Community Experience Distilled

What you need to know about Unity 5

Learn the techniques and explore the new features of Unity 5

Francesco Sapio

[PACKT] PUBLISHING

What you need to know about Unity 5

Learn the techniques and explore the new features of Unity 5

Francesco Sapio

PACKT PUBLISHING

BIRMINGHAM - MUMBAI

What you need to know about Unity 5

First Published: June 2016

Production reference: 1100616

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

www.packtpub.com

About the Author

Francesco Sapio obtained his computer science and control engineering degree from Sapienza University of Rome, Italy, a couple of semesters in advance, scoring summa cum laude. Now he is studying a master of science in engineering in artificial intelligence and robotics.

He is a Unity3D and Unreal expert, a skilled game designer, and an experienced user of the major graphics programs.

Recently, he authored the book *Unity UI Cookbook* (Packt Publishing) that teaches readers how to develop exciting and practical user interfaces for games in Unity. Furthermore, he has also been a reviewer for the following books: *Unity Game Development Scripting* (Packt Publishing) and *Unity 5.x by Example* (Packt Publishing).

Francesco is also a musician and a composer, especially of soundtracks for short films and video games. For several years, he worked as an actor and dancer. He was a guest of honor at the theatre Brancaccio in Rome.

In addition to this, he is a very active person, having volunteered as a children's entertainer at the Associazione Culturale Torraccia in Rome. He also gives private lessons in mathematics and music to high-school and university students.

Finally, Francesco loves math, philosophy, logic, and puzzle solving, but most of all, creating video games—thanks to his passion for game designing and programming.

You can find him at `https://linkedin.com/pub/francesco-sapio/b8/5b/365`.

I'm deeply thankful to my parents for their infinite patience, enthusiasm, and support throughout my life. Moreover, I'm thankful to the rest of my family in particular to my grandparents because they always encouraged me to do better in my life with the Latin expressions "Ad Maiora" and "Per aspera ad astra".

Finally, a huge thanks to all the special people around me whom I love in particular to my girlfriend; I'm grateful for all your help in everything.

About the Reviewer

Lauren S. Ferro is a gamification consultant and designer of game and game-like applications. She has worked, designed, consulted, and implemented strategies for a range of different purposes from professional development, recommendation systems, and educational games. She is an active researcher in the area of gamification, player profiling, and user-centered game design. Lauren runs workshops both for the general public and companies that focus on designing user-cantered games and game-like applications. She is also the developer of the game design resource Gamicards, which is a paper-prototyping tool for both games and game-like experiences that is centred on the users' preferences.

www.PacktPub.com

Support files, eBooks, discount offers, and more

At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books, eBooks, and videos.

`https://www.packtpub.com/books/subscription/packtlib`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print and bookmark content
- On demand and accessible via web browser

Free access for Packt account holders

If you have an account with Packt at `www.PacktPub.com`, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Table of Contents

What you need to know about Unity 5

This eGuide is designed to act as a brief and practical introduction to Unity 5. It is full of practical examples that will get you up and running quickly with the core functions of Unity 5.

We assume that you know a bit about what Unity 5 is, what it does, and why you want to use it, because this eGuide won't give you a history lesson in the background of Unity 5. What this eGuide will give you, however, is a greater understanding of the key basics of Unity 5 so that you have a good idea of how to advance after you've read the guide. We can then point you in the right direction of what to learn next after giving you the basic knowledge to do so.

What you need to know about Unity 5 will:

- Cover the fundamentals and the things you really need to know, rather than niche or specialized areas.
- Assume that you come from a fairly technical background and understand what the technology is and what it broadly does.
- Focus on what things are and how they work.
- Include practical examples to get you up, running, and productive quickly.

Overview

Before we start, let's spend some time on what we are heading toward.

Unity is a game engine that is free and also for commercial use. However, a Pro version exists with some advance features. This engine started to become very popular for the creation of games because of its simplicity. Furthermore, it is possible to create mobile games as well. In fact, Unity can export both for Android or iOS devices.

In Version 4.6, a new UI system was introduced, but it was finalized with Version 5.x. In this recent version, you can experiment with all of its features. Due to its simplicity, making UIs in Unity has never been easier.

You can see more on the `official website`.

An Introduction to Unity

In this introduction guide, we will explore the potential of Unity and show you the basics. Rather than a complete guide, consider this as an introduction to the main topics with the respective references to the official documentation of Unity.

In this first chapter, we will see how to create a new project and move inside the Unity Interface. At the end of this chapter, we will also learn how to transform and parent game objects.

Getting ready

Let's start by downloading **Unity** from the `official website`.

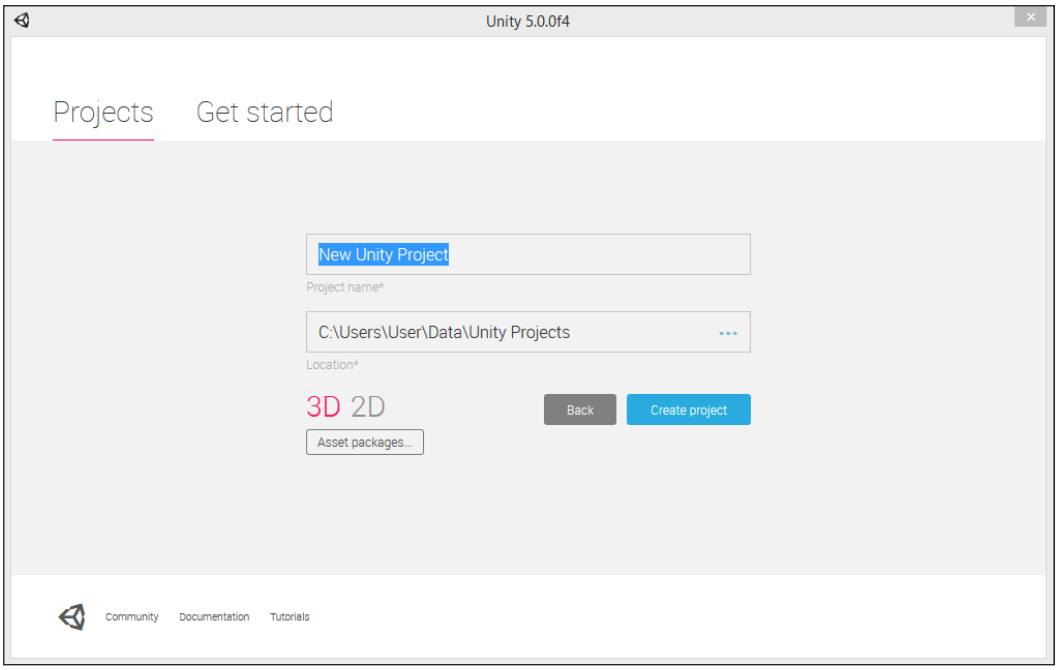
You can install either the free version as well as the Pro version. In this eGuide, we won't use any advanced features that are contained in the Pro version, but feel free to choose the one that you prefer.

If you decide to use the Pro version, you need to know that there are two ways that we can activate an installation: online or manually. Once you download, install, and run the Unity Editor, you will be asked to select the version of Unity that you want. In this case, select Unity Professional.

Manual activation is a slightly more complicated process than online activation, and more details regarding the steps involved can be found at `https://unity3d.com/unity/activation`.

Creating and opening projects

The first time you open Unity, you will see a screen that contains your recent projects, and they can be opened by clicking on one of them. If you want to create a new project, click on **New**, and you will see the following screen:

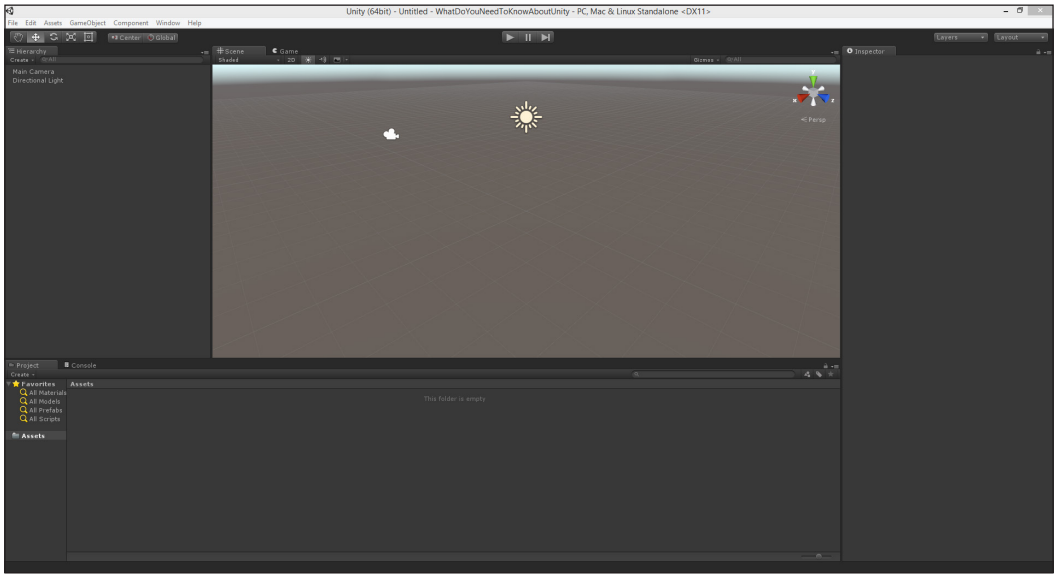


Here, Unity asks you where to store the project on your computer, the name of your project, and whether it is in 2D or 3D. This last choice will change how Unity will import your file because it needs to know how to deal with the files that you want to import. It's good practice to set it here. However, it is always possible to change how assets are imported later on.

To create your project, click on the **Create Project** button.

The interface of Unity

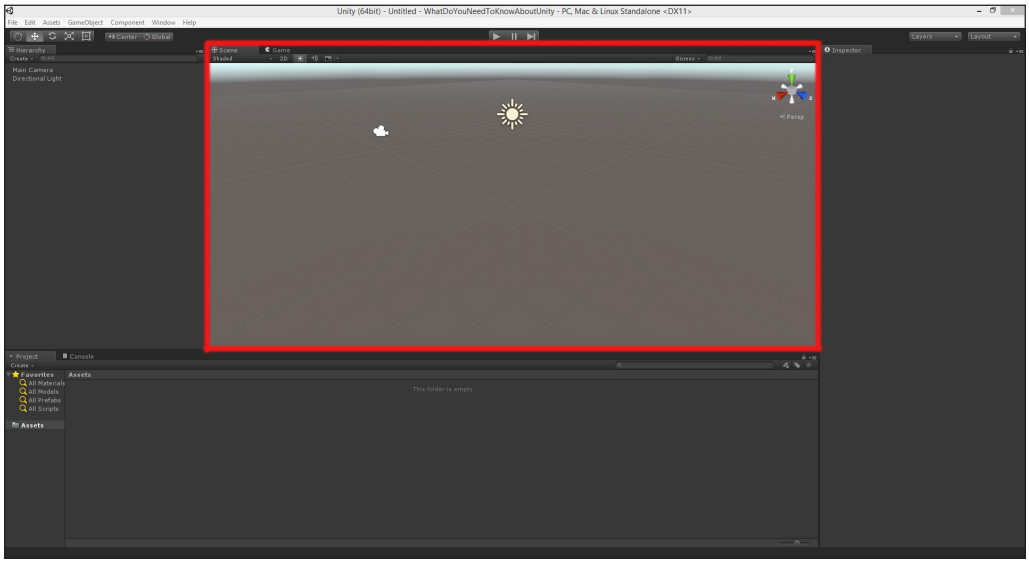
After we create our project or open an existing one, we can see many panels, as shown in the following image:



This may be a bit overwhelming in the beginning, but at the end of this chapter, everything will be a lot clearer. Let's go through these panels to see in detail what we can do with them.

The Scene View panel

The `Scene view` panel is the most important one. You can see it highlighted in the following image:

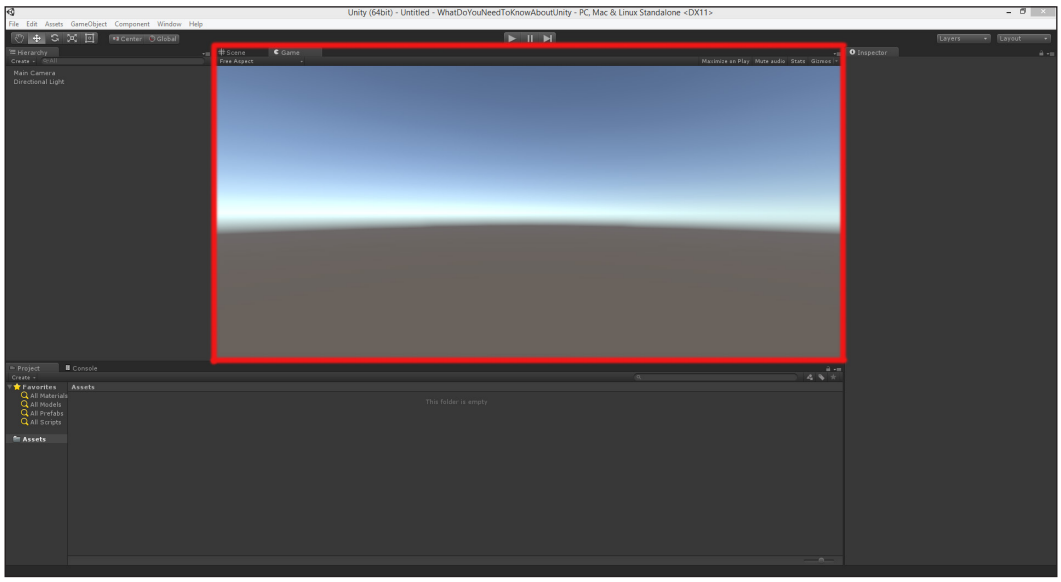


This allows you to change and modify your scene. A scene is where your game takes place. In a scene, you can create your 2D or 3D world where the player will move or interact. It's worthwhile to keep in mind that a game can also have more than one scene. In fact, each game that is created using Unity has different levels called **scenes**. Each scene can contain different objects that populate its environment. Using very simple scripts, the player can move in the game between scenes. However, every time a scene is loaded, it starts from the beginning. For instance, if a player has moved an object, and then reloads the scene, the object will be where it was in the first place. Retaining changes is possible, but they need to be stored in memory and then loaded back in runtime by using scripts.

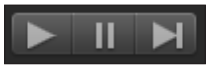
You can learn more about the Scene view in the `official documentation`.

The Game view

This view may not be immediately visible from the beginning because you need to click on the **Game** tab. You can find both the **View** and **Game** tabs highlighted in the following image:



In this view, we can take the guise of the player and see their perspective. This view is very useful to test your game and check whether everything is working as it should. You can run a simulation by clicking on the **Play** button on the top of the Unity Interface. Along with it, you can find the **Stop** and **Pause** buttons as well, as shown in the following image:

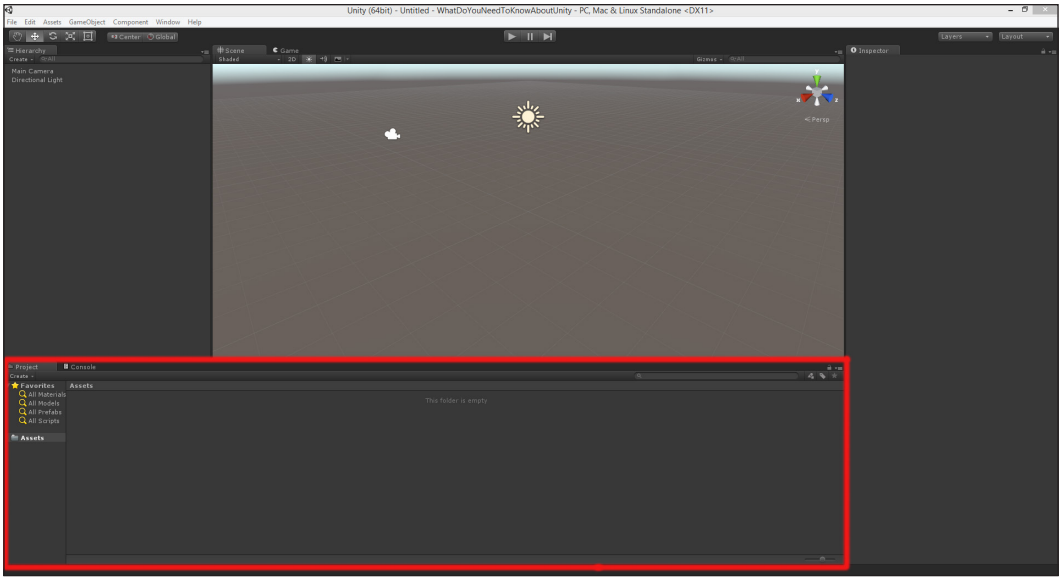


They will be always visible due to their importance to test or simulate the game.

Additional information can be found in the `official documentation`.

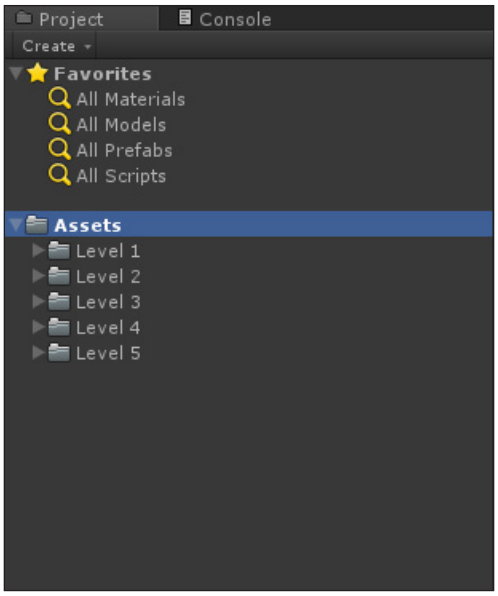
The Project panel

You can find this panel highlighted in the following picture:

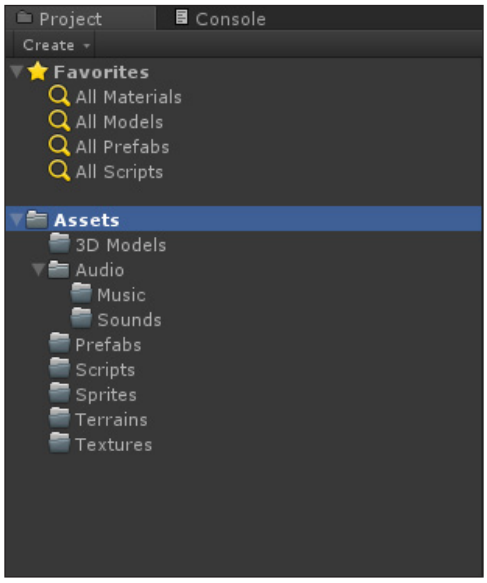


In this panel, we can view all the files that we have imported to the project. They may be ordered into folders. We recommend that you keep your files ordered as much as possible, especially if your project is large. As a result, you will be able to find what you need much faster, especially as your project begins to develop.

An example of ordered folders to store all the assets that can be sorted by levels is as follows:



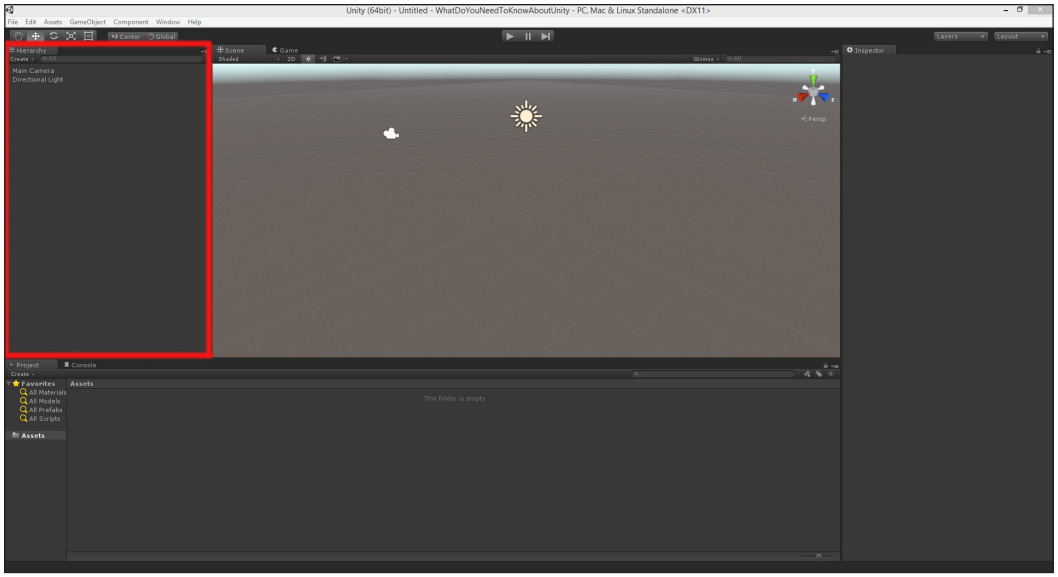
However, this structure is good for small projects or games. For bigger projects, a better structure will order all the assets by type, as in the following image:



You can find more information about the Project panel at `http://docs.unity3d.com/Manual/ProjectView.html`.

The Hierarchy panel

In the following image, the **Hierarchy** panel is highlighted:

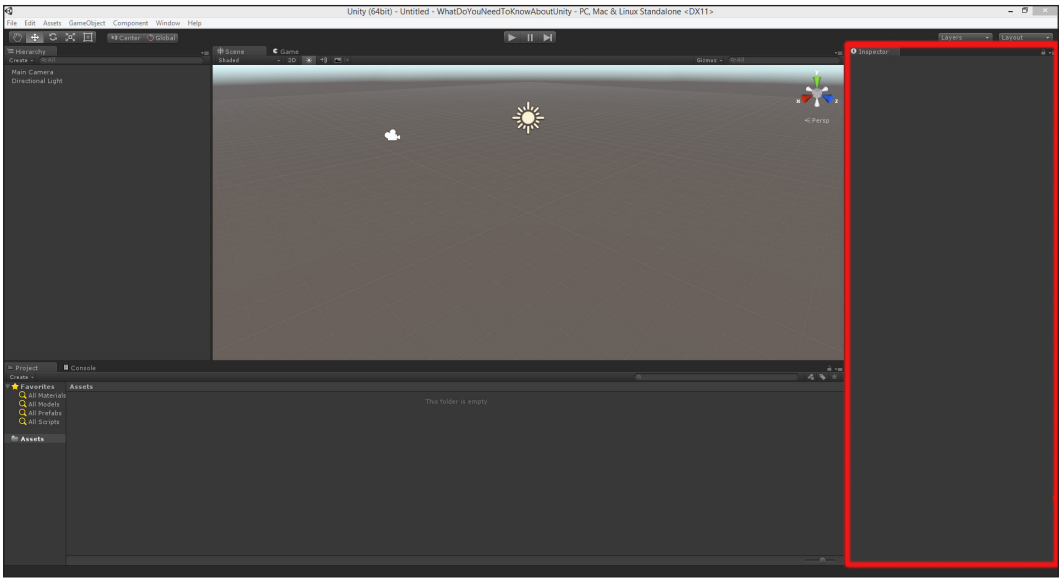


Do not confuse this with the **Project** panel. Here, you can see all the objects or Game objects, as Unity calls them that are present in the scene. For instance, you may import a tree into your project. However, in your scene, you can also place two trees of the same type. In the **Hierarchy** panel, you will see two trees even if there is only one in your project.

More information about this panel can be found at `http://docs.unity3d.com/Manual/Hierarchy.html`.

Inspector

Along with the **Scene View** panel, the **Inspector** is the most important tab. In fact, it is the core of Unity, where you can change all the settings and options for each single object in your scene or project. It is highlighted in the following screenshot:



As you can see, it is now empty. However, once an object is selected, it will display its details, parameters, and variables, which you can change to suit your needs.

In the official documentation, you can find out more about `Inspector`.

Other parts of the interface

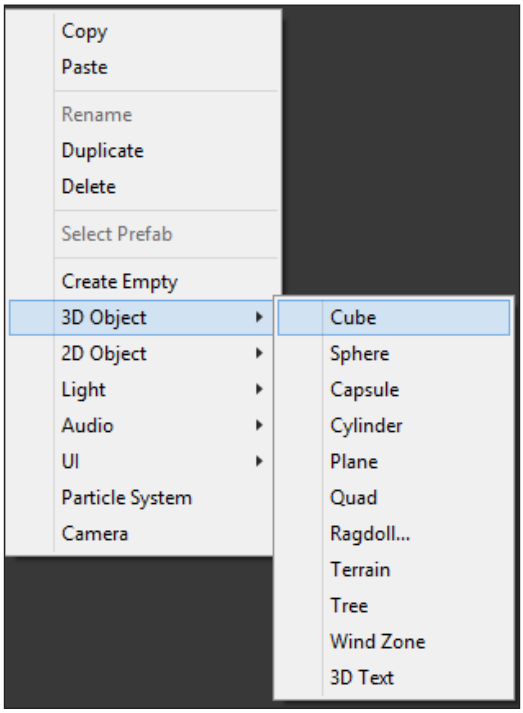
Of course, we only scratched the surface of what kind of panels or interfaces are available in Unity. Some of them may be accessed from the top bar, others from Inspector, such as the **Sprite Editor**. However, we don't need to worry about them now. I'm sure that after this introduction course in Unity, you will be able to explore Unity and discover new functionalities on your own or with the help of a guide.

If you want to explore the other parts of the interface, feel free to look at the official documentation at `http://docs.unity3d.com/Manual/UsingTheEditor.html` and `http://docs.unity3d.com/Manual/Sprites.html`.

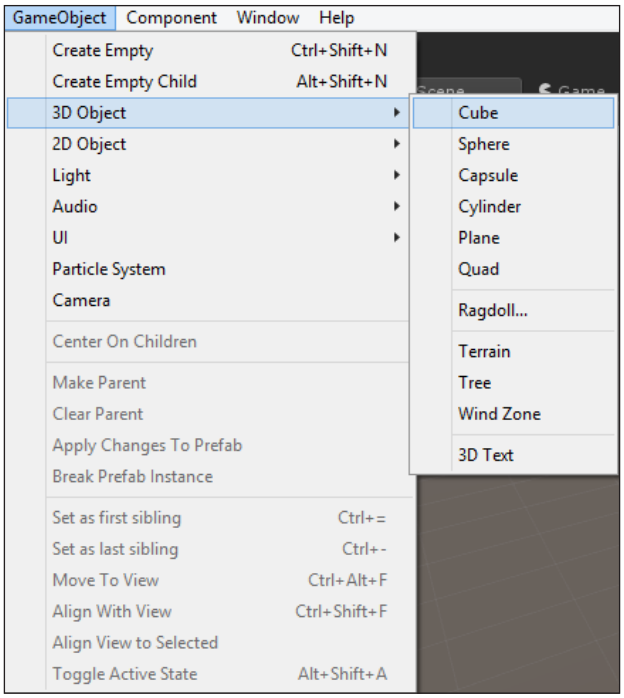
Placing objects

This section assumes that you are familiar with the panels that we presented before. If not, take your time to learn them because they are important, and we will use them throughout the rest of this eGuide.

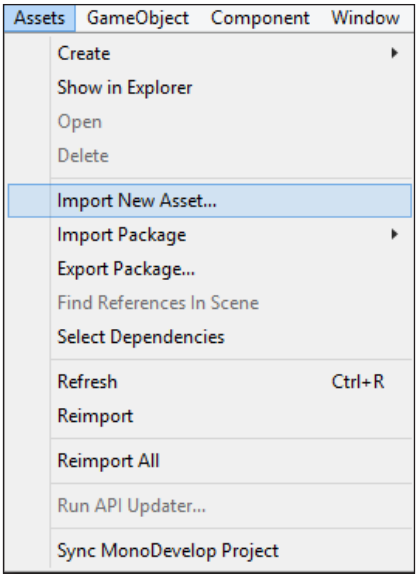
Let's begin by creating a `Game` object in our scene. There are different ways to do this. Right-click on the **Hierarchy** panel, select whether you want a 2D or 3D Object, and then select the kind of object that you want. For instance, you can choose **3D Object | Cube**:



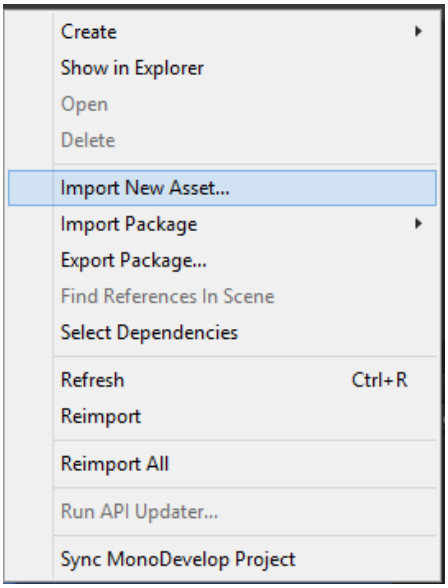
Another way to do this is using the top bar menu, arriving to a similar menu for the choice of which object to place:



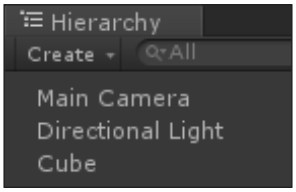
Another important way to place objects inside your scene is to drag them from the **Project** panel. However, to do this, you need to import at least one asset. To import an asset, you can select **Import New Asset...** from the **Assets** menu:



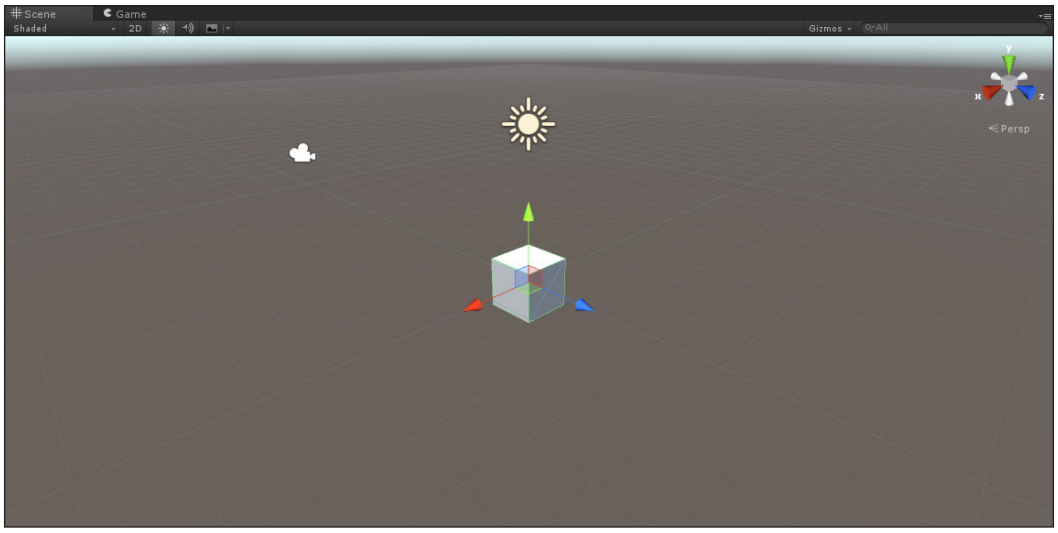
Alternatively, you can do this by right-clicking on the **Project** panel, and then clicking on **Import New Asset…**:



Once you place something in the scene, you will see the object appear in the **Hierarchy** panel like in the following image:



It will also appear in the Scene view, as shown in the following image:



As you can see, this is in perspective. Thus, we may want to look at our objects from different positions, so let's learn how to do this in the next section.

Navigating in the interface

The term navigation refers to the ability to look around from within the **Scene View** panel. If you never used a program with a 3D view, you may find this a little bit difficult in the beginning, but with a tiny bit of practice, this is very easy to master.

By holding the right-mouse button down on the **Scene View** panel, it is possible to rotate your camera or point of view. While still keeping the right-mouse button pressed, you can use the arrow keys on your keyboard to fly around the scene. Hold down the *Shift* key with an arrow to move faster. Therefore, we can watch the cube that we placed in a different perspective.

Before we move to the next section, we recommend that you spend some time mastering the navigation because this is really important.

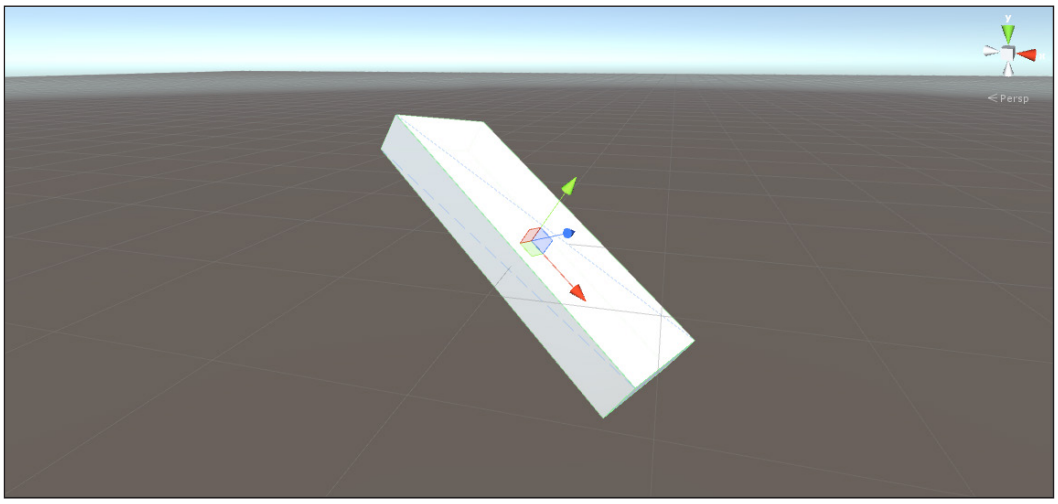
Transforming objects

Now that we know how to move around, let's learn how to manipulate game objects.

There are different ways to achieve this. First of all, we can use the **Inspector**. If we select an object in a scene, you should see a section called **Transform** in the **Inspector**. Here, you can modify the position, the rotation, and the scale along each single axis, as they are represented as vectors.

Rotation is not actually stored as a 3D vector like the other two, but it is stored as a **Quaternion** (a 4D Vector). A Quaternion is another representation of a rotation in the 3D space with four values. This is because with **Euler Angles**, the representation of the three angles for each axis (the 3D Vector with three values) cause different numerical instability in the 3D space. However, this process is quite invisible if you tweak the angles in the **Inspector**, but it needs to be kept in mind when you script.

I encourage you to change these values and see what happens in the **Scene View** panel. For instance, as an exercise, you can try to achieve something like the following:



As you can see, this appears to be a rectangular polyhedron (more specifically, a hexahedron). However, in Unity, this is just a cube scaled differently along each axis and then rotated a little bit. If you can achieve this very quickly, you are on your way; otherwise, keep practicing.

Another way to modify objects is directly in the **Scene View** panel. If you look above the **Hierarchy** panel, you can see the following five buttons:

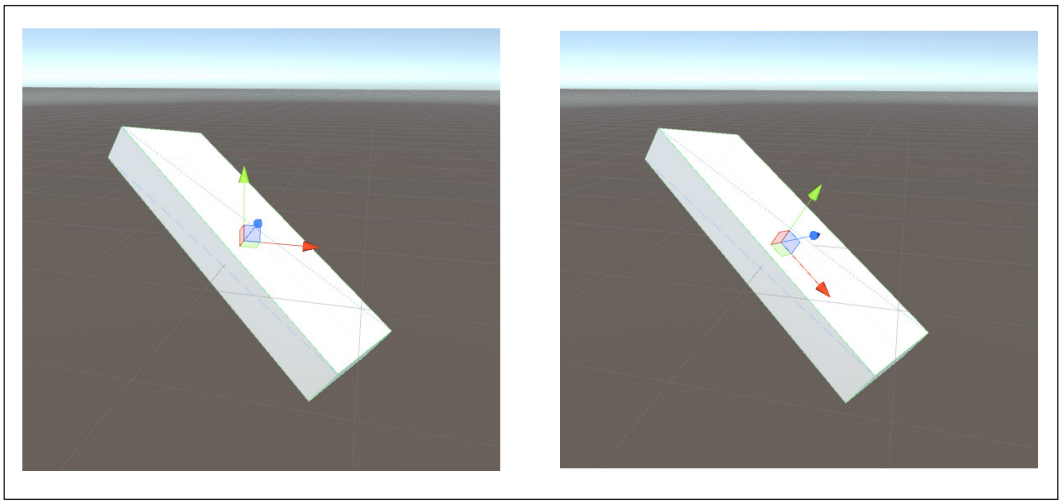
This is an ordered list form left to right of what they do:

- **Move**: This allows you to pan in the scene. It may be useful to navigate through the scene but not to transform objects. It may change into **Orbit** or **Zoom**, according to the navigation settings and shortcuts.
- **Translate**: This allows you to view the three position axes on the selected object. By dragging one of these axes, you can move the object along this specific axis. If you drag the square between two axes, you can move the object on the plane generated by these axes.
- **Rotate**: This allows you to view the three rotation axes on the selected object. By dragging one of these axes, you can rotate the object on this specific axis.
- **Scale**: This allows you to view the three scale axes on the selected object. By dragging the cube at the end of one of these axes, you can scale the object on this specific axis. If you drag the cube in the center of the object, you can scale the object uniformly.
- **Rect**: This is used to translate, rotate, and scale 2D objects, such as UI elements. We will look at this one later on in this eGuide.

There are two other important toggles that help transforming the objects in the scene. They can be found next to the previous buttons, and they are the following:

With the first button, you can decide whether you want to transform the object starting from its center or its pivot point. They are often the same, but when you import your custom 3D models, the pivot point can differ from the center.

With the second button, you can decide whether you transform the object according to its own frame or the world one. In order to understand the difference between these two, let's take the object of the previous exercise and select the positioning tool. Now, let's compare them, as follows:



On the left, we can see the axis on the world frame, whereas on the right-hand side, we can see the axis in its own frame.

Parenting game objects

Now, imagine we have a complex scene, and we want to move a building with all of the furniture inside that we have carefully placed. We would like to avoid repositioning everything. For this reason, Unity allows you to parent an object to another one, where the first one is called **child**, and the second one is called **parent**.

By moving the parent, all of its children (and the children of the children and so on) will move along with it. So, in the previous example, all of the furniture will be parented with the building. As a result, when we move the building, all the furniture will move with it. However, if we want to move only a closet, we can do this without moving the building by selecting only this object in the **Hierarchy** panel. Therefore, this makes the parenting system a powerful way to handle scenes with multiple objects.

Parenting an object in Unity is really simple. Select the future child in the **Hierarchy** panel, and drag it over another `game` object that becomes the parent. This is it. You can iterate the process and create a real tree of parenting, such as the one in the following image:

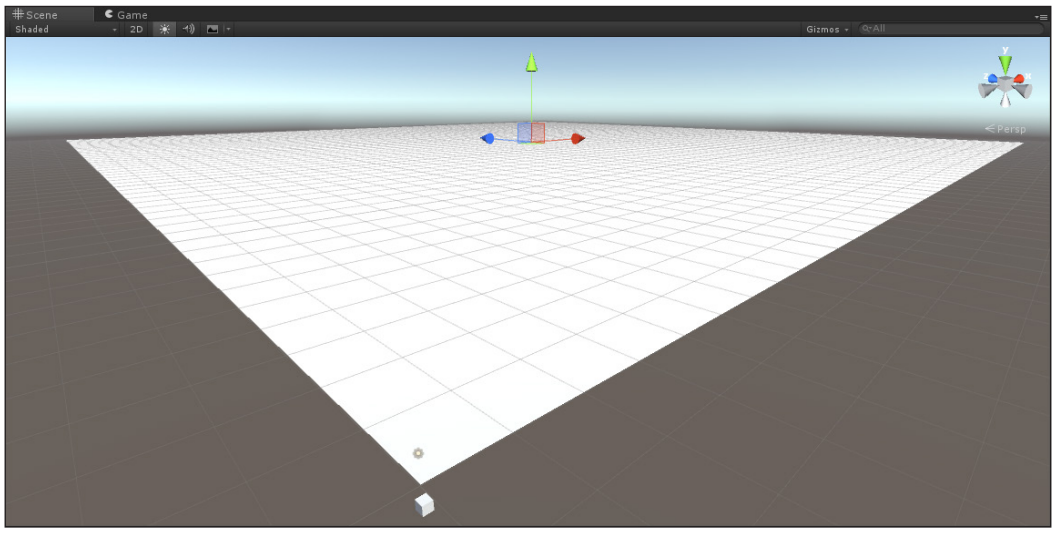
Setting Up a Scene – Part 1

In the previous chapter, we saw how to create a new project in Unity, and we learned the fundamentals of the interface. In this chapter, we will take a better look at how to create scenes in Unity.

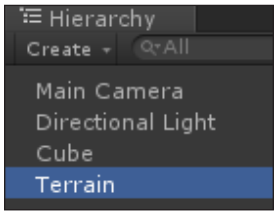
Using the Terrain tool

One of the fastest ways to create a 3D environment is to use the **Terrain** tool embedded inside Unity.

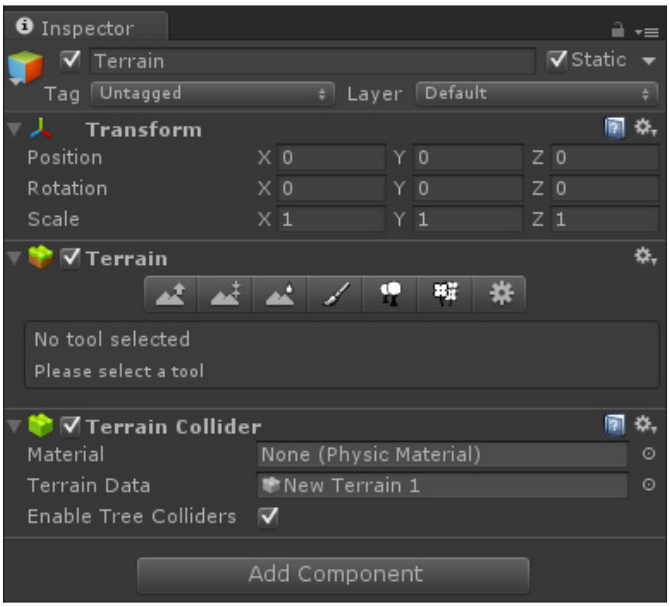
To begin, let's create a new terrain by right-clicking on the **Hierarchy** panel and then **3D Object** | **Terrain**. As you can see, a large rectangle has now appeared in your **Scene View** panel (you may want to navigate in the **Scene View** panel to find the angle that you prefer):



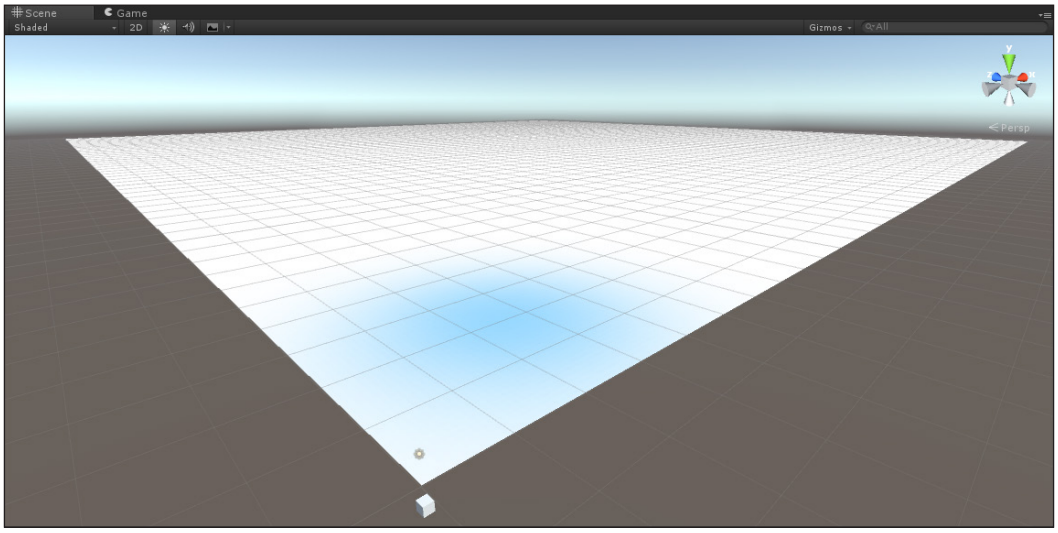
This is our terrain, even if it appears flat now. Next, we can click on **Terrain** in the **Hierarchy** panel to select it:



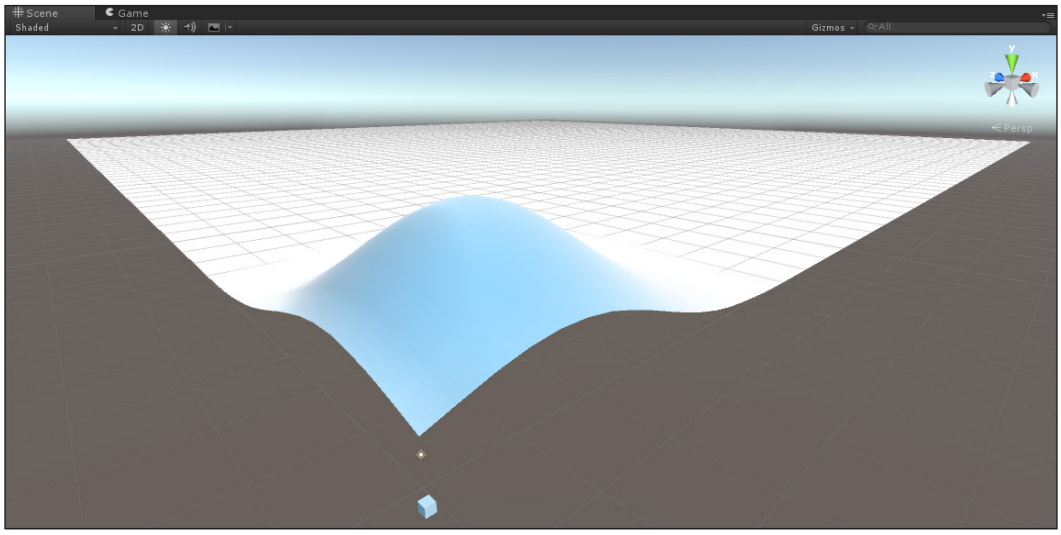
We can then see the **Inspector** change and assume this form:



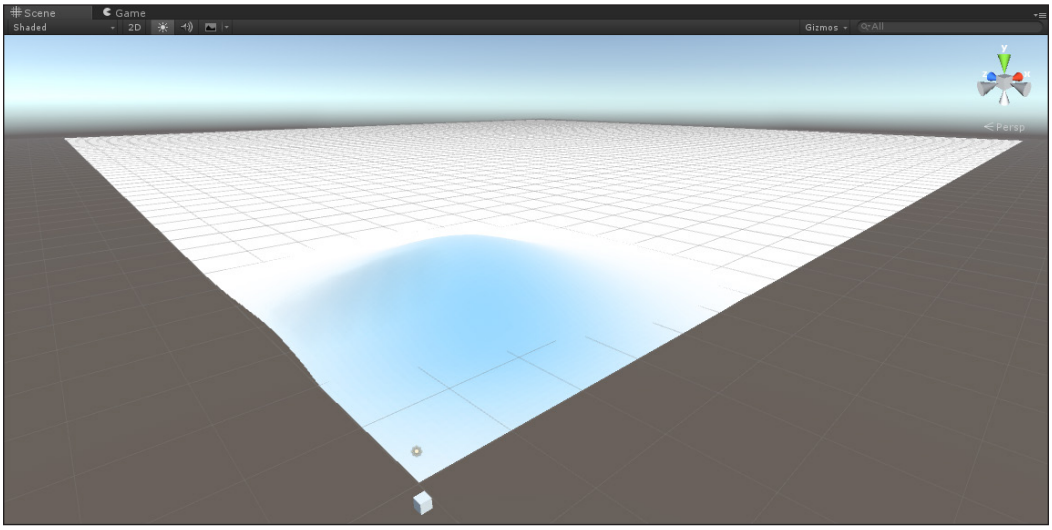
Note that inside it, there is a bar with different icons, and these are different brushes that we can use to paint our terrain. What this means is that we can raise up our terrain with just few clicks and create mountains very easily. Let's click on the left-most button, leave the default settings for the brush (the settings just below the bar with the different brushes), and bring our mouse in the **Scene View** panel.



As you can see, the terrain is a little bit blue where your mouse cursor is. It shows you how big your brush is. If you click on this area, you will see your terrain change shape and rise up like in the following screenshot:



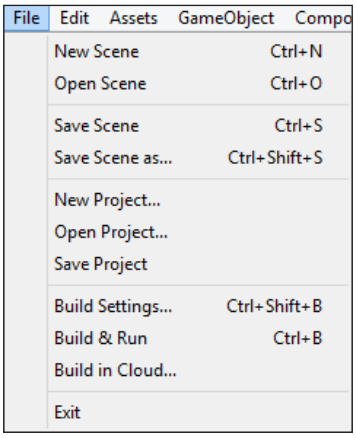
If you want to lower your terrain instead, hold *Shift* while clicking on the terrain with your brush. In addition to this, we can change our previous shape into the following one:



If you want to learn more about the **Terrain** tool and its other brushes, check out the official documentation of `Unity`.

Changing scenes

If we go through the top bar menu and select **File**, we should see some options that are related to our scene:



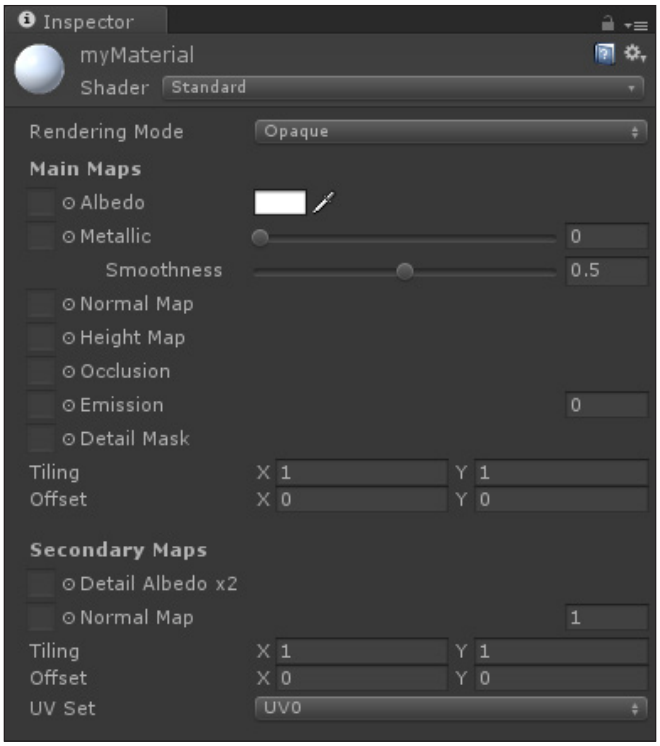
If we like what we created, we can click on **Save Scene** and choose where to save it. It's good practice to have a folder in your project called `Scenes` where you can save all your scenes. On the other hand, if we are not interested in saving it, just click on **New Scene**; and when Unity asks us whether we want to save our Scene, just click on **Don't save**. As a result, our **Hierarchy** panel is now almost empty (it contains the main camera and the directional light as default) and so is our **Scene View** panel.

Creating a material

Now that we are in a new scene, let's create a new material. Materials are very important because they allow us to make the world of the player colorful, instead of just an opaque white.

Right-click on the **Project** panel and then navigate to **Create** | **Material**. We will see it appear in our **Project** panel, and we need to type in a name. In this case, we can go with `myMaterial` and then press *Enter* on your keyboard:

If we select it, like in the previous image, then we will see many details in the **Inspector**, as shown in the following screenshot:

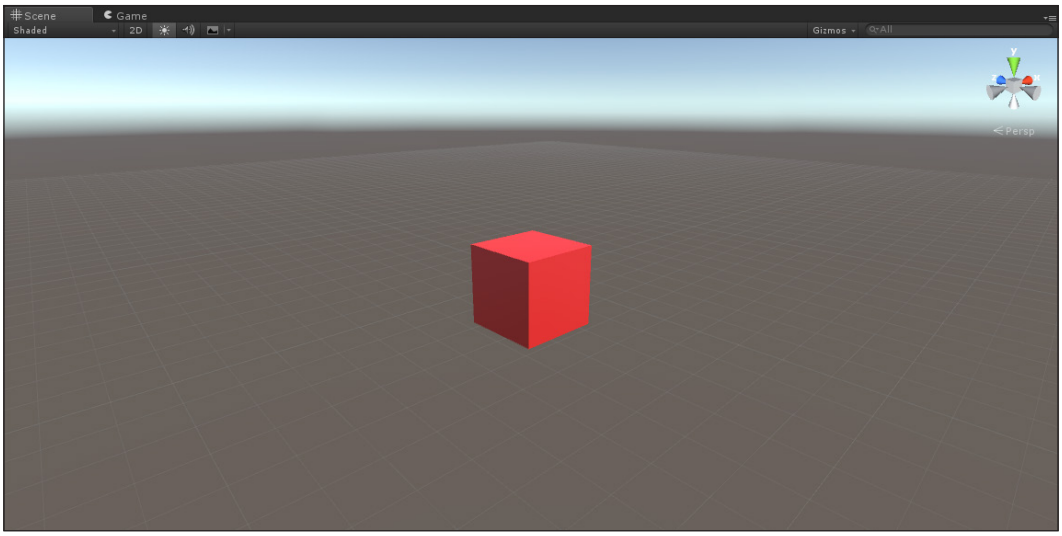


The most important one is **Shader**. It affects how a material will be rendered, this means how it will appear to the player and in the game. For this eGuide, let's stick with the default **Standard** option. An option that we may want to change is **Albedo**. If we click on it, Unity will display a Color Picker where we can choose a color, for example, red:

If we also have a texture, such as a brick texture (possibly with also a normal map), we can import them into the project, and then place them respectively in the **Albedo** and **Normal Map** parameters by dragging and dropping them.

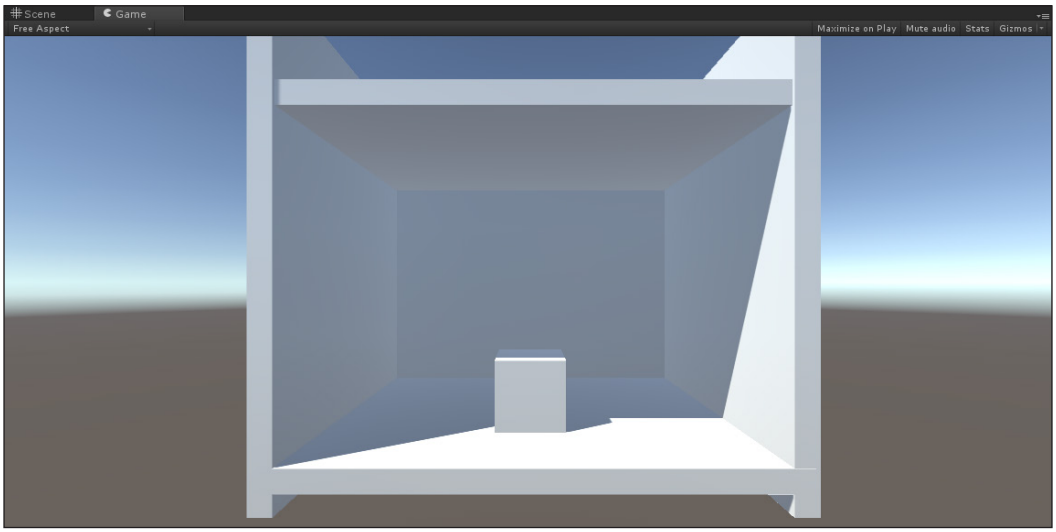
If you are not familiar or you don't know what these are, you can learn more about them in the official documentation of `Unity`.

Now, let's create another cube like we did in the previous chapter. Now, from the **Project** panel, drag and drop `myMaterial` on the cube in the **Scene View** panel (also on the **Hierarchy** panel works). As a result, our cube will assume the color that we picked for the material (and the texture if you had placed any), in this case, red:



Getting lit up

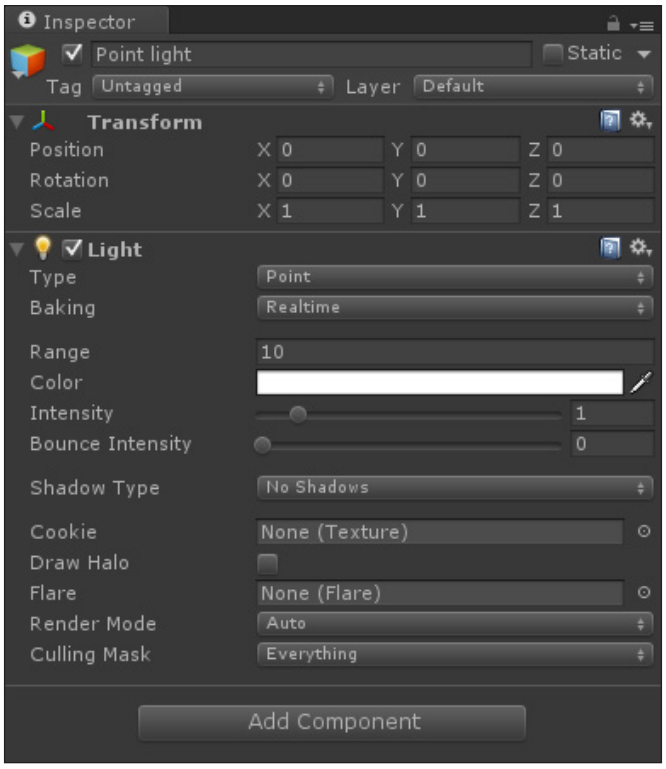
In order to better understand how lights in Unity work, let's create a scene like the following one:



As we can see, the scene is affected by a directional light, which comes as a default in every new scene.

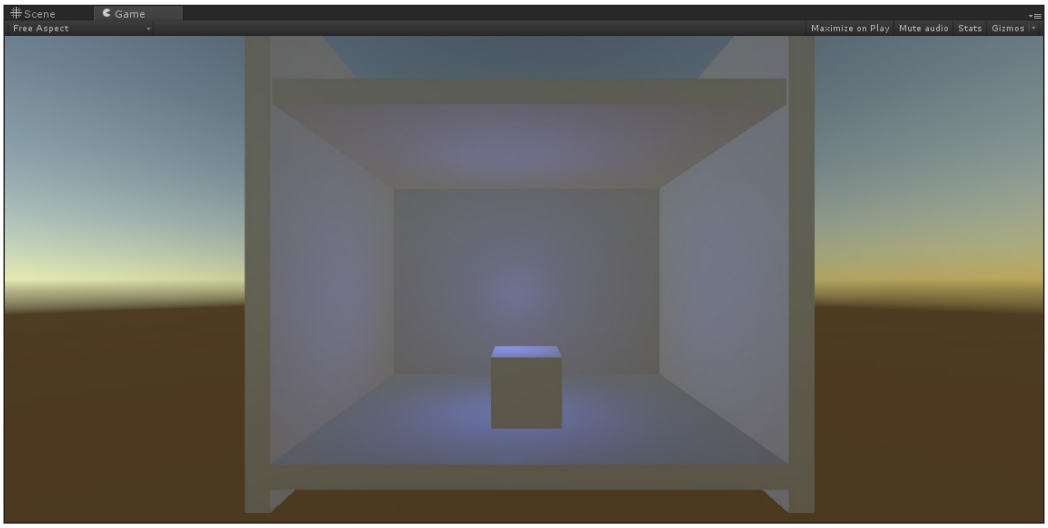
If we delete the default directional light and switch to the **Game View** panel, we can see that our cube looks really dark. This is because there are no more lights in our scene.

To add a light to our scene, let's right-click on the **Hierarchy** panel and then select **Point Light** from **Lights**. If we select it, we can move it like any other object in the **Scene View** panel. Furthermore, you will see some special settings for lights in **Inspector**:



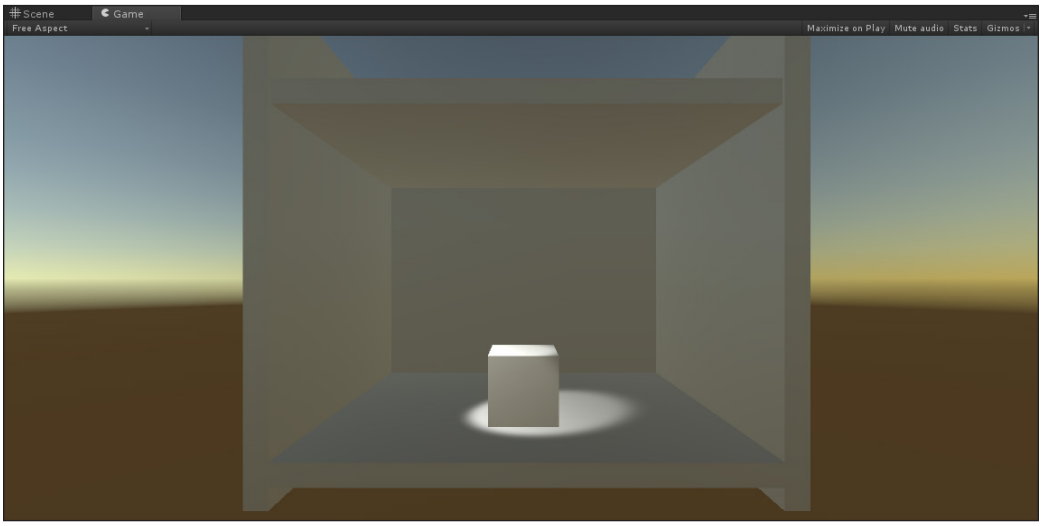
The two most important parameters that you may want to tweak in your game level are **Intensity** and **Color**. As these names suggest, the first one controls how much light is emitted, and the second one controls the color.

For instance, by selecting a blue color and placing the light just above the cube, we achieve this kind of result in the **Game View** panel:



We suggest that you always experiment with these values and understand how they influence the appearance of the scene.

Another kind of light is a spotlight. This is an example of it in our scene:



Of course, feel free to use as many lights as you want and find different combinations among them to illuminate your scene.

However, even if lights are really beautiful inside games, they are really expensive from a computational point of view. This means that you cannot use too many of them if you want your game to run smoothly and with a high frame rate.

Of course, there are also other kind of lights, and you can experiment with them before you move on to the next section. You can read more at `http://docs.unity3d.com/Manual/LightingOverview.html`.

Probe lights are particular because they can achieve amazing effects, and they are cheap because they are baked. However, you need to adjust a range of settings as well as baking the lights in the scene before using them. Especially because the objects in your scene may have particular materials that are affected by probe lights. You can read more about them at `http://docs.unity3d.com/Manual/LightProbes.html`.

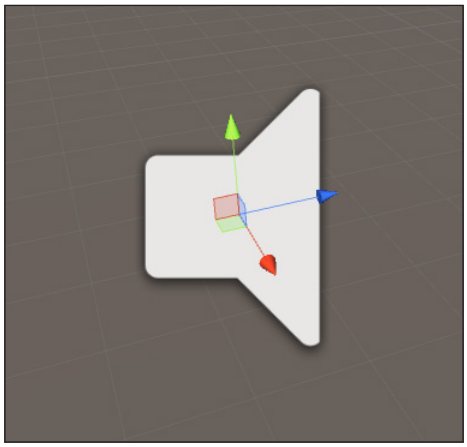
Including audio and music

A great way to surround the player with a nice atmosphere is to include audio and music into your game. If this is your first time in game development, you need to know that there are mainly two kinds of audio sources: 2D and 3D. It doesn't matter if your game is set in 2D or in 3D, but rather if the sound's source is 2D or 3D. A 2D source means that the player will listen to the sound independently whether they are in the game level. Usually, we use these kind of sources for music. On the other hand, 3D source is a sound that has a specific position in the map. Imagine if a door opens; the player should listen to its sound only if they are nearby.

Besides the kind of source that you choose, you need to attach an Audio Listener to an object in the scene, usually the player. This component allows listening to audio sources in the scene. When a new scene is created, an Audio Listener is always attached to the Main Camera.

As we have the Main Camera in the game, we don't need to attach an Audio Listener. Instead, let's create a new object by right-clicking on the **Hierarchy** panel and then clicking on **Create Empty**. You can also rename it `MusicObject`. Now, navigate to **Add Component** | **Audio** | **Audio Source** to add an Audio Source.

To easily and quickly identify where the Audio Sources are in your scene, Unity places an icon called **Gizmos**. You can see it in the following image:



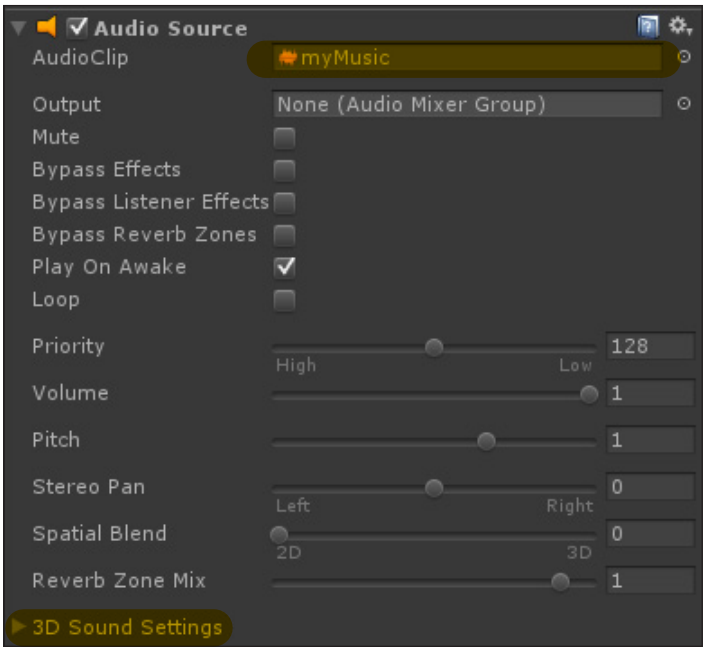
Now, you should import some music in your game. Keep in mind that Unity only supports the following formats:

- `.mp3`
- `.ogg`
- `.wav`
- `.aiff/.aif`
- `.mod`
- `.it`
- `.s3m`
- `.xm`

If you have trouble finding music for your game, there are a lot of online libraries where you can download sounds and music. Some of them are free even for commercial use. Some examples are `https://www.freesound.org` or `http://soundbible.com/`.

In any case, always remember to check the licenses and their conditions.

After you import your music, you can drag and drop it in the **Inspector** (after you select the **MusicObject** from the **Hierarchy** panel) inside the `AudioClip` variable. If you have any trouble understanding this passage, you can look at the following image for clarity:



Finally, we can click on **Play**, and we will see our cube with the light and listen to the beautiful music that you imported.

To learn more about the audio in Unity, you can refer to the `official documentation`.

Setting Up a Scene – Part 2

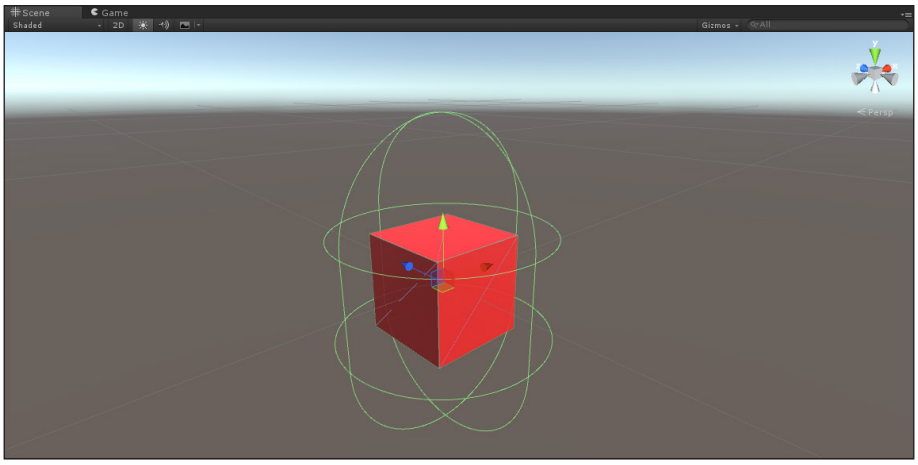
In the previous chapter, we saw how to create a terrain and set materials. Furthermore, we also saw how to use lights and music in our project. In this chapter, we will look at other features that we can use to create scenes in Unity.

Handling physics

Unity implements a physic engine on its own. As physics is generally expensive in a Game Engine, not everything is simulated into the physics of the game. Thus, in some way, we need to tell to Unity what we want to be affected by physics.

Two main components allow us to specify a behavior with the physic engine of Unity.

The first one is **Collider**. This can have different shapes, and it specifies which kind of form the object has in the physic engine. Usually, custom shapes are more expensive, but we may want to approximate a human body with a Capsule Collider, which we can see in the following image:



Even if it is an approximation, it works in most cases, except if we try to achieve something very specific.

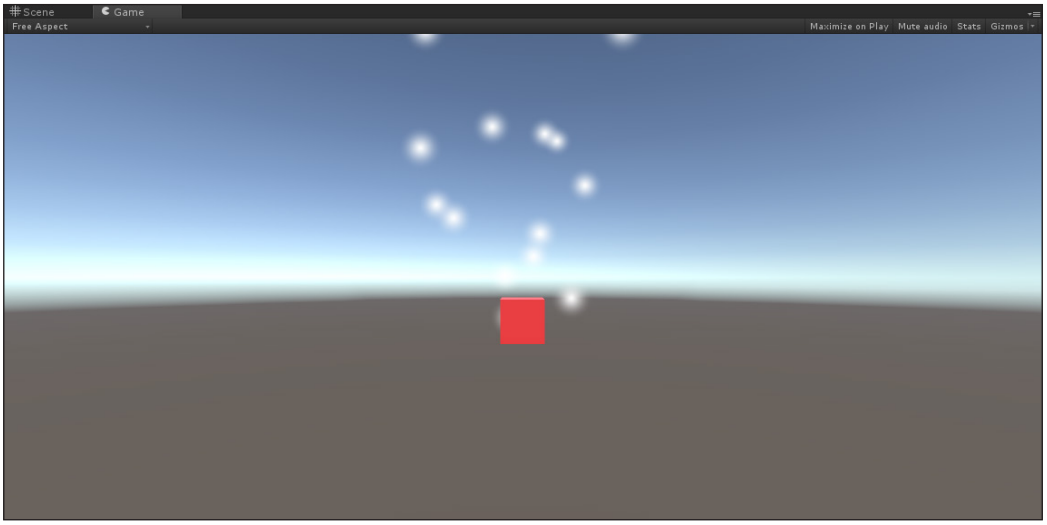
The second component is **RigidBody**. This tells Unity that the object where this component is attached should act as a rigid body. This means that it is affected by gravity and drag. For instance, create a cube and attach a rigid body on it (a cube collider is attached by default). Then, create another cube below the first one and scale it to create a kind of plane. Now, if we press **Play**, we can see that the first cube falls by the effect of gravity on the second one. However, the second one won't move due to the fact that it is not a rigid body (a rigid body is not attached on it).

These components are quite intuitive to use; therefore, feel free to tweak the parameters to see what happens. Finally, the official documentation is always a good place to quickly learn what a specific parameter does. You can consult it at `http://docs.unity3d.com/Manual/PhysicsSection.html`.

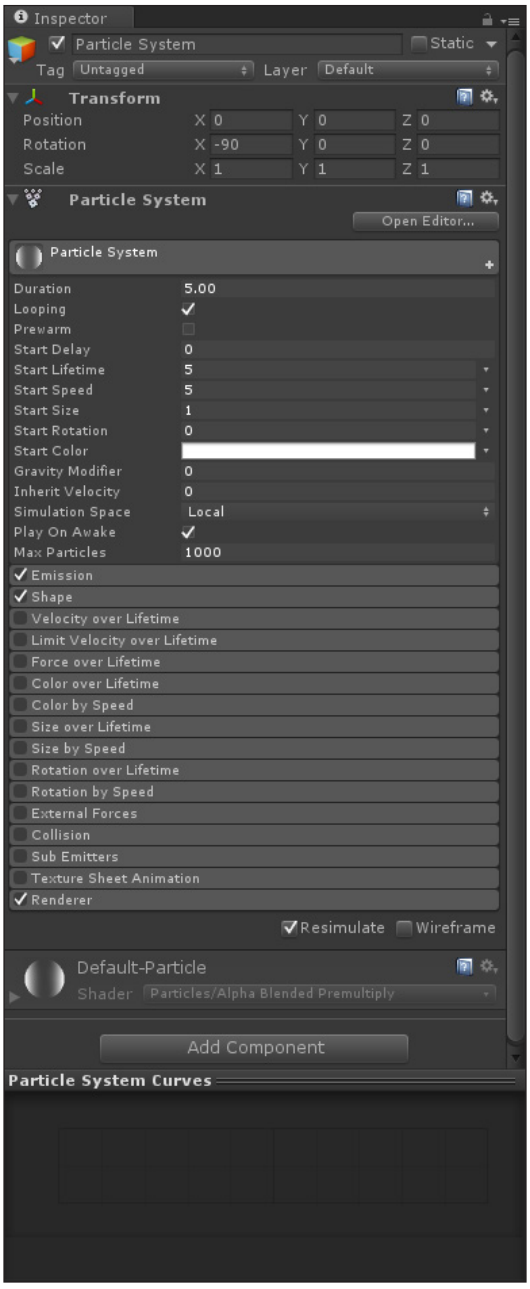
Adding special effects

A great way to add atmosphere to your game is using some special effects, such as explosions or dust in the air. You can do this using something called a **particle system**. This simulates some particles in the 3D space, and by varying how they look and behave, it is possible to create many special effects. Unity implements a particle system using a Particle System Component.

To quickly create a particle system, just right-click on the **Hierarchy** panel and then click on **Particle System**. As a result, we will see some moving white particles in the scene above the cube that we left from the previous chapter:



If we go to the **Inspector** panel, we will be overwhelmed by a lot of tabs. Each of them contains many options about the particle system, as shown in the following image. In fact, this is a very complex component that requires a little of bit of practice to get started.



We don't have the time to go through each parameter, but let's look at the most common ones:

- **Duration**: This allows us to set how long a particle will live, which means how long a single particle will be displayed onscreen.
- **Looping**: This is a checkbox that determines whether this particle system will loop or not. For instance, an explosion is not a loop effect (it is one shot), whereas dust in the air is.
- **Start size** and **Start color**: These two respectively set how big and what color the particles are. The name of these parameters begins with **Start ...** because particles can change shape and color over time.
- **Rate**: This important parameter is found inside the **Emission** tab. It allows us to set how many particles will be emitted over time.

With this said, you can learn more about particle systems in the `official documentation`.

However, the best way to learn particle systems is to have an effect in mind and try to recreate it, so the secret is just practice.

UI basics

UI means User Interface, which can include leaderboards as well as inventories, score, health, minimaps, and so on. Unity implements it using UI components.

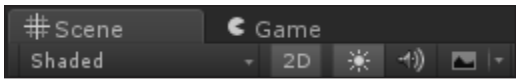
You can create a UI by right-clicking on the **Hierarchy** panel, and then select **Text** from **UI**. As you noticed, Unity added some components to your scene. For the moment, we can just ignore them. However, it is really important that all the UI is parented to the **Canvas** object; otherwise, it will not be displayed.

We just added a UI Text. If you select it, you can change its text variable to what you prefer in the **Inspector**.

We should see the following in the **Scene View** panel:

If you want to move it, you should first select the **Rect** tool because they are 2D objects, and using the **Rect** tool makes it much easier. You can find this tool in the top-left corner of the Unity interface:

Once the **UI** object is selected, you can focus on it by pressing *F*, and switch to the 2D mode, by clicking on the **2D** button in the **Scene View** panel:



If you want to switch back to the 3D mode, just click this button again.

We invite you to try our other UI components and experiment with them. However, there are a lot of places where you can learn more about the UI. The first place that is full of useful information is always the `official documentation`.

Lastly, you should definitely consider buying a book that is specifically about UI. For instance, the *Unity UI Cookbook*, Packt Publishing, has a perfect set of recipes that are ready to use. You will find all the basic concepts and much more, and you can get it at `https://www.packtpub.com/game-development/unity-ui-cookbook`.

Giving Life to the Scene

Except for the particle system that we saw in the previous chapter, our scene is quite static. Unity supports different ways to animate a scene, and we will briefly look at some of them here.

Animations

Unity allows us to both import custom animations made by other software and create our own animation inside the engine itself.

Importing animations

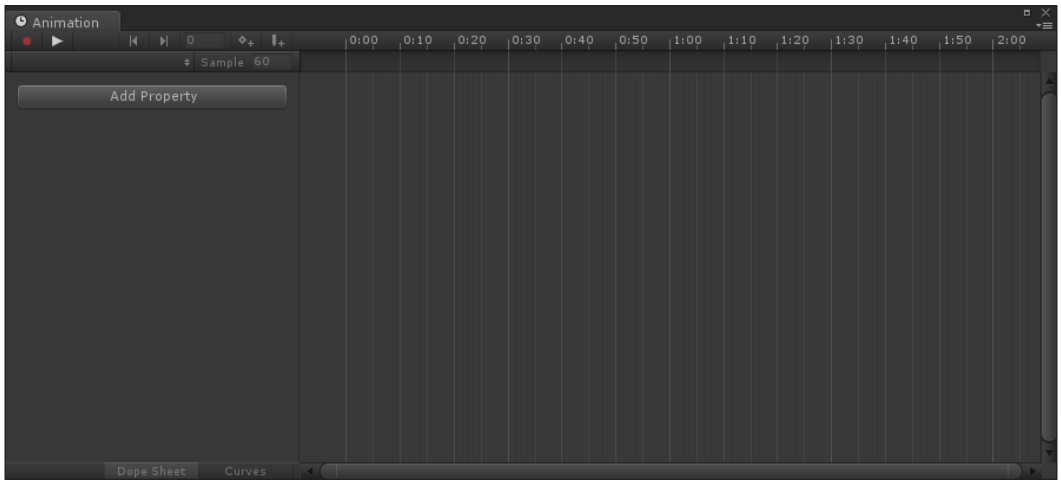
If we already have all our animation done with other software, it's really easy to import them. In fact, they can be imported as any other asset in your project. In **Inspector**, you can tweak some parameters, such as the play speed. However, some problems may happen because there is an entire workflow to properly import animations if they have something of particular value. We don't have room here to break through it, but you can learn a lot in the `official documentation`.

Creating animations

Of course, Unity is not an animation software, so don't expect it to create sophisticated animations. However, if you need just to open or close a door, the Unity animation system can help!

You can open the animation panel by selecting **Animation** from the **Window** toolbar.

A screen like the following will appear:



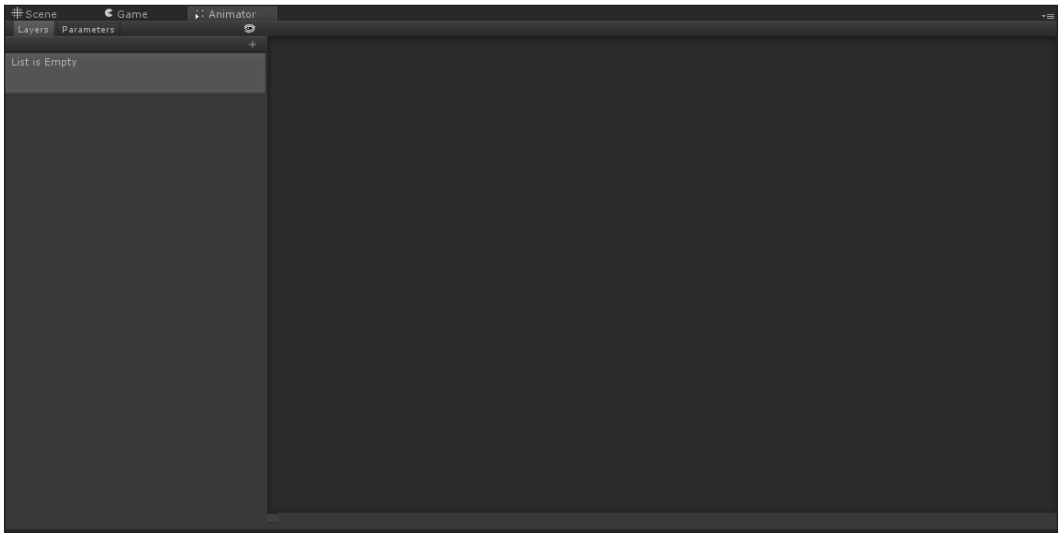
By selecting objects in the **Hierarchy** panel and then using the **Rec** button, you can record what you do with this object with keyframes. This may seem odd if you have never used any video editing tool, but the principle is the same.

You can learn more about the Animation system in the `official documentation`.

Mechanim

It is worthy to mention that Unity has an animation system when using more than one animation is required. This tool is called **Mechanim**, and it allows us to create entire graphs with conditions to trigger a transition from one animation to another one.

You can open this editor by selecting **Animator** in the **Window** menu. You will see something like the following:



You can learn more in the `official documentation`.

Scripting objects

One of the most powerful features of Unity is the ability to write custom scripts, both in JavaScript and C#. This allows us to fully customize the game that we are creating, as we want. This topic is so vast, you can find entire books only focusing on one part of the kind of code you can write.

In this section, we will see just how to add a very simple script to make the cube in our scene float.

Let's begin by creating a new script. Right-click on the **Project** panel and then select **C# Script** from **Create**. We can rename it as `myFirstScript`. If you double-click on it, we use another program, and this is `Monodevelop` by default, which is a development environment to write code. As we can see, our script is not empty, but there is something. In particular, there are two `Start()` and `Update()` functions. The first one is called the first time the script commences execution, and the second one is called every frame. We will use the last one. In fact, at every frame, we need to tell the cube where it will be.

Therefore, inside the `Update()` function, let's write the following:

```
void Update () {
  transform.position = new Vector3(0, Mathf.Sin(Time.time), 0);
}
```

As you can see, we have a `sin()` function that makes our object move up and down. As we are changing the *y* coordinate. Furthermore, we used `Time.time`, which is the time from the beginning of the game, so we know where the cube should be in every frame.

Save the script, and you can attach it to our cube by dragging and dropping it from the **Project** panel to the cube in the **Hierarchy** panel.

Finally, if you press **Play**, the button above the **Scene View** panel, we can see our cube goes up and down.

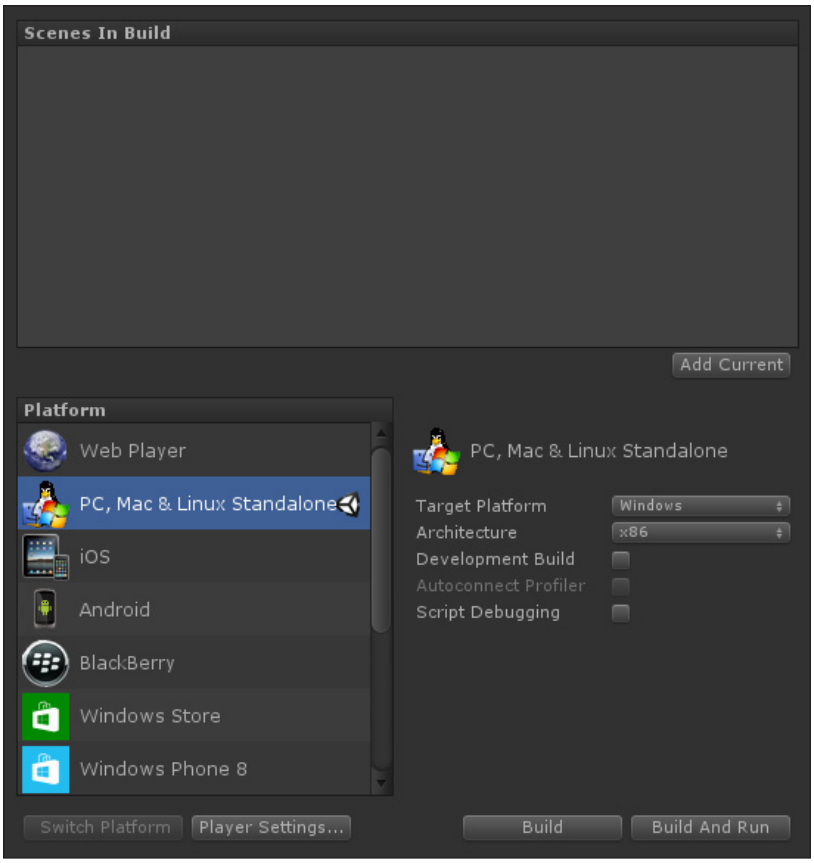
You can learn more about scripting in the `official documentation`.

Going Further and References

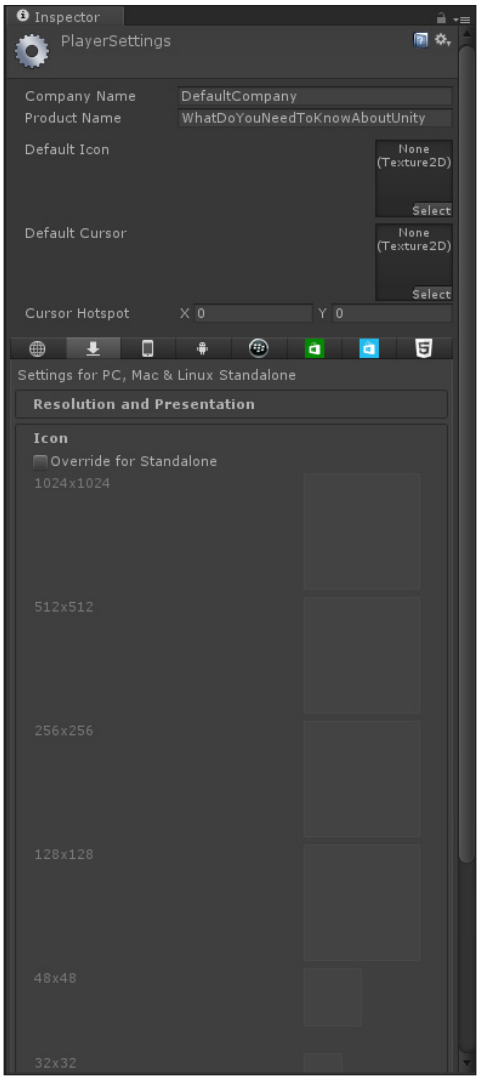
In this last chapter, we will look at how we can export our project so that it can be played by those who don't have Unity installed. At the end of this chapter, some further readings and references will be given in order to allow the reader to go further in this amazing world of Unity.

Exporting the project

In order to export your project, you need to go on the top bar and select **Build Settings…** from File. As a result, you will see the following screen appear, where you can set the building settings:



First, we need to add a scene. We only have one scene, so we add it by clicking on **Add current** scene. Then, we choose a platform, such as PC or Android. If you want to really go deeper in the settings, you can click on the **Player Settings…** button, and you will see this screen in the **Inspector** panel:



Here, you can granularly set how your game will be exported. Now, we can just leave the default settings, but if you are interested in learning more, you can check the `official documentation`.

Therefore, if we want to export the project in the previous screen, we just have to click on **Build** and Unity will ask us where to save it.

Congratulations, you just exported your first project!

Summary

This is the end for this short guide. We hope that you enjoyed going through it and you learned something as well.

We saw how to create a project in Unity and explored part of the interface where we can create our games. You learned the basic concepts of placing objects in our scene and how to navigate in the **Scene View** panel.

Then, we saw how to quickly shape environments with the **Terrain** tool. After this, we moved to creating materials and applying them to `game` objects. We enriched our scene with lights, music, and particle systems. You also learned how to handle physics inside Unity.

We looked at how to create User Interfaces (UIs), and how to import and use animations. We also wrote a small script to make an object float.

Finally, we saw how to export our project and how to change its settings.

What to do next?

Broaden your horizons with Packt

If you're interested in Unity, then you've come to the right place. We've got a diverse range of products that should appeal to budding as well as proficient specialists in the field of Unity.

To learn more about Unity 5 and find out what you want to learn next, visit the Unity tech page at `https://www.packtpub.com/tech/unity`

If you have any feedback on this eBook, or are struggling with something we haven't covered, let us know at `customercare@packtpub.com`.

Get a 50% discount on your next eBook or video from www.packtpub.com using the code: