Getting Started with Unity 5

This book will guide you through the entire process of creating a 3D game, from downloading the Unity game engine to publishing your game.

Beginning with an overview of the Unity engine and its interface, you will walk through the process of creating a game environment and learn how to use built-in assets as well as assets created with third-party 3D modeling tools such as Blender.

Moving on, you will create your very own animation clips from within Unity and learn scripting in Unity. You will master exciting concepts including mini-mapping, the game navigation system, sound effects, shadows, and light effects. By the end of the book, you’ll have learned advanced topics such as cross-platform considerations that enable your games to run on every platform.

Who this book is written for

If you are a game developer interested in learning Unity 3D from scratch and becoming familiar with its core features, then this book is for you. No prior knowledge of Unity 3D is required.

What you will learn from this book

- Create, organize, and manage your game project with the Unity interface
- Develop a 3D game environment with a custom terrain, water, sky, mountains, and trees
- Import and use custom assets and asset packages to add characters to your game
- Review existing animations and create custom animation clips to bring your game characters to life
- Build custom scripts to make your game characters interactive
- Add a graphical user interface to your game to enable easy user interaction
- Explore advanced Unity concepts including workflow, scaling, physics, and cross-platform considerations
- Customize your game with sound effects, shadows, lighting effects, and rendering options

Leverage the power of Unity 5 to create amazing 3D games

Dr. Edward Lavieri

In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'Getting Jiggy with the Unity Interface'
- A synopsis of the book’s content
- More information on Getting Started with Unity 5

About the Author

**Dr. Edward Lavieri** is a veteran game designer and developer with a strong academic background. He earned a doctorate of computer science from Colorado Technical University, and three masters of science degrees in management information systems from Bowie State University. His formal education includes instructional design (from Capella University) and operations management (from the University of Arkansas), demonstrating his passion for academic pursuits. He has developed and taught computer-related courses since 2002 and currently teaches at Southern New Hampshire University. Edward retired from the US Navy after 25 years as an intelligence specialist and command master chief.

As the founder and creative director of three19, a software design and development studio, Edward is constantly developing software. He uses Unity as one of his primary game development tools. He focuses on developing adaptive learning systems, educational games, and mobile apps.

Getting Started with Unity 5

With the pervasiveness of games and the use of gamification in nearly every industry, the desire to discover how to use state of the art development software has never been so great. There is an increasing number of software tools available to help developers create amazing games for consoles, the Web, desktop computers, and mobile devices. Game engines are among the most powerful of these tools available. The Unity 3D game engine is one of the elite game engines. It has been used to create popular 2D and 3D games by large game studios and indie developers. With a free version available, and the release of Unity 5, the time has never been better to start using Unity.

Getting Started with Unity 5 covers one of the most popular game engines available. This book will guide you through the entire process of creating a 3D game, from downloading the Unity game engine to publishing your game. You will enjoy having complete coverage of exciting topics including player-controlled characters and animation. Whether you are just getting started as a game developer or have experience with Unity or other game engines, this book will provide you with a guided tour of developing games with Unity 5. With clear explanations, tips, and ample screenshots, you will be provided with detailed steps to develop your game.

This book takes a practice hands-on approach to learning Unity 5. As you progress through each chapter, you will build a 3D interactive game called Little Farmer Colt. As you create the game, you'll learn key features of Unity 5 including creating a game environment, animating characters, scripting, and more. All meshes, models, textures, animations, and other assets are available on the book's website.

By the time you complete the lessons in this book, you'll have full confidence to start using Unity 5 to create your own games.

What This Book Covers

Chapter 1, Getting Jiggy with the Unity Interface, surveys the game engine landscape, comparing other game engines with Unity to give you a full appreciation for Unity's capabilities. You'll learn how to download and install Unity 5. You'll also learn about Unity 5's user interface and core tools, including transform tools and cameras.

Chapter 2, Creating the Game Environment, will teach you about game design by reviewing the book's game design of Little Farmer Colt. You'll start creating the game environment with terrain, mountains, trees, a lake, and the sky.

Chapter 3, Working with Assets, is all about the assets used to develop games with Unity 5. You'll learn about assets, where to get them, how to make your own, and how to
import them into your game. You'll also learn about free 3D object creation software to create game objects compatible with Unity 5.

*Chapter 4, Animating the Game Characters*, is about Unity animations and player controllers. You'll see our game characters come to life and give your users the ability to control the player.

*Chapter 5, Scripting the Game*, is dedicated to teaching you how to script. You'll start with a primer on C# and experience with MonoDevelop. You'll also gain experience creating your own C# scripts for your game.

*Chapter 6, Adding a Graphical User Interface*, contains key information on graphical user interfaces and their importance to games. You'll gain hands-on experience with the Unity 5 user interface system. You'll learn to create a heads-up display, an in-game mini-map, and full-screen navigation menus.

*Chapter 7, Polishing and Optimizing the Game*, will teach you several ways to make your game shine and operate optimally. You'll gain appreciation for the importance of audio and visual effects in games and how to create them. You'll gain experience with sound effects, shadows, lighting effects, and camera rendering. You'll also learn about optimizing scripts.

*Chapter 8, What's Next?*, is the final chapter where you'll receive several specific suggestions on how to take the *Little Farmer Colt* game to a higher level. You'll also learn about Unity workflow, project scalability, cross-platform issues, and the importance of attribution. You'll also gain an appreciation for advanced topics.
The purpose of this chapter is to familiarize you with Unity and its interface. We'll start with a discussion of game engines to see how Unity stacks up. You'll be guided through the download and installation process. We'll then discuss the Unity project and its file structure in and out of Unity.

We'll also take a look at the Unity interface. We'll experiment with different layouts to give you an idea of the different workflows you can set up. I'll explain each view in Unity and discuss the purpose of each one. We'll spend extra time in the Scene view using the transform tools, so that you'll be comfortable using them. This chapter will end with a discussion on how cameras are used in Unity.

Through this chapter, you will:

- Understand the Unity engine
- Be able to download and install Unity
- Understand the Unity projects
- Be comfortable with the Unity layouts
- Understand the purpose of each view
- Be able to navigate using the Scene view
- Be familiar with the transform tools
- Be familiar with Cameras
Why Unity?
There are several dozen game engines available for 2D and 3D game development. So why choose Unity? I probably do not have to convince you of that because you are reading this book. In case you still need convincing, let me tell you about some of the competition, first.

Unreal and CryEngine are among the most capable game engines available. Large game studios have created 3D games using these game engines for many years. Both engines have been used to create high-grossing games. The pricing models and complexities of using them are enough to make indie game developers and small game development studios look the other way.

What about easy-to-use game engines? Game Salad, Game Maker, and Construct 2 are all easy to learn and use. In fact, you can create simple games with these engines without having to program or script a single line of code. These are all 2D game development engines and lack the capabilities of the larger engines.

This does not put Unity in the middle of the easy-to-use and most capable game engines. Along the spectrum of capabilities, Unity is to the far right. See the following diagram:

Let's review the following table so that these game engines can be compared based on their capabilities, learning curve, and what programming languages are used to develop games with each engine:

<table>
<thead>
<tr>
<th>Game Engine</th>
<th>Learning Curve</th>
<th>Language</th>
<th>2D/3D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unity</td>
<td>4 of 5</td>
<td>C#, JS, Boo</td>
<td>Both</td>
</tr>
<tr>
<td>Unreal</td>
<td>5 of 5</td>
<td>C++</td>
<td>3D</td>
</tr>
<tr>
<td>CryEngine</td>
<td>5 of 5</td>
<td>C++, Lua</td>
<td>Both</td>
</tr>
<tr>
<td>Game Salad</td>
<td>1 of 5</td>
<td>GML</td>
<td>2D</td>
</tr>
<tr>
<td>Game Maker</td>
<td>1 of 5</td>
<td>N/A</td>
<td>2D</td>
</tr>
<tr>
<td>Construct 2</td>
<td>2 of 5</td>
<td>JS</td>
<td>2D</td>
</tr>
</tbody>
</table>
The Unity game engine sits in a sweet spot between tremendous capabilities and difficulty to learn. This makes it the engine of choice for many developers. As you'll see later in this chapter, we declare either 2D or 3D when we create a game project.

There are some other beneficial reasons for using Unity over the other game engines:

- The ability to program with C#, JavaScript, Boo, or any combination of these languages in the same game.

  ![Note](image)

  While Unity supports scripts in the different supported languages (C#, JavaScript, and Boo) in the same game, I do not recommend this. The best practice is to pick a single programming language and use it throughout the game.

- The ability to test play games in a separate view (window) without having to create builds or leave the development interface.

- The ability to make changes to the game while it is being played and not having those changes impact on the saved version. This is a great way to experiment and test.

- The ability to develop once and then deploy to mobile (iOS, Android, Windows Phone 8, and BlackBerry 10), desktop (Mac, Windows, and Linux), Web (Safari, Firefox, Chrome, and Internet Explorer), and console devices (Xbox One, Xbox 360, PlayStation 3, PlayStation 4, PlayStation Vita, and Wii U).

In addition to these great reasons to select Unity as your game engine of choice, it is free for most purposes. There is a Pro version that costs money, but if you want to get started without having to spend any money, Unity is for you.

### Getting your hands on Unity

If you already have Unity installed on your computer, you can skip this section. Otherwise, you can follow the steps in this section to download and install Unity on your computer:

2. You see a Download link at the top-right section of the web page. Click that link to go to Unity's download page, as shown in the following screenshot:
3. On the download page, you'll see a **Download Unity** button that includes the current version of the engine. For example, the button might say **Download Unity 5.1**. The web page will know what operating system your computer has and will download the correct version. Click on the **Download** button. The installation file is quite large, well over 1GB, and could take several minutes to download, especially if you have a slow Internet connection.

4. Once the download completes, simply double-click on the **installation file** and let the installer do the work. Accept any defaults if prompted. Unity is an industrial strength game engine, so do not be alarmed if the installation process takes a while.

5. Now you are ready to launch Unity for the first time. Double-click on the **Unity icon**.

**Projects**

The first thing we must do to use Unity is to create a project. Just like word processing software using a document to create a letter, Unity uses projects to create games. Projects are the enveloping structure for all the scripts, game objects, art, and other source files that are used in a game. You'll see later in this chapter how the external file structure of project files is replicated in Unity.

When Unity is launched, the first screen you should see is the welcome screen window, as shown in the following screenshot:
From this dialog window, you can easily open projects that you’ve recently worked on; they’ll be listed on the left under the Projects link. You can also open a new project by clicking on the New project button in the upper-right section of the window.

Let’s walk through creating a new project. Our first step is to click on the New project button. This brings up the new project dialog window, as shown in the following screenshot:

There are four decisions you need to make on this screen before you click on the Create project button. First, you’ll need to give your project a name. This is like naming a file, and I recommend including the version number. For example, if you are working on the first version of a new game called Java For Everyone, you might name your project Java4Everyone 1.0, java_for_everyone_1.0, or something similar.

The next step is to tell Unity where you want the project saved. You can accept the default or click on the eclipse icon to open a File Explorer window. Using this window, select where you want your project to be saved. I recommend selecting a folder or directory that is automatically backed up or linked to a cloud-based file system. This will help to ensure that you do not lose your work.
Next, we need to indicate whether we are creating a 2D or 3D project. If you are creating a platform or side-scrolling game, you'll select 2D. Otherwise, you'll select 3D. You can certainly have both 2D and 3D components to your game. Unity gives us a lot of flexibility in this regard. So, our initial selection simply indicates our primary focus.

Our last decision to make is what, if any, asset packages we want to import. If you already know what packages you will need, you can have them loaded when the project is created. Alternatively, you can easily import packages after you've started working on your project. This is the method I recommend because it ensures you only load what you absolutely need. We'll discuss how this is done in a later chapter.

Once you've made the four decisions, click on the **Create project** button, as shown in the following screenshot:

![Create project screenshot](image)

Unity will create your project and depending on how many packages you selected to be included, it could take more than a few seconds to create. If the window disappears and you do not see any indication that your project is being created, don't panic. Unity is working in the background and will open the main interface as soon as the project has been created.
If you look at your filesystem, you’ll see that Unity has created a folder with the name of your project. In that folder, you’ll find four subfolders titled Assets, Library, ProjectSettings, and Temp. Unity keeps projects well organized with these primary folders. As new game objects are added to your project, they will be placed in these folders. You can also create additional folders to be even more organized.

<table>
<thead>
<tr>
<th>Documents</th>
<th>Java 4 Everyone 1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Date modified</td>
</tr>
<tr>
<td>Assets</td>
<td>1/27/2015 2:03 PM</td>
</tr>
<tr>
<td>Library</td>
<td>1/27/2015 2:03 PM</td>
</tr>
<tr>
<td>ProjectSettings</td>
<td>1/27/2015 2:03 PM</td>
</tr>
<tr>
<td>Temp</td>
<td>1/27/2015 2:03 PM</td>
</tr>
</tbody>
</table>

It is a good idea not to move files around or change folder names. There is a chance you’ll break linkages inside your project. Instead, make any desired changes within Unity.

**Layouts**

One of the wonderful things about working with Unity is that you can customize the way the user interface is laid out. You can use one of the predefined layouts of 2 by 3, 4 Split, Tall, or Wide, or you can create your own. Layouts refer to how the various views in Unity are arranged on the screen. You’ll learn about views in the next section.

To change a layout, we simply click on the Layout button that is located in the far top-right corner of the Unity interface:
Let's look at each layout to see the differences. The first layout is the 2 by 3 layout. This layout provides a nice arrangement with the Scene and Game views on the left, the Hierarchy and Project views in the middle and a full Inspector view on the right, as shown in the following screenshot:

The 4 Split layout provides four different views of the same scene, as shown in the following screenshot. This is a good way to review how lighting and shading is implemented in your game. We'll talk about lighting and shading later in the book.
The **Tall** layout provides a tall, but not wide view of the **Scene** view with other views located on the right, as shown in the following screenshot:

![Tall Layout Screenshot](image1)

The **Wide** layout provides a wide view of the **Scene** view, with other views located on the bottom and on the right, as shown in the following screenshot:

![Wide Layout Screenshot](image2)
The **Default** layout is a variation of the **Wide** layout. The difference is that with the **Default** layout, the **Hierarchy** view is on the left, as shown in the following screenshot:

You can switch between views anytime you want without it impacting your game. Most Unity developers do not work in just one view. Different views provide different benefits and are appropriate for different tasks. You can also modify any view by dragging any of the borders of a view. If you want to save a view, make any changes to the current view, then select the **Layout** button and select **Save Layout**. You will be prompted for a name.

**Views**

Unity views provide the ability to see, or view, specific components of the project. There are five views in Unity.
The Scene view
Scenes in Unity are the equivalent to levels in a game. You'll have a different scene for every game level. The Scene view is where we put visual objects such as characters, buildings, terrain, and more. We can also move, rotate, and scale these objects in the Scene view by using transform tools, which we'll discuss in the next section. Take a look at the following screenshot:

The Game view
One of the great features of Unity is that we can play our game right in the game engine. This means we do not need to compile the game and play it outside of Unity. If you're already a game developer, you'll appreciate the time saving. The Game view is where the game is played during development.

Any changes made to the game while the game is being played will be reflected in the game but are not saved when gameplay ends. This represents a wonderful way to experiment and not risk breaking scripts or other game components.
Like with most games, Unity games are played from a player's perspective. Typically, a player controls an in-game character. We do not see the game world through the in-game character's eyes. It might seem that way, but we are actually seeing the game world as rendered by a camera or cameras. We'll talk more about this later. For now, think of a camera hovering just above and behind our in-game character's head. This makes it seem like we are seeing through their eyes:

The Hierarchy view

This view displays a hierarchical list of every object in the scene. When you double-click on an object in this view, two things happen. First, the object is selected and viewable in the Scene view. Unity orients the Scene view so that the selected object is in the front and center. The second thing that happens is that the Inspector view is populated to display the selected object's settings and components. You'll gain great exposure to this later in the book. Take a look at the options shown in the following screenshot:
The Project view

The Project view provides us with a file structure of our game components including objects, art, scripts, and so on. The file structure here is what is displayed on your system's file structure. So, if you want to make changes, do it in the Project view, not directly on your computer. You'll notice that this view has two columns. The first column contains the file structure I mentioned as well as Favorites, which will help you to find assets quickly. This is especially handy for large projects.

The second column contains folder contents. If, for example, you click on a folder in the left Project view column, that folder's contents will be displayed in the second column. You can drag objects directly from this view into the Scene view to add objects to your game. I'll walk you through this later in the book. Take a look at the options shown in the following screenshot:
The Inspector view

When objects are selected in another view, their details are revealed in the Inspector view. In many cases, you can make changes to an object directly in the Inspector view. You’ll be doing this a lot throughout the book.

Remember, layouts are made up of views and you have complete control over how the interface is laid out. For the majority of this book, I’ll use the 2 x 3 layout because of its utility. You can use any view you want and will still be able to follow along with the instructions and guides provided in the rest of this book.

Transform tools

Transform tools in Unity allow us to interact with the Scene view, edit terrain, move objects, and make modifications. There are five buttons that make up the transform tools, as shown in the following screenshot; they are located in the top-left corner of the Unity Interface:
The first button is the **Hand** tool or **View Tool** (️). When this tool is selected, our cursor in the **Scene** view turns to a hand. This lets us know what mode we are in. With this tool selected, we can scroll with our mouse to zoom in and out of the scene. If you click on the left mouse button, you are able to pan around the scene. With the right mouse button clicked, you are able to look around based on the current position of your cursor.

If you hold down the **Alt** key on a PC or **Option** key on a Mac and click on the left mouse button, you can orbit around the current area. Pressing that same key and the right mouse button allows you to zoom in and out of the scene.

The second button is the **Translate** tool (️) and is in the shape of a quad arrow. When an object selected and then click on the translate tool, the object will have three gizmos, one for each axis. Clicking and dragging any of these gizmos moves the object along the respective access, as shown in the following screenshot:

![Translate Tool Screenshot]

The third transform tool is the **Rotate** tool (️), which looks like two rotating arrows. This tool allows us to rotate an object along any axis (x, y, or z). Instead of line and arrow gizmos, this tool is instantiated with three colored rings, one for each axis. Clicking a ring and dragging it rotates the object along that axis, as shown in the following screenshot:

![Rotate Tool Screenshot]
The fourth transform tool is the Scale tool ( ), which is represented with line and block gizmos. Like the other transform tools, there is one gizmo for each axis. Clicking and dragging one of these gizmos increases or decreases the object along the selected axis. For example, you can make a cube wider, narrower, taller, or shorter. If you want to maintain aspect ratio, you can click on the center square instead of the red, blue, or green square. Now, when you click-and-drag, your object will grow or shrink in perfect aspect ratio, as shown in the following screenshot:

The final transform tool is the Rect tool ( ) and is represented by a rectangle with intersecting points. The Rect tool can be used to move, resize, and rotate an object in the Scene view. So, this is a versatile tool that also has corresponding properties that you can edit directly using the Inspector view. Take a look at the following screenshot:
Cameras

Cameras render scenes so that the user can view them. Every scene must have at least one camera. In fact, when a new scene is created, Unity creates a camera named Main Camera. As you'll see later in this book, a scene can have multiple cameras. In the Scene view, cameras are indicated with a white camera silhouette, as shown in the following screenshot:

Cameras are game objects and can be edited using transform tools as well as in the Inspector view. We can classify Unity cameras based on their projection. A perspective projection camera renders a scene based on the camera angle, as it exists in the scene.

The other project is orthographic. An orthographic perspective camera renders a scene uniformly without any perspective. This type of camera is commonly used for top-down games and is the default camera used in 2D and Unity's User Interface (UI) system.

When a camera is selected in the Hierarchy view, its frustum is visible in Scene view. A frustum is a geographic shape that looks like a pyramid that has had its top cut off. The top plane is parallel to its base, in other words, the near and far plane.
We'll use cameras in our game later in the book.

There are additional user interface components to Unity that we did not cover. As we need them in the development of our game, they'll be introduced prior to use.

**Summary**

In this chapter, we took a quick look at and compared game engines. We also took a closer look at Unity and reviewed some of its key features. We downloaded and installed Unity so that we're ready to start developing games. We discussed Unity projects and created one in order to become comfortable with the process. To close out the chapter, we walked through the Unity interface including looking at layouts, views, transform tools, and cameras.

In the next chapter, we'll start designing the game we'll be making through this book and you'll learn how to create the game environment.
Where to buy this book

You can buy Getting Started with Unity 5 from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.