# Appendix A: 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, Windows users will need to install three pieces of software: Visual Studio Community 2015, the Windows 10 SDK, and MonoGame 3.4. We'll discuss how to install each of them below.

Mac users should follow the instructions in Section A.4. instead of the Windows installation instructions in Sections A.1. through A.3.

Linux users should follow the installation instructions here to install MonoDevelop and MonoGame 3.4.

## A.1. Visual Studio Community 2015

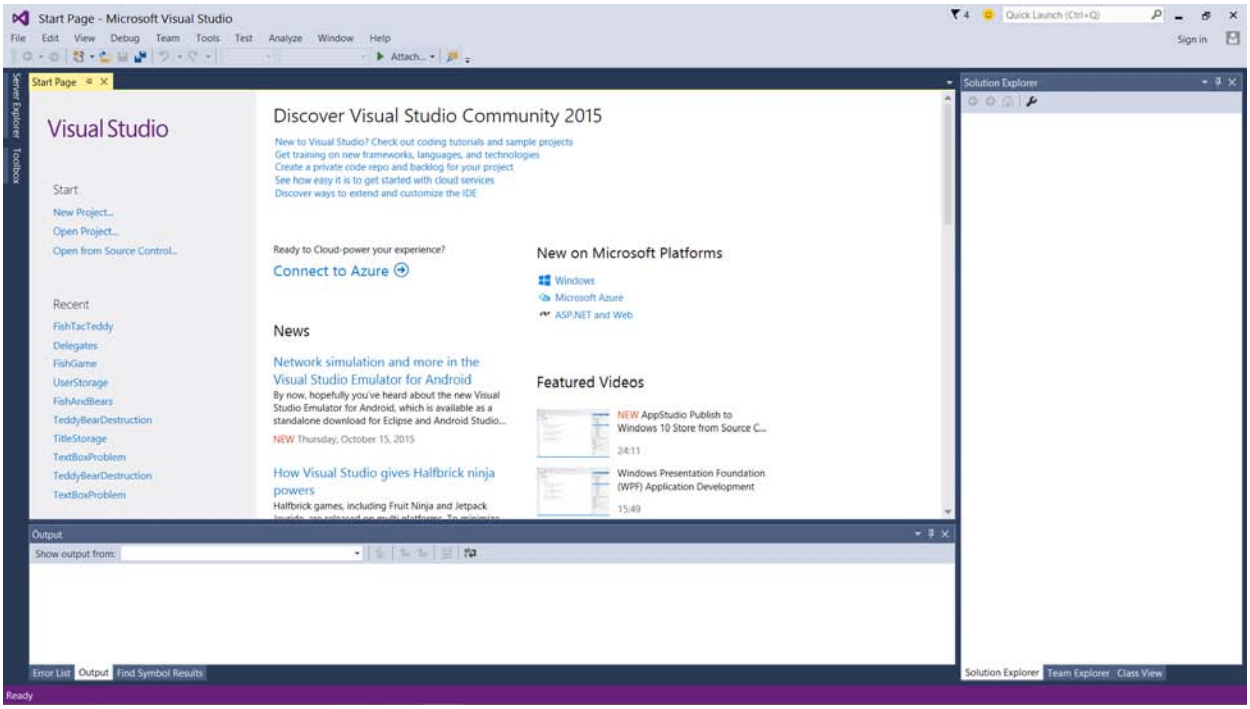
Microsoft's commercial Integrated Development Environments (IDEs) are called Visual Studio, and they come in various capability (and cost) levels. Because there are lots of people who want to program as a hobby rather than as a commercial endeavor, Microsoft also distributes free IDEs. We used the free Visual Studio Community 2015 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 2015 you can just open up and use the projects as is.

Here's how you can get this IDE installed:

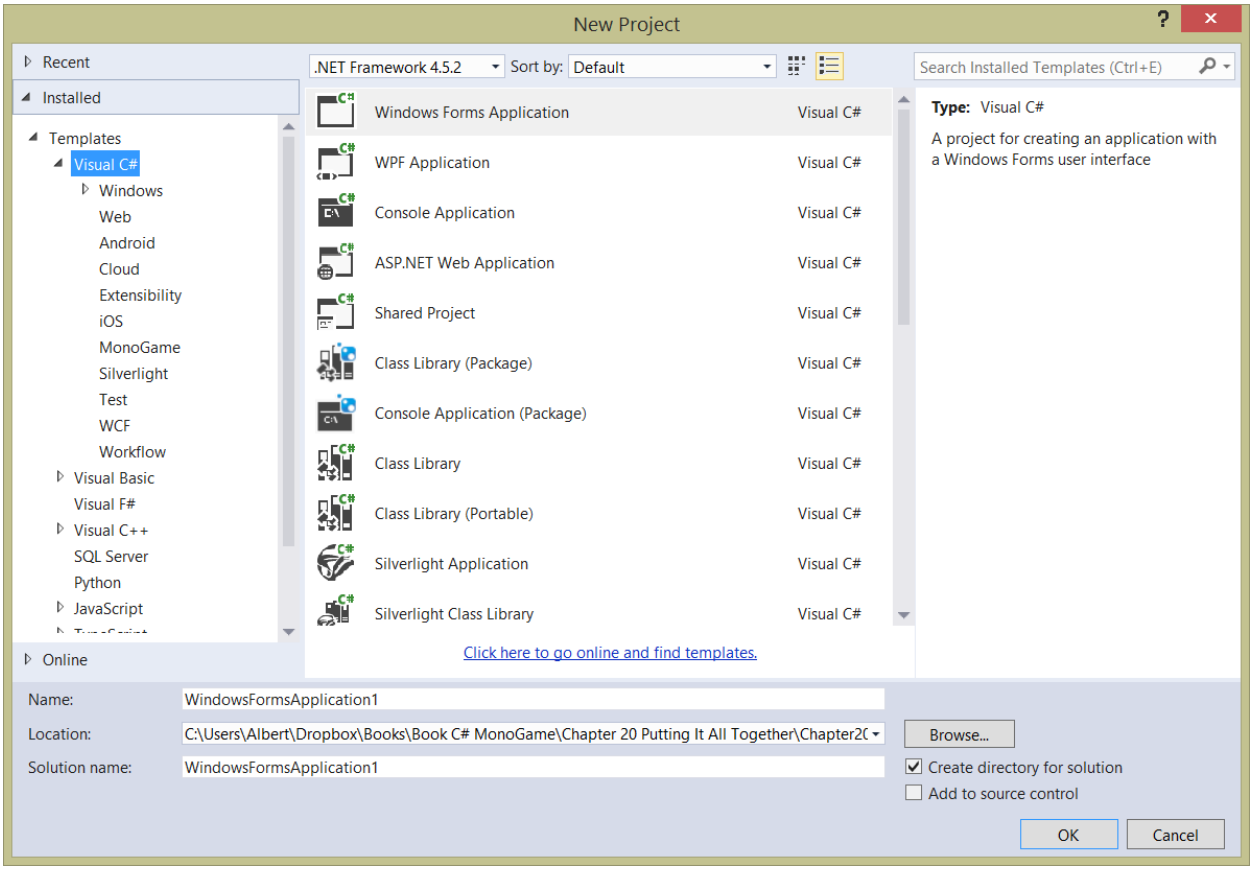
1. Download Visual Studio Community 2015 here.
2. Run the install file you downloaded. Just accepting all the default values as you go along should work fine, but you can of course change them if you want.

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

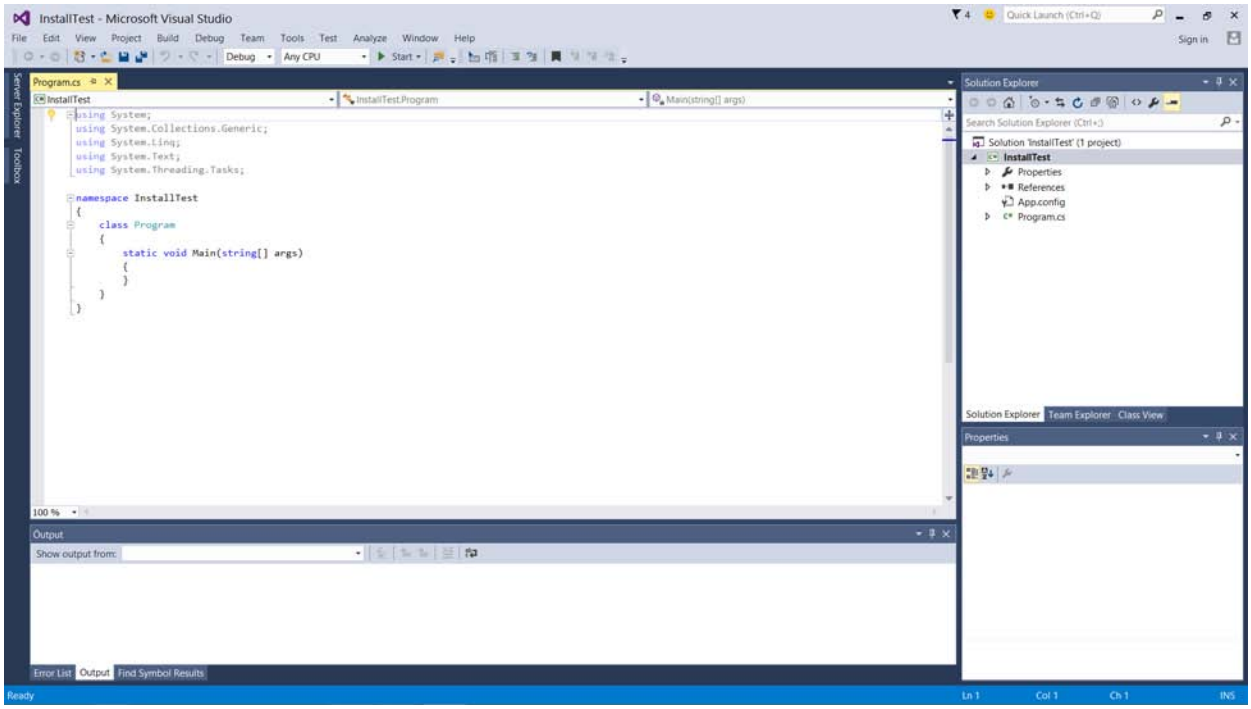
We put a shortcut on our desktop to make launching the IDE easier; for us, the IDE is installed at C:\Program Files (x86)\Microsoft Visual Studio 14.0\Common7\IDE\devenv.exe Of course, you can also use Windows search on Visual Studio and click the Visual Studio 2015 result instead if you prefer. When you start up the IDE, you should end up with something like the window below. Yours won't match exactly, and you definitely won't have a long list of recent projects, but it should be similar.



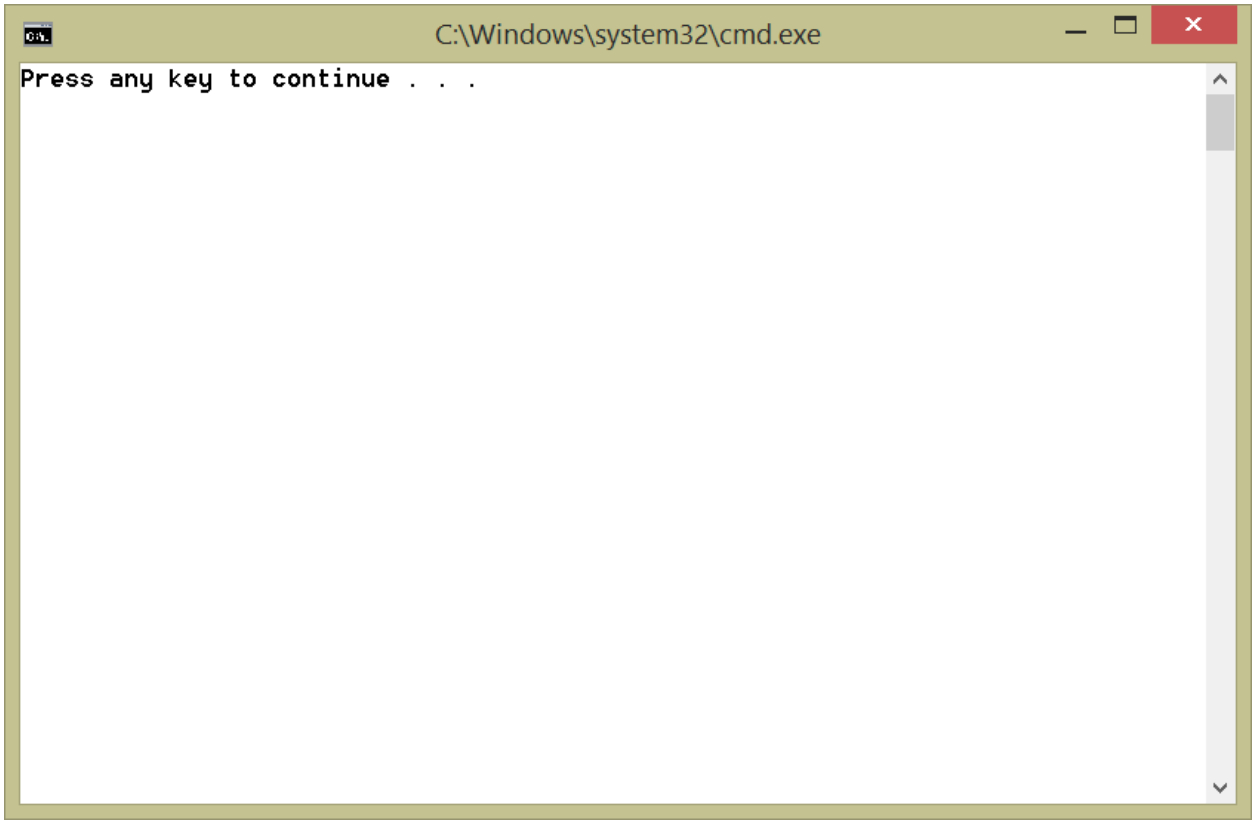

We're going to compile and run a simple program to make sure the IDE installed correctly. Click New Project …, which will give a popup screen like the one below. You may need to click Visual C# in the Installed Templates pane to make the middle pane look like the one shown below.




Click Visual C# in the Installed Templates pane on the left, then click the Console Application icon in the Templates pane (it's the third one down in the figure above), change the Name from ConsoleApplication1 to InstallTest in the text box at the bottom of the screen, change the Location to put the project wherever you want it to be saved, then click the OK button. After a moment of two, you'll come back to the main screen of the IDE, which should look like this:




When we told the IDE we wanted a new console application project, it automatically gave us a template for that kind of application. Compile the application by pressing F7. Once you see a "Build succeeded" message in the lower left corner of the screen, you can actually run the application. Do that by pressing Ctrl-F5; when you do, you should get the popup window below. Just press any key to return to the IDE.




Your 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 on the tan bar at the top of the window, select Defaults, click the Colors tab, and change away.

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 2015 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 you 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.

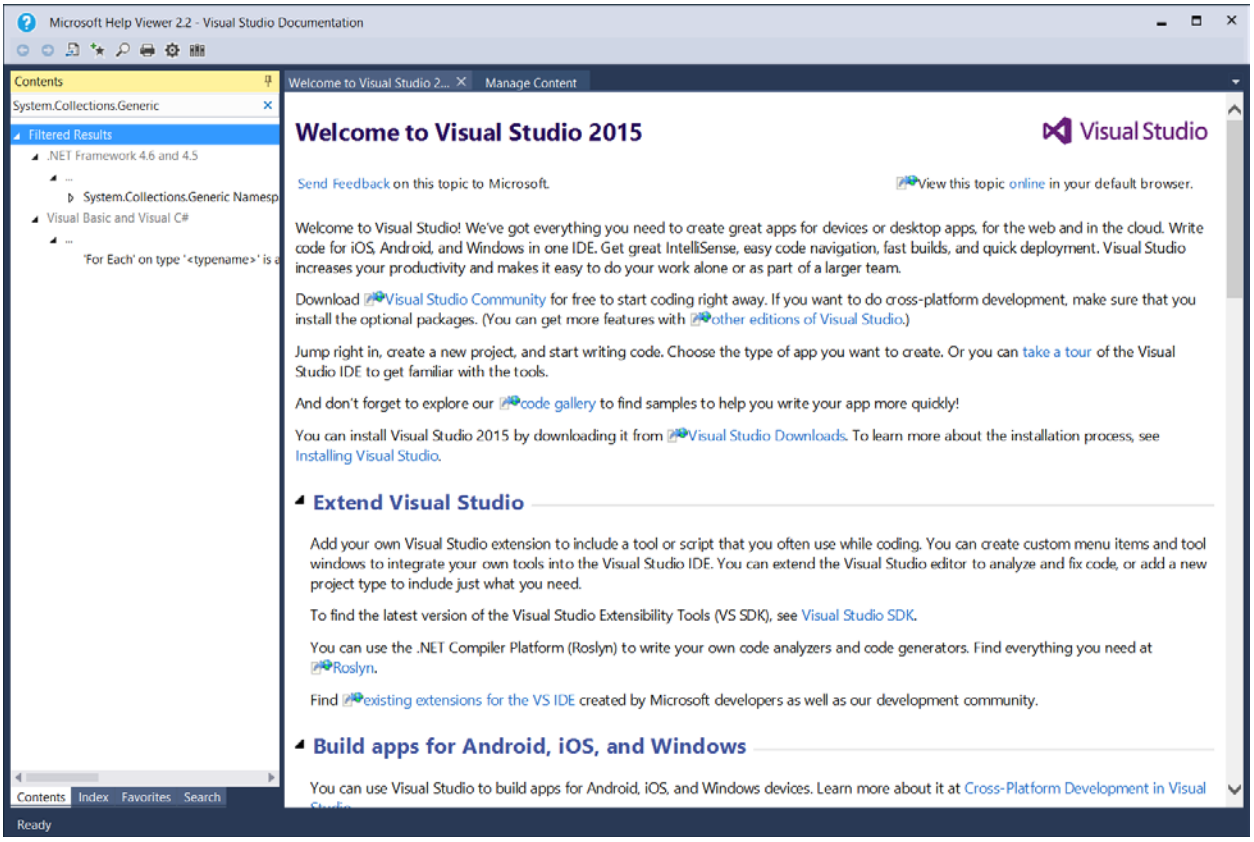
Start up the IDE, then click Help > Add and Remove Help Content. The IDE starts up the Microsoft Help Viewer, which lets you add local content. At a minimum, you should add the following topics:

.NET Framework 4.6 and 4.5 (already set to be added by default)
Visual Studio 2015: Fundamentals (already set to be added by default)
Visual Studio 2015: Visual Basic and Visual C# (click the Add link to add)

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

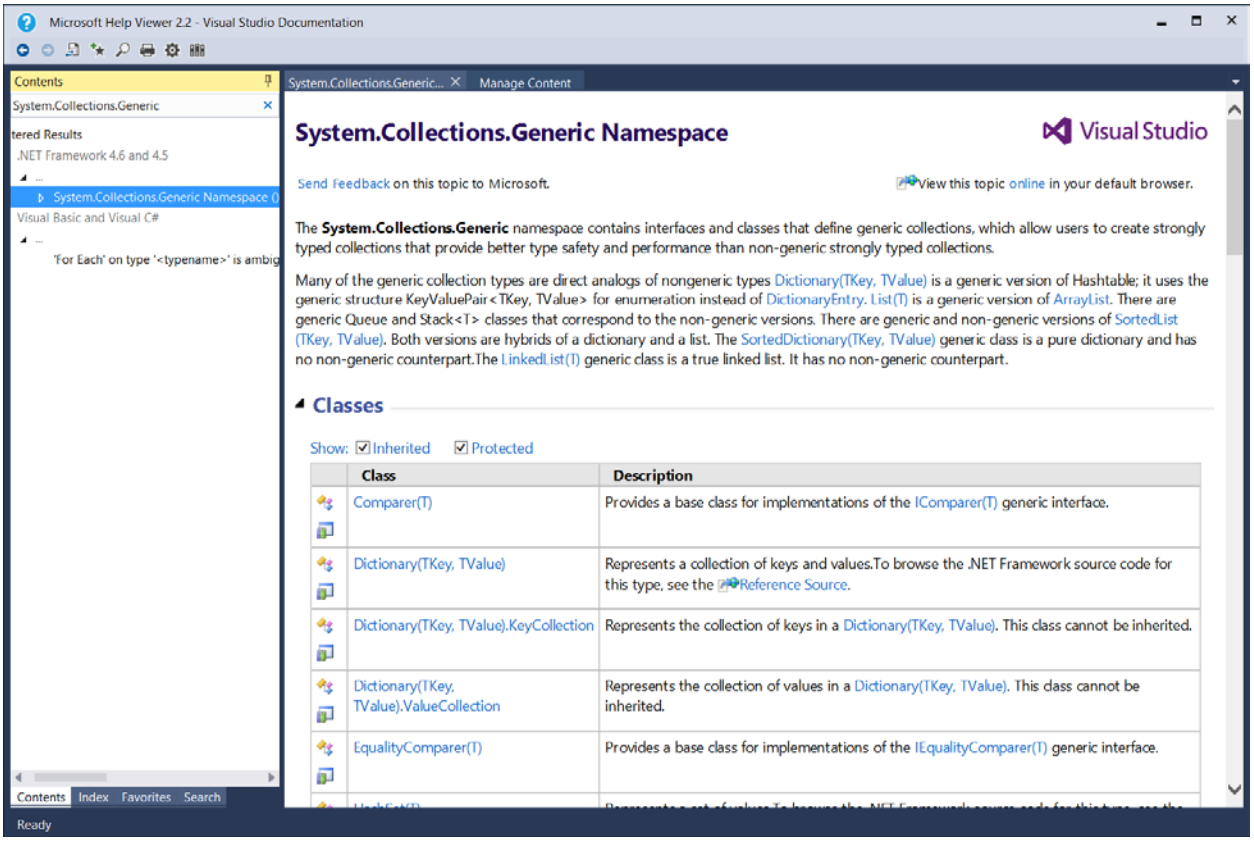
Finally, you can read the help documentation through a browser (which will access the online documentation) or through the Microsoft Help Viewer (which will access the local help content). Both approaches provide access to the same help content, they just look slightly different – and, of course, if 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 (or click the Search button). 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.




The above steps get local help installed for C#, but they don't get local help installed for the XNA Framework code we'll be using for our MonoGame games. Let's take care of that now (remember, you can always just use the online help if you don't want to do the steps below).

1. Download the XNA Documentation zip file from the Burning Teddy web site.
2. Extract the contents of the downloaded file.
3. Start up the IDE, then click Help > Add and Remove Help Content.
4. For Installation source ... near the top of the pane on the right, select the Disk radio button.
5. Click the box with three dots in it to the right of the text box that shows the disk location, navigate to where you extracted the downloaded file in Step 2, select the .msha file, and click the Open button.
6. Click the Add link next to XNA.
7. Click the Update button near the lower right of the popup and wait for the documentation to be installed.

If that all worked properly, we're ready to move on to installing the Windows 10 SDK.

## A.2. Windows 10 SDK

MonoGame 3.4 (which we'll discuss and install in the next section) supports developing Windows 10 Universal Apps. To use MonoGame 3.4, we need to install the Windows 10 SDK; that's what we're doing in this section.

Here's how you can get the Windows 10 SDK installed:

1. Download the Windows 10 SDK here.
2. Run the install file you downloaded. Just accepting all the default values as you go along should work fine, but you can of course change them if you want.

## A.3. MonoGame 3.4

In 2006, Microsoft released XNA Game Studio Express, a game development framework to make it easier for students and hobbyists to develop games. Microsoft developed and released a number of versions of the framework over the subsequent years; the final version, XNA Game Studio 4.0 Refresh, was released in 2011. In January 2013, Microsoft announced they would no longer be actively developing new versions of the XNA framework.

Starting in 2011, the MonoGame open source project has been developing support for the complete Microsoft XNA 4 framework. Almost all of that framework is implemented in MonoGame 3.4. As the MonoGame web site notes, Microsoft Studios actually released several Windows 8 and Windows 8 Phone games in 2013 using MonoGame!

The XNA 4 framework supported building games for Windows 8, Windows Phone 8, and the Xbox 360. In contrast, MonoGame 3.4 supports development for iOS, Android, MacOS, Linux, Windows, Windows Store, Windows Phone, Xbox 360, PlayStation 4, and Ouya (with support for PlayStation Mobile, Xbox One, and RaspberryPi in the works). Go here to see some of the games MonoGame developers have published. To make a long story short (too late), MonoGame 3.4. is a great industrial-strength cross-platform game development framework.

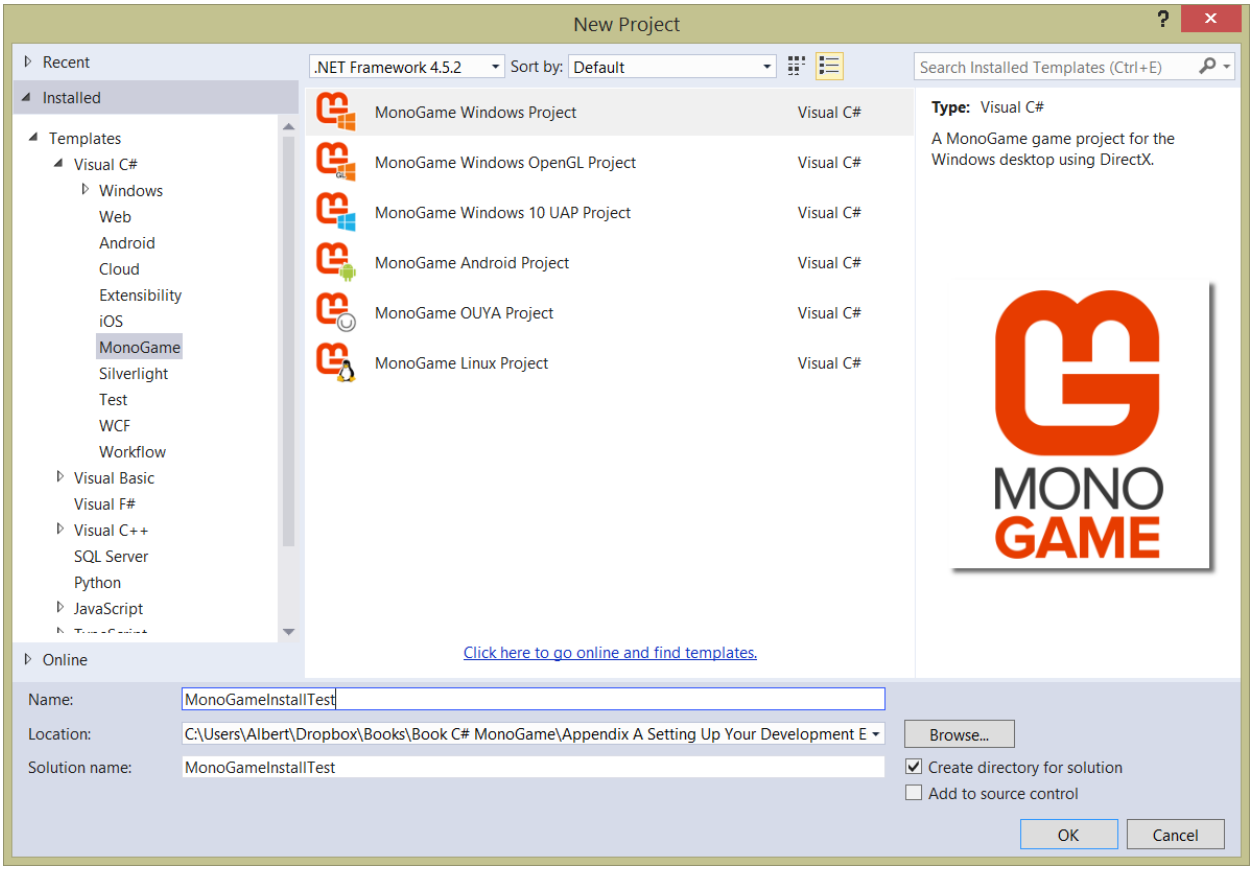
Here's how you can get MonoGame 3.4 installed:

1. Download MonoGame 3.4 here.
2. Run the install file you downloaded. Just accepting all the default values as you go along should work fine, but you can of course change them if you want.

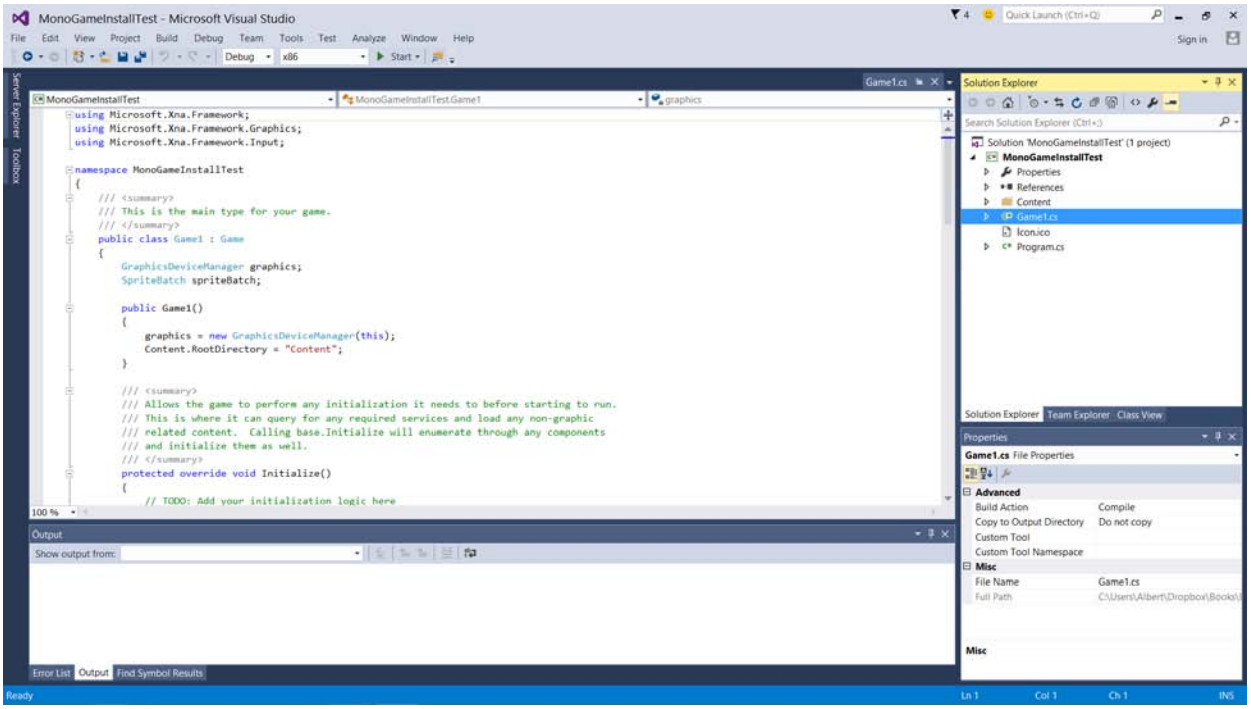
When MonoGame 3.4 installs, it checks for compatible versions of the Visual Studio IDEs, so there's nothing special you need to do during the install. Let's check to make sure the install went okay.

Start up the IDE, click New Project …, and click on the MonoGame subfolder under Visual C# in the Installed Templates area of the pane on the left. When you do, take a look at the templates listed in the Templates pane in the middle. Instead of the Visual C# installed templates you saw before, you'll see the templates that were installed when you installed MonoGame.

Click the MonoGame Windows Project icon, change the Name to MonoGameInstallTest, set the location where you'd like the project to be saved, and click OK.




After a moment of two, you'll come back to the main screen of the IDE. Left click the Game1.cs file in the upper right pane; your IDE should now look like this:




When we told the IDE we wanted a new MonoGame Windows Project, it automatically gave us a template for a Windows game. If you scroll through the code, you'll see that the Windows game template has a LOT more code than the console application template had.

Compile the application by pressing F7. Once you see a "Build succeeded" message in the lower left corner of the screen, you can actually run the application. Do that by pressing F5; when you do, you should get a cornflower blue window. Just click the X in the upper right corner of the window or press the <Esc> key on your keyboard to return to the IDE. For MonoGame games, we'll use F5 to run the program instead of Ctrl-F5 like we did for the console application. If (when!) you write programs that "crash" while they're running, using F5 will take you directly into the debugger so you can work on fixing your code.

If everything worked as described here, MonoGame is installed and you're ready to go. If it isn't working, you should try working through the install process again or get online to ask for help.

## A.4. Mac Users

The IDE you'll be using to develop your games is Xamarin Studio. Here's how you can get Xamarin Studio installed:

1. Download Xamarin Studio here.
2. Run the .dmg file you downloaded by double-clicking it. Select Install Xamarin and wait for the installation to complete. You don't need to install Xamarin.Android or Xamarin.iOS for the topics in this book, you can just install Xamarin.Mac.

Next, you need to install MonoGame 3.4. Here's how you can get MonoGame 3.4 installed:

1. Download MonoGame 3.4 here.
2. Before running the installer package, make sure your computer has a /usr/local/bin/ directory; if it doesn't, create one! This issue may be fixed in newer MonoGame 3.4 MacOS installers, but we needed to do this step for our installation to work.
3. Run the .pkg file you downloaded by double-clicking it. You may get a security alert that says the .pkg file can't be opened because it's from an unidentified developer. If that happens, right click the file instead and click Open. Work through the installation steps.