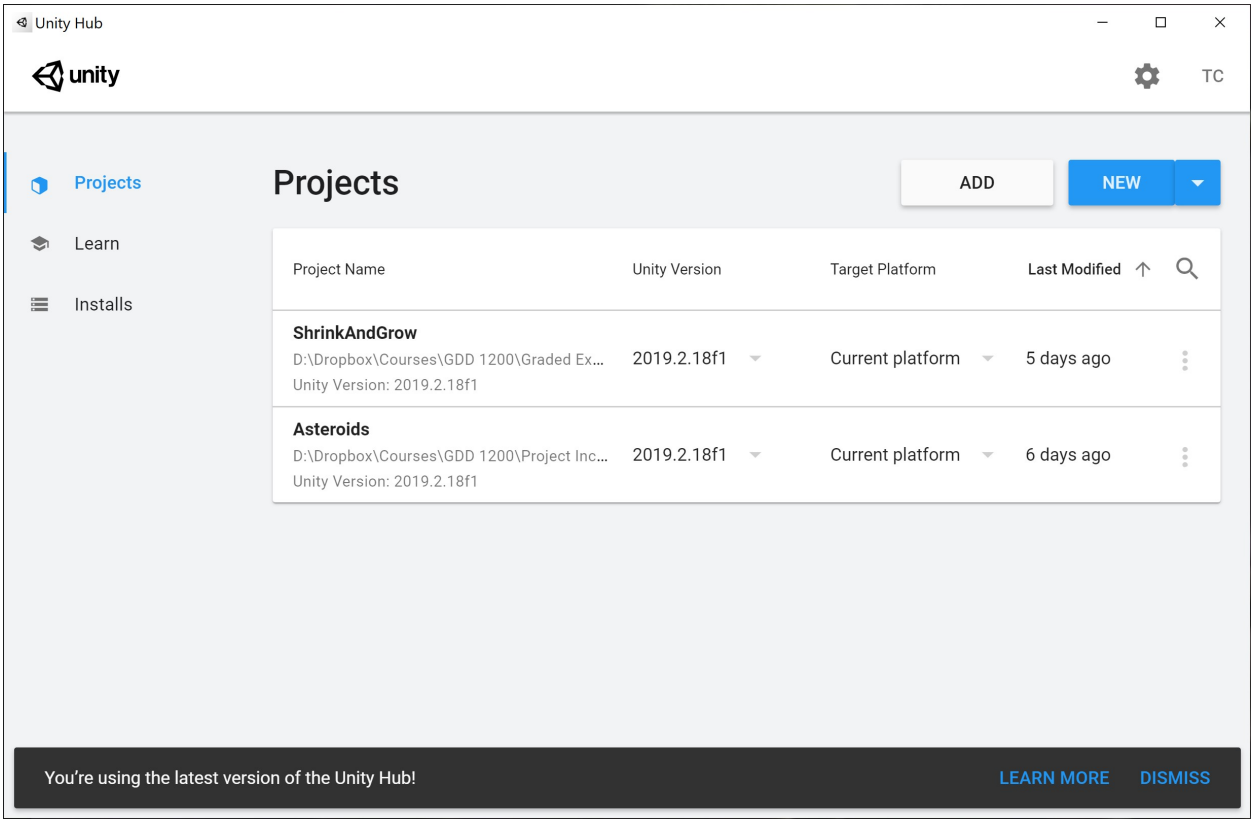
That's all you need to get started with Visual Studio, so let's move on to Unity.

A.2. Unity

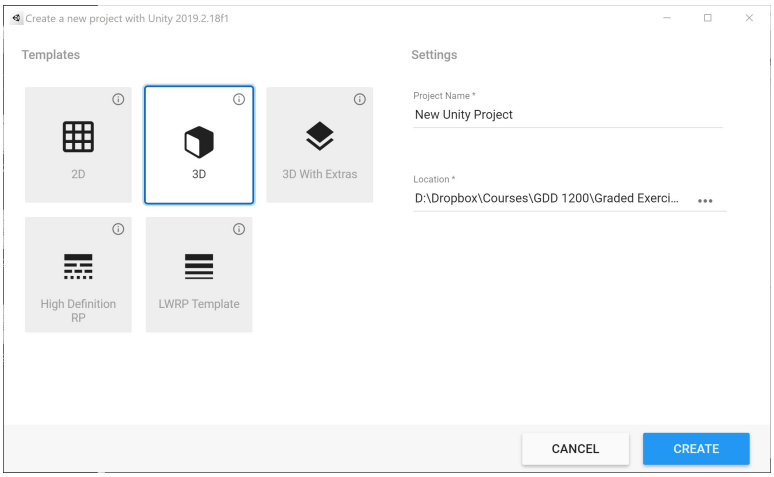
Here's how you can get Unity installed:

1. Download Unity Hub from http://unity.com/. At the time of this writing, you'll need to click Get started, select the Individual tab, click Get started under Individual, click Start here under First-time Users, and click Agree and download. The sequence of steps may be different by the time you're doing this, but the bottom line is you're downloading Unity Hub (even though the web pages don't say you're doing that, your downloaded file should be named something like UnityHubSetup.exe)
2. Run the install file you downloaded. Just accepting the default values as you go along should work fine, but you can of course change them if you want

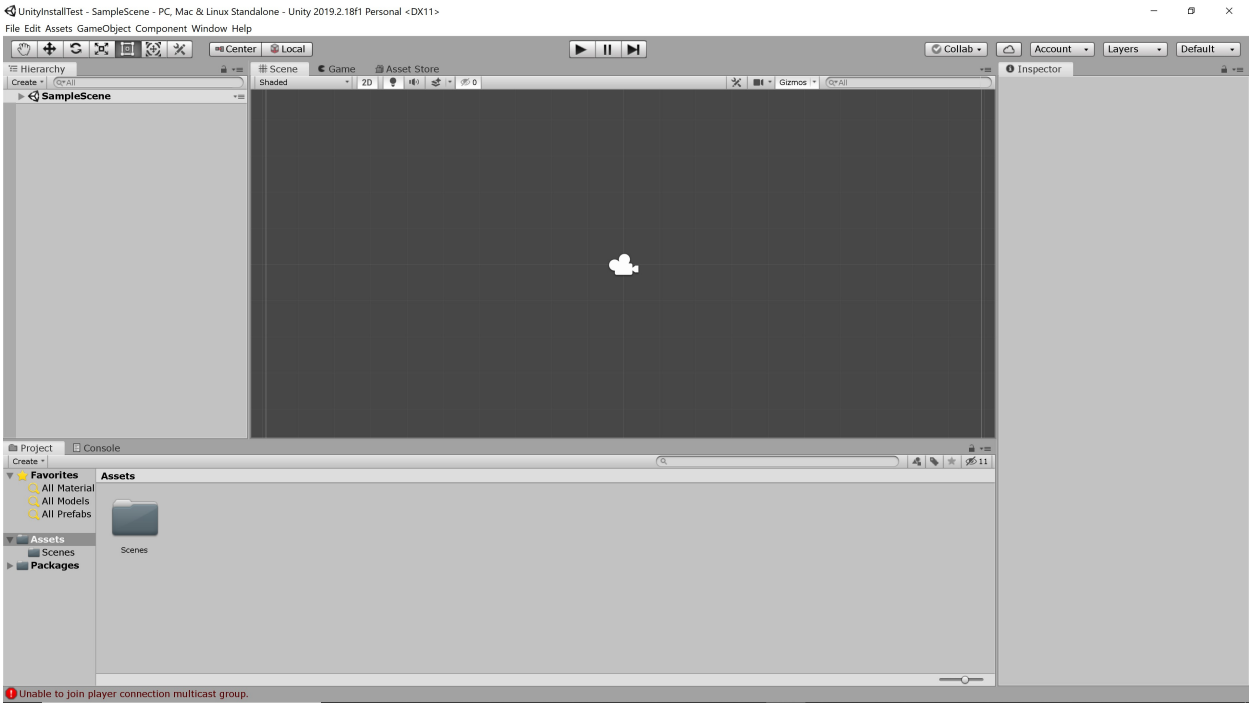
Unity Hub lets you have multiple versions of Unity installed on your computer and using Unity Hub is the standard way to install new versions of Unity. By default, Unity installs a shortcut to Unity Hub on your desktop when it installs. Double click that shortcut; you should end up with a window like the one below (though you won't have the list of recent projects, of course!). You may actually get a popup asking you to sign into your Unity account first; you can choose to create an account and sign into it or work offline, your choice.



Click New near the upper right corner; you should end up with a window like the one below.

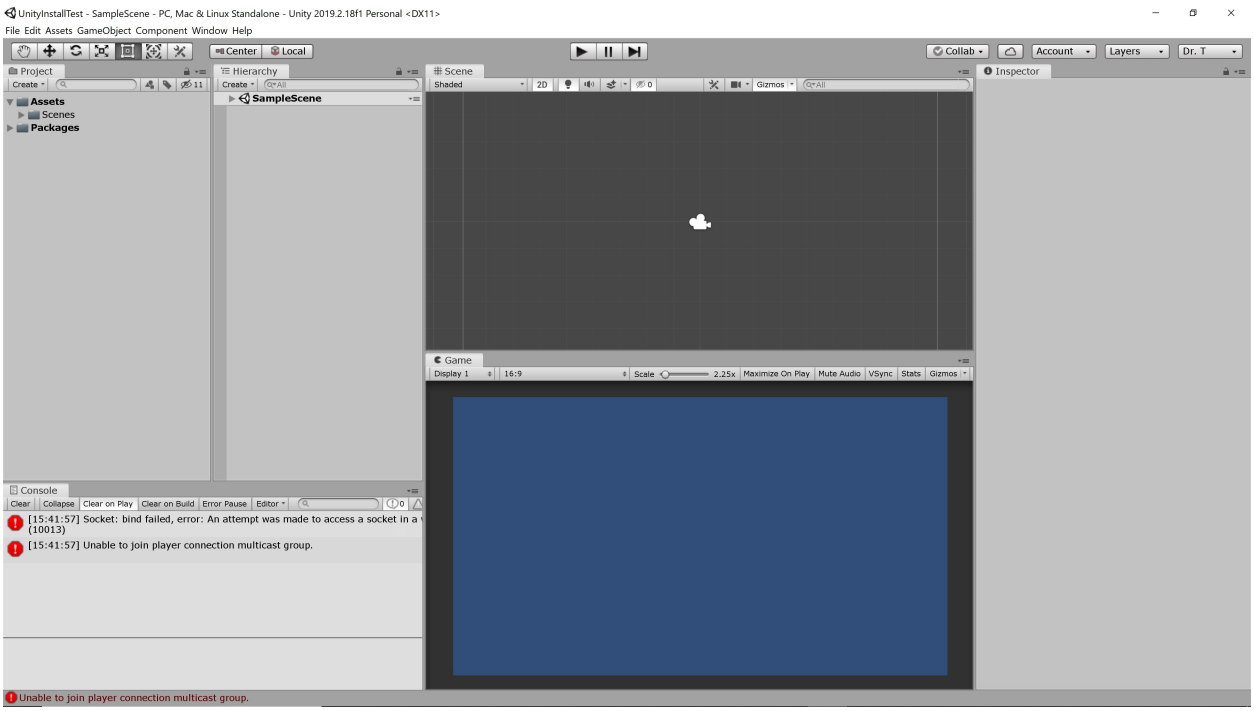


Click the 2D button to create a 2D project, change the Project Name to UnityInstallTest, change the Location to wherever you want to save the project, then click the Create button at the lower right. After a moment of two, you'll come to the Unity editor, which should look like this:

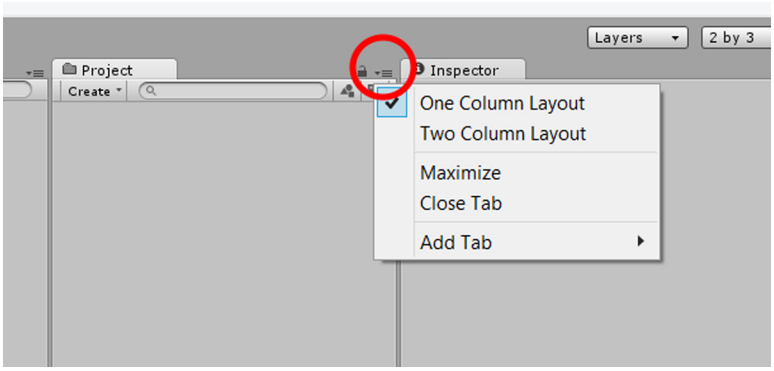


If the editor has a Services tab on the right, you can just right click on that tab and select Close Tab.

This is the default layout for the Unity editor, but you can change the layout by clicking the layout dropdown near the upper right and selecting a different layout. You can even configure and save your own layout; because we prefer a different layout when we develop in Unity, we'll show you how to configure and save the layout we'll use throughout the book (a layout that's also used in Jeremy Gibson's excellent Introduction to Game Design, Prototyping, and Development book). You don't have to do this – you can use any layout you want! – but go ahead and follow along if you want to use the layout we use. Once we're done our layout will look like this (don’t worry about the errors showing up in the image below, they won’t affect us):

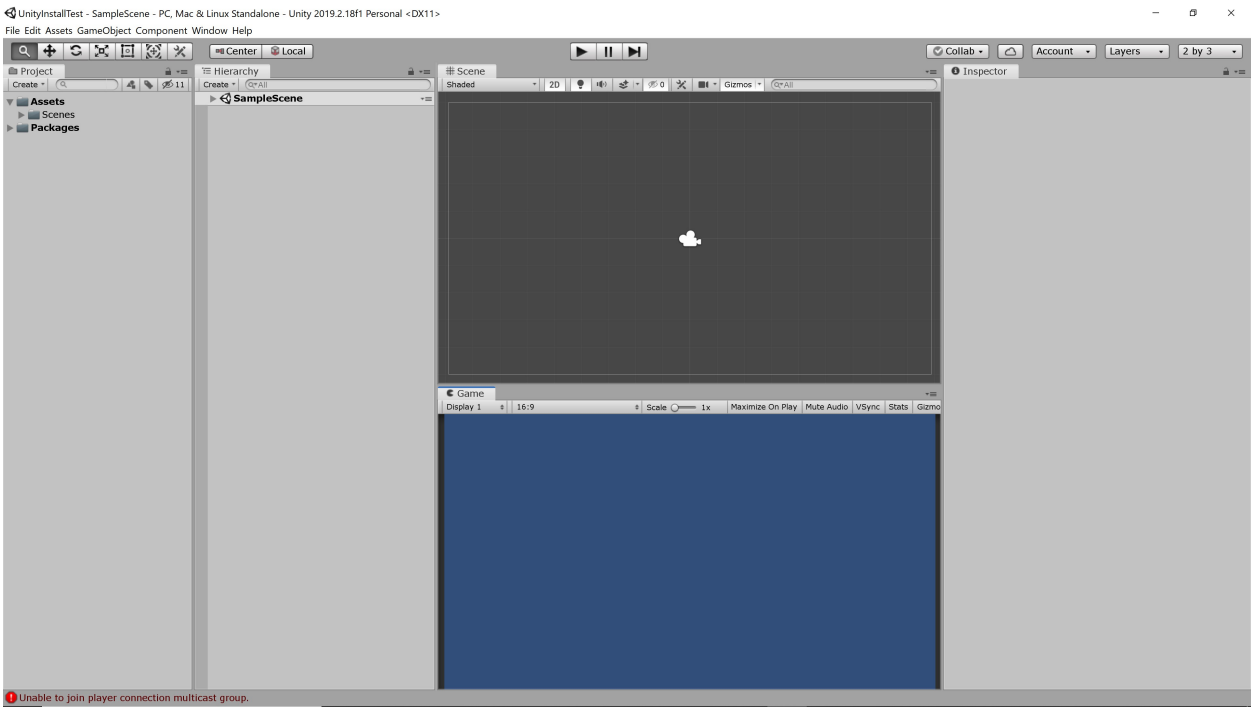


We'll use the 2 by 3 layout as our starting point, so select 2 by 3 from the layout dropdown on the upper right. Left click on the options selector on the Project window (circled in red in the figure below) and select One Column Layout.



We can drag windows around in the Unity by holding the left mouse button down on the tab for the window, then dragging the window to the desired location. This might take a little practice to make sure the window is "docked" where you want it rather than just floating free, but you should be able to get the hang of this pretty easily.

Click the dropdown that says Free Aspect at the top of the Game view and select 16:9. Move the windows around until your setup looks like the image below. As is typical in many apps, you can move the borders on particular windows by dragging their edges.



The last thing we want to add to our layout is a Console window (which acts much like the command prompt window in our console apps). Select Window > General > Console from the menu bar at the top of the Unity window and drag the resulting window onto the bottom of the Project window. After doing that, you'll need to drag the Hierarchy window onto the right side of the Project window so they appear side-by-side above the Console window.

Finally, click the layout dropdown (which currently shows 2 by 3) and select Save Layout... Call the layout whatever you want (we called ours Dr. T) and click the Save button. Now you can use this layout whenever you want by selecting it from the layout dropdown after you create a new project.

Before we finish, let's make sure Unity is set to use Visual Studio Community 2019 as our script editor. Select Edit > Preferences ... from the top menu bar, then click External Tools in the pane on the left. Click the dropdown box next to External Script Editor; if Visual Studio 2019 is an option, select it and close the dialog. If Visual Studio 2019 isn't an option in the dropdown, select Browse... Navigate to where your devenv.exe file for Visual Studio Community 2019 is installed and double-click it. Close the dialog.

We'll actually add and test our first C# script in Unity in Chapter 2, so you're done setting up your development environment.

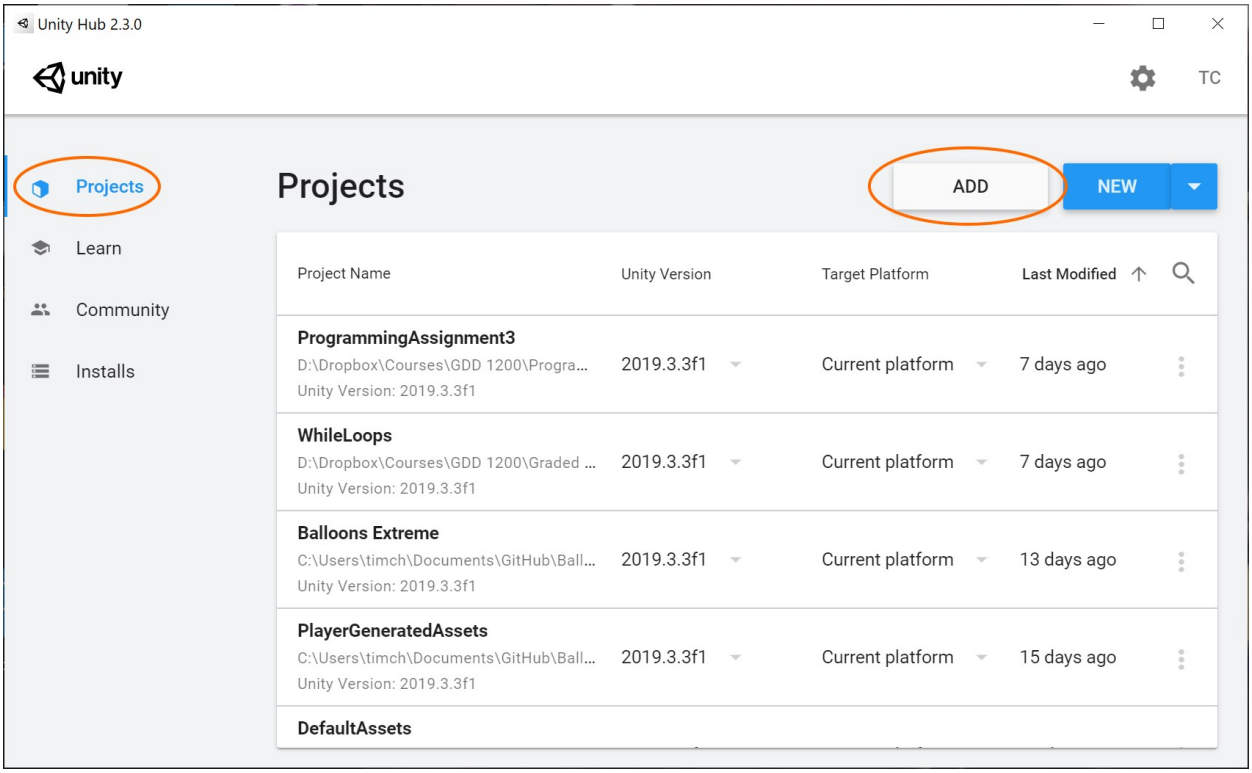
Opening Downloaded Unity Projects

Disclaimer: Although projects built in earlier Unity versions will theoretically work in later versions, we can only guarantee that all the code provided on the Burning Teddy web site will work on version 5.6.0.

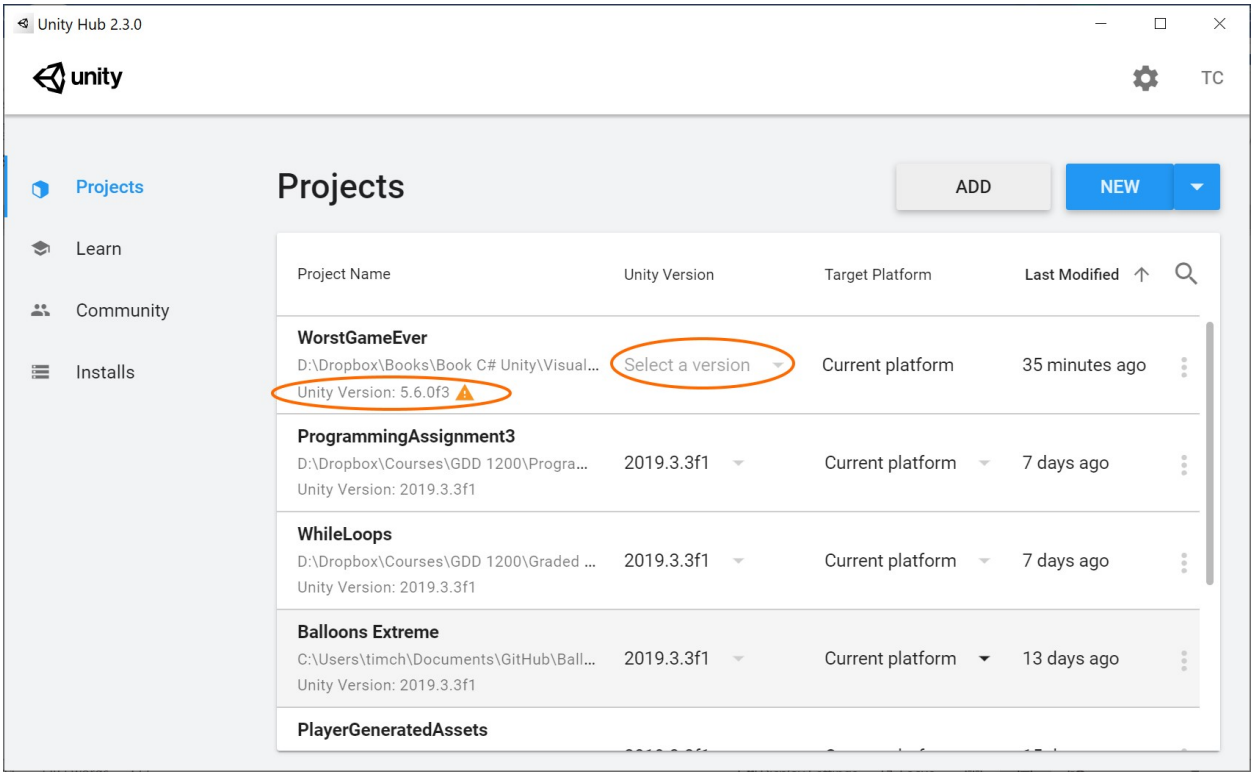
When you download a Unity project, you need to add it to your projects in Unity Hub and (probably) update the Unity version as well. Let's see how that works.

Download the Chapter 6 code from the Burning Teddy web site and extract the zip file somewhere on your computer.

In Unity Hub, select Projects on the left and click the Add button.



Navigate to the folder called WorstGameEver (it will be wherever you extracted the Chapter 6 code zip file) and click the Select Folder button. Unless you happen to have the exact same version of Unity installed as I used to create the project, your WorstGameEver project will look like the image below. Notice that the version I used to create the project is shown below the project name and Unity Hub is asking you to Select a version of Unity to use for the project.



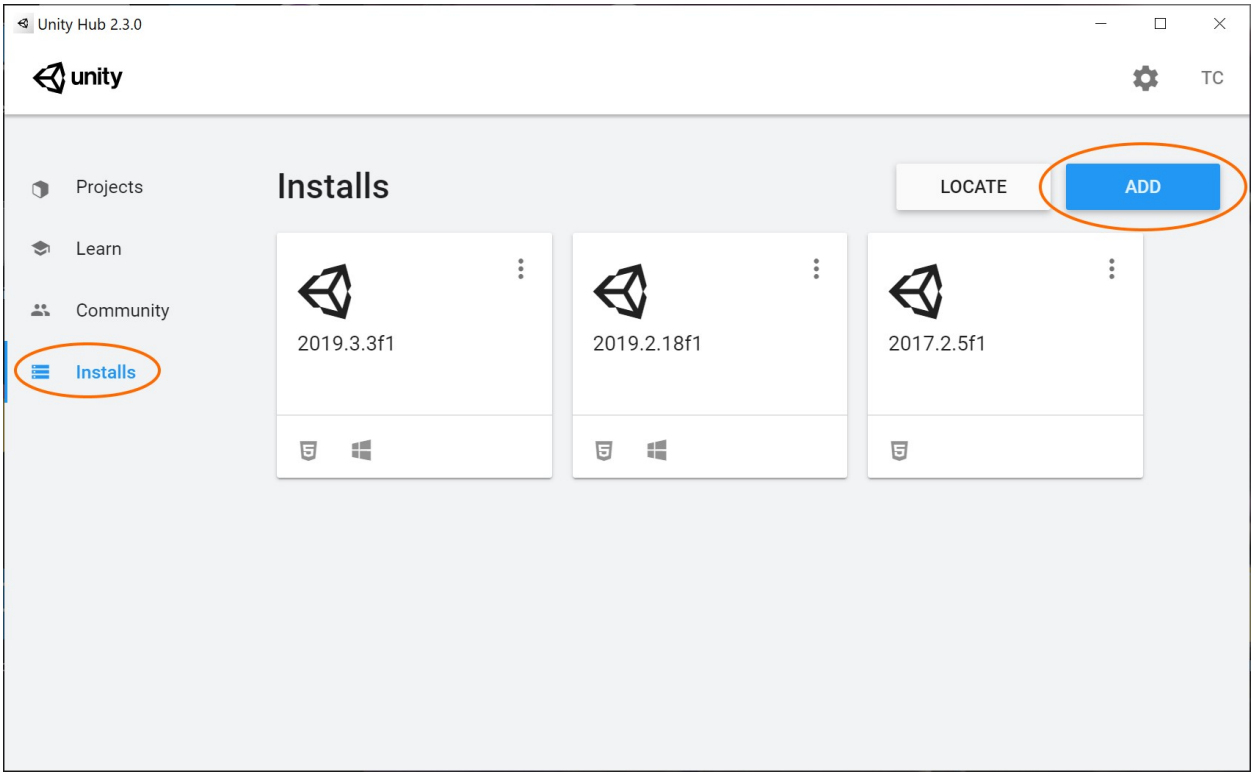
Click the down arrow next to Select a version to select the Unity version you want to use for the project; the list of versions that are shown are the Unity versions you currently have installed on your computer.

To open the project, click the project name in Unity Hub. Unity will ask you if you want to upgrade the project; you should click Confirm and Yes buttons in the dialog boxes that appear as necessary to complete the upgrade.

Installing New Unity Versions

Unity releases new versions pretty regularly; you don't have to install each new version, but you'll probably want to install the latest version of Unity periodically. Here's how you do that.

Open Unity Hub, click Installs on the left and click the Add button on the upper right.



Select the Unity version you want to install from the dialog box that appears (I usually pick the first choice in the Latest Official Releases section) and click the Next button. Check the modules you want to be installed; I usually check WebGL Build Support, Windows Build Support (IL2CPP) and Documentation (to get local documentation installed). Click the Done button, confirm that Unity Hub can make changes to your computer (if necessary), and wait patiently while your selected version is installed.