Unity 2D Game Development Cookbook

Unity is a powerful game development engine that provides rich functionalities to create 2D and 3D games. Unity 2D Game Development Cookbook is a practical guide to creating games with Unity. The book aims to serve the purpose of exploring problematic concepts in Unity for 2D game development, offering over 50 recipes that are easy to understand and to implement, thanks to the step-by-step explanations and the custom assets provided. The practical recipes provided in the book show clearly and concisely how to do things right in Unity. By the end of this book, you'll be near "experts" when dealing with Unity. You will also understand how to resolve issues and be able to comfortably offer solutions for 2D game development.

What this book will do for you...

- Import files from other popular software into Unity: Textures, Models, Animations, and Audio clips
- Build up an actual game scene with cameras, lights, and the logic to control the game flow
- Manage collisions, and implement physics and game controls
- Make prefabs and instantiate them as game objects in the game scene
- Import and configure music and sound effects to be added to your games
- Design and control the user interface with the built-in methods of Unity

Inside the Cookbook...

- Straightforward and easy-to-follow format
- A selection of the most important tasks and problems
- Carefully organized instructions for solving the problem efficiently
- Clear explanations for what you did
- Apply the solution to other situations

Over 50 hands-on recipes that leverage the features of Unity to help you create 2D games and game prototypes

In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'Importing 3D Models and Animations'
- A synopsis of the book’s content
- More information on Unity 2D Game Development Cookbook

About the Author

Claudio Scolastici is an Italian game designer with a background as a researcher in the fields of psychology, artificial intelligence, and cognitive science.

He is employed by Italian game developer SpinVector as Technical Game Designer and Unity specialist.

In November 2013, he released a book titled Mobile Game Design Essentials for Packt Publishing, as well as tutorials about AI modeling and scripting at Digital-Tutors.

To those who never told me to look elsewhere...
Unity 2D Game Development Cookbook

There was a time when building games was a cumbersome and almost exclusive activity, as you needed to program your own game engine or pay a good amount of money to license one.

Thanks to Unity, creating video games today is still a cumbersome activity, though less exclusive and expensive!

With this book, we aim to provide you with a detailed guide to approach the development of an actual 2D game with Unity. As it is a complex process that requires several operations to be performed, we will do our best to support you at every step by providing all the relevant information to help you successfully make games with Unity.

Packt cookbooks offer knowledge in the form of recipes that describe individual tasks and how to perform them. This way, you are provided with a quick-reference guide that you can read in whichever order you may see fit for your actual development needs.

We thus encourage you to freely move back and forth between chapters to take full advantage of the flexible structure of cookbooks.

Enough of the premises, let's start by taking a look at the Unity interface!

The goal

In the last decade, a large section of the game development industry moved back to its garage roots, so to say, and opened its arms to embrace small groups of very motivated people who want to make games. The revolution of (almost) free 3D engines such as Unity and UDK allowed these small groups with no money to invest to give birth to their gameplay ideas and challenge the market by building up actual, professional games.
With this book, we plan to provide you with a detailed guide to approach game development with Unity. As game making is a complex process that requires several operations to be performed, we will do our best to support you in each step, providing all the relevant information to help you successfully move through the creation of your next game with Unity.

This book provides knowledge in the form of recipes that describe individual tasks and the steps required to perform them. This way, you are provided with a quick-reference guide that can be checked in any order you see fit for your actual development needs.

We thus encourage you, the readers, to freely move back and forth between chapters and take full advantage of the flexible structure of this book.

The mean

As a reference template to help you better understand the practical operations explained throughout this book, we plan to create a game prototype, featuring 2D gameplay with 3D graphics.

We believe this solution nicely fits two distinct needs: on one side, 2D gameplay is lighter to prototype, allowing us to describe the many features of Unity without the burdens of 3D mathematics.

On the other side, using 3D graphics (specifically for the game character and other game objects), we have the opportunity to discuss very important Appendix features of Unity, which would go unnoticed elsewhere. In the end, Unity is an engine to make 3D games, mainly!

The interface

With regard to the operations described and the pace we move between topics at, we assume you are already familiar with the Unity interface and its basic operations. Anyway, for those of you who may be a bit rusty with the Unity pipeline, let's begin our journey with a quick look at the Unity interface and the operations required to start a new project and configure the folders directory.
The preceding image shows the layout we are used to working with in Unity. Let's give a quick description of the main panels and windows available:

- **Main Menu**: This is where you **Load\Save** projects and game scenes, create and import new assets, and create game objects of different types to be added to the game scene. This is also the place where you add specific components to game objects to improve their features.

  Finally, this is where you configure the **Render** and **Project Settings** and where you configure the **Build Settings** for your games.

- **Hierarchy panel**: This panel lists all the objects that have been added to your game scene so far. Here you can select a specific element to be manipulated in the game scene or add components to improve its behavior and capabilities.
Project panel: This window lists the project folders and their contents. If you want your Unity project to keep nice and clean, we suggest you make extensive use of folders, by adding one specifically for each type of game asset (models, animations, textures, audio clips, animator controllers, and so on) you plan to have in your game. This way, whenever you need to access a certain asset, you know exactly where to search for it!

The following image shows an example folder directory of a project of ours:

![Example folder directory]

Editor window: This is the main Unity panel, the one that is used to actually assemble the game. Any GameObject that is required by your game must at some point be instantiated (by physical drag and drop or by code) here!

Game scene: This panel shows what the game looks like from the player's perspective. It displays the output of the main camera from the game scene and it is very useful to actually test what's happening, especially when you are studying specific graphic solutions or the disposition of GUI elements on the screen.

Inspector panel: This panel allows you to edit the assets available in the Project panel. It contains a lot of functionality, and we will often refer to the Inspector panel and the object properties displayed here, especially upon importing new assets (Models, Animations, Textures, and Audio clips) in our project.

If this super-quick description doesn't suffice, we recommend you go online and check for a beginner's tutorial about Unity. There are plenty available, both for free and for a price (for example on www.digitaltutors.com). Unity itself offers plenty of resources; you could start with this one: http://unity3d.com/learn/tutorials/modules/beginner/editor.
The words

As mentioned, we assume you are familiar with this interface and know how to navigate between panels. We also assume that you have a clear understanding of the basic terminology of making games: you know what a mesh or a material is, you know what we mean when we talk about animation clips and timelines, you know what a collision or a particle system is and what GUI stands for...don't you?

The assistants

As Unity is not provided with an embedded editor to create graphic contents (both 2D and 3D) or audio, we look to third-party software to accomplish these tasks.

With regard to graphics, we assume Photoshop (CS6) as the reference software for 2D images and Maya (2014) as our 3D editor of choice. These are both worldwide industry standards, and we believe that by taking this decision, we are actually helping you get familiar with tools that, sooner or later, you will have to confront in game development.

Still, as this book focuses on Unity, we take care of providing you with the required graphic assets to follow the recipes, as we cannot afford to provide you with a guide to powerful software such as Photoshop, Maya, or Blender. We encourage you to learn at least the basic operations with these software, as it will help you better deal and take advantage of the power of the Unity engine.

Start a new project in Unity

We are now ready to start a new project and deal with the Unity project directory.

When starting a new project in Unity, we are asked to set a name for it and add what we need. Follow us in the next recipe, which shows you how to perform this fundamental task.

Getting ready

Assuming you have already installed Unity, you are nicely ready to proceed.

How to do it...

1. Launch Unity. The Project Wizard window opens to start a new project.
2. In the NewProject tab, select a destination directory and type a name for the project. Our choice is Unity_Cookbook.
3. No need to flag any packages from the list. We will import packages as we need them through the development process. Simply hit Create. You can look at the following image for reference:

![Image of Unity Project Wizard]

**How it works...**

Well, this is fairly simple. Unity sets a new project in the chosen destination directory, creating a file structure to store anything required to run the project, edit it, and upload additional contents to be used in the scene.

**Adding a folder to the project directory**

**Getting ready**

We just need a new open project to perform this task, so you should be ready from our last recipe.
The file structure

We are almost ready to begin working on our project. We would just like to have a look at the structure of the directory we are using for this project and explain the criteria to effectively manage it.

The following screenshot displays the structure of the directory with an example list of folders:

As you can see, we have a number of folders, one for every important asset that we are planning to use for the game. We prefer having all different types of assets well separated in a reasonable and meaningful number of folders, so we always know where to search for what.

For now, we just have one folder for the coding, one for the interface elements, one to save our game scenes, and the last one for 2D textures.

If you think we don't have enough folders, well, you are totally right. But don't worry, the list is going to grow very soon, starting with our next recipe!

With this next recipe, we'll show you how to add a folder to our project directory. Stick with us!
How to do it...

1. In the Project panel, right-click anywhere in the window and select Create Folder, as shown in the following screenshot:

2. Type a name for this folder. We recommend the name Models, so we can use this folder to import Models by following the recipes of the next chapter:
How it works...
Unity is extremely flexible with regard to adding moving folders into its Project structure. Folders can be equally created inside or outside the software interface, and files can be both imported or simply copied from one folder to another outside of the software environment. When the focus gets back to Unity, it automatically updates changes we made. Thanks Unity, we appreciate that!

What this book covers
Making a game from scratch is no easy task. Games, even those that look basic at first glance, are a collection of elements, or assets, belonging to different fields of practice and requiring distinctive skills to be assembled.

With this book, we aim to provide you with what you need to know to make games with Unity. Each chapter covers a single topic and provides a set of practical recipes to learn how things are actually done in Unity.

The following is the list of topics we cover with this cookbook:

Chapter 1, Importing 3D Models and Animations, deals with the process of exporting 3D models from Maya into Unity and the operations required to correctly set up imported FBX with models and animations in Unity Inspector.

Chapter 2, 2D Assets for Unity, explains how textures are imported in Unity, how to set them up, and what a texture atlas is. It also deals with materials and how to add lights to a scene.

Chapter 3, Animating a Game Character, discusses Mecanim, the built-in tool to animate characters in Unity, introduced with Unity 4. It explains how to import and configure animation clips, how to create transitions between them, and how to blend animations.

Chapter 4, Taking Control, shows how to improve the graphic appeal of your character by adding a normal map to its material, and it also introduces the topic of game controls, explaining the difference between the Character Controller and the RigidBody components.

Chapter 5, Building Up the Game Level, shows how to create the assets to build up an actual game level made of platforms and gaps for the character to jump. We also implement the game controls and improve the gameplay by adding collectible game objects.

Chapter 6, Game Scenes and the Graphic Interface, explains how multiple scenes are added to a game. We also introduce the topic of finite state machines to control the screen flow of the game. Finally, we explain the basics to create a Graphic User Interface.
Chapter 7, *Improving the Gaming Experience*, shows you how to add audio to the prototype, by importing and configuring audio clips. We also introduce Particle Systems and show you how video clips can be played in the scene.

Chapter 8, *Sprites, Spritesheets, and 2D Animation in Unity*, deals with sprites, spritesheets, and sprite animation. It also explains the features of the built-in Sprite Editor of Unity.
In this chapter, we prepare the assets to build up our game prototype, starting with the process of exporting 3D models from Maya, our 3D editor of choice, into Unity. We also take care of correctly setting up the imported models and animations in Unity Inspector once they get imported.

In this chapter, we will cover the following recipes:

- Setting up a scene in Maya
- Using groups to rotate FBX files
- Exporting FBX files from Maya
- Configuring imported FBX files in Unity
- Exporting animations
- Configuring imported animations in Unity Inspector

**Introduction**

When building up a game, we usually start by importing the graphics assets to actually build up and prototype the gameplay.

In this specific case, we decided to begin with 3D models. Before importing the models, you should take care of bringing in the textures. Feel free to switch between *Chapter 1* and *Chapter 2, 2D Assets for Unity*, which focuses on textures and materials. A cookbook is specifically designed to leave the readers free to access the contents in whichever order they prefer.
We assume that you have the assets to test the operations explained throughout this book; in case you don't, you can download the contents available on Packt Publishing website.

When importing models from a 3D software into Unity, there are several settings to be defined: scales, source materials and textures, rigging and animations, and many others. We will discuss the most important setting soon.

For the importing process to be fully successful, it is also important that the scene in the 3D editor is properly set. When modeling stuff with a 3D editor for a 3D engine, it is important that scales, lights (if available), and cameras match between the scenes, or your models won't fit the game levels properly.

For the recipes of this chapter, we decided to pick Maya as our reference 3D editor. We do not mean that Maya is the best software, but there are plenty of reasons for this choice. Native Maya files are supported by Unity, and the LT version of Maya allows you to perform "one-click-exporting" of Maya scenes directly into Unity (http://videos.autodesk.com/zencoder/content/dam/autodesk/www/products/autodesk-maya-lt/video/send-to-unity-fbx-export-video-1280x720.mp4).

Maya is also an industry standard for 3D artists, and it is supported by both Windows and OS X (while 3D Studio Max, for example, isn't). You can check out 3D forums to delve into the differences between 3D software. The following is a list of very popular forums to begin with:

- http://forums.cgsociety.org/
- http://www.polycount.com/forum/

### Setting up a scene in Maya

The first point to keep in mind when setting up a scene in Maya is that the standard unit in Maya is 1 cm, while the standard unit in Unity is 1 m. So, whenever you export an FBX file from Maya into Unity, Unity scales it down to 0.01 percent of its original size.

Another very relevant point is that Maya and Unity are affected by strange kinds of idiosyncrasies that put them on opposing sides, with regard to what left or right and front or bottom mean. This is not something that only happens between Maya and Unity. Many 3D software disagree about the concepts of right and up. To get an idea, have a look at the following image, taken from Unity's forum:
As you can see, the red arrow, representing the left-right axis in the 3D world, may point to the left or right on different software or file formats, and the green and blue axes may switch to alternatively point to the forward or upward directions.

With Maya and Unity, what happens is that the front in Maya is the back in Unity. So you model the front of a character in Maya, and when you import it into Unity, it shows its back.

How do we deal with this? There's more than one option available, and turning the camera by 180 degrees in Unity is not the only one. We will show you how to deal with this problem but, before this, we need to learn how to actually export an FBX file from Maya and add it to a Unity scene, which is what we will do in the first recipe of this chapter.

**Getting ready**

For this recipe, we need a Maya scene with a 3D model, any model with at least one material applied to it. A textured model is provided with the contents of this book.
How to do it...

In this recipe, we will show you how a model is exported from Maya using the default FBX exporter panel and how the FBX file will get imported in Unity.

1. Open your model in Maya.

2. Open the outliner panel, and from the hierarchy, select the root node of your model. Remember that it is good practice to name the root node with a meaningful name, such as root. It can turn out to be useful, for example, when managing the exporting process through scripts (as shown later).

3. Now, in the top menu window, navigate to File | Export Selection. The Maya exporter panel will open, as shown in the following screenshot. Don't bother with the panel on the right-hand side with the actual settings; we will get back to it in a while.
4. Be sure that FBX export is selected from the drop-down menu at the bottom.
5. Put a name you like in the File name field.
6. Click on Export Selection to save the file in your destination folder.
7. Now open Unity and, in the project panel, right-click and select Import New Asset.... from the menu:

8. Select your saved FBX file from the Explorer window that opens and click on Import.
9. Alternatively, you could have directly exported the FBX file from Maya into the Assets/Models directory of your Unity project.
10. Now select the FBX file from the project panel and drag it onto the scene. The following screenshot shows what happens in Unity:

![Screenshot of Unity scene with a flipped model]

**How it works...**

The operation of exporting FBX files from Maya is simple: select the actual root node in the hierarchy and click on the **Export** button. But, as you can see, unless we use some precautions, the result of importing an FBX file from Maya is that the model is flipped by 180 degrees on the y axis in Unity. This happens because the blue arrow that represents the z axis in Maya points in the opposite direction in Unity. As a consequence, the model shows its back to Unity’s camera.

As we write, there are rumors that this issue is going to be solved in forthcoming Unity versions. For now, we will provide custom solutions we have used ourselves.

**Using groups to rotate FBX files**

An efficient solution to dealing with the discrepancy between Maya and Unity is to act on Maya’s side and rotate the model on its y axis there. Though, as we write, this problem is going to be solved soon by Maya LT, we offer a solution here that prevents the imported FBX file from acting strangely once they are turned into prefabs in Unity. The idea is to use the so-called "groups" to apply the transformations required and yet get a clean hierarchy for the prefab to appear in Unity. Let’s see how to do it.
Getting ready

Open the scene again with the model we used before and be ready to follow our instructions.

How to do it...

In this recipe, we will show you how to use groups and hierarchies in Maya to export a model that will not show its back once it gets imported into Unity. Have the Maya scene open on your screen.

1. From the outliner panel, select the root node of your model. Be sure that the model is at the 0,0,0 position with 0,0 rotation.

2. With root node selected, press Ctrl+G to create a group in the hierarchy.

3. Double-click on the newly created group name to edit it and type rot_180 (this is actually just for reference so we know what the group means).

4. Set a value, namely 180, for the rotation on the y axis in the Transform Attributes panel.

5. With the rot_180 group selected in the hierarchy, press Ctrl+G again to create another group. Name this group export after double-clicking on the group name in the hierarchy.

6. Now you can select the export node to export the selection in order to get an FBX file out of this model.

How it works...

By using one group for flipping the model on the y axis and another to make a selection featuring neither rotations nor translations for the export, we made sure that the FBX file won’t have any unexpected rotation or position offsets that will affect its behavior once it gets scripted into the code in Unity.
There's more...

Another technique we will only mention here is to use your programming skills and code an `AssetPostprocessor` class to handle the process automatically.

`AssetPostprocessor` is a class in Unity, provided with several methods to act on the pipeline for importing assets into Unity.

What one could do is add a custom attribute to the model in Maya, something like turn me 180 degrees on the y axis when imported, and let the `AssetPostprocessor` class read this attribute and perform the transformation.

You can learn more about the `AssetPostprocessor` class by checking the scripting reference guide at [http://docs.unity3d.com/ScriptReference/AssetPostprocessor.html](http://docs.unity3d.com/ScriptReference/AssetPostprocessor.html).

Exporting FBX files from Maya

Now that we are done with our first import, we can approach the many settings available in the Maya FBX exporter panel. There are several operations that require our attention, so follow us in our next recipe.

Getting ready

For this recipe, you just need the Maya model we used so far.

How to do it...

1. Open the scene with the model in Maya.
2. From the outliner panel, select the root node of your model. Be sure that the model is at the 0,0,0 position with 0,0,0 rotation.
3. In the top menu window, navigate to File | Export Selection and the Maya exporter panel will open.
4. Be sure that FBX export is selected from the drop-down menu at the bottom of the panel.
5. Put a name you like in the File name field.
6. From the Options... panel on the right-hand side, let's examine the first group of settings. Edit the General Options, Reference Options, and Include Options tab, as shown in the following screenshot:
7. Now we can move to the next group of settings. In **File Type Specific Options**, make sure that the **Include** and **Geometry** settings are configured as shown in the following screenshot:
8. Next comes the animation-related group of properties. Since we are not importing animations with an FBX file, unflag the **Animation** option entirely. This action will disengage all the following properties (see that they are barred in the following screenshot).

9. Unflag **Cameras, Lights**, and **Embed Media**; we don't need any of them either.

10. Flag **Input Connections** in the **Connections** tab. Refer to the following screenshot for the last three steps:

11. Finally, in the **Advanced Options** tab, check that **Units** is set to **Automatic**.

12. Check that **Axis Conversion** is set to **Up Axis: Y**.

13. Check that **FBX File Format** is set to the latest Maya version available for your Maya installation (Binary **FBX 2014**, as we write). Refer to the following screenshot to be sure you have set everything correctly:
14. Set a destination directory for the export, most likely the Asset\FBX directory in your Unity project.

15. You can now press Export Selection to have the FBX file sav ed and ready to be used in Unity.

How it works...

As you can see, there is quite a lot that can be tweaked with regard to exporting FBX files from Maya. What we offered here was a basic outline that will do for most cases. It is very likely that, depending on the specific need that would rise with your own project, you may need to use different settings on specific tabs. In such cases, we suggest you to refer to the official Unity documentation, where an entire section dedicated to exporting from Maya is available.

Anyway, at some point, you will have determined the optimal settings configuration for your project and won't need to reset them every time. Once you get your optimal configuration, the only change will be with regard to the Animation tab. Don't worry about it now; we'll get to importing animations soon.
There's more...

If you check Unity’s reference manuals (http://docs.unity3d.com/Manual/HOWTO-ImportObjectMaya.html), you may learn that Unity actually imports native Maya files, which means you can directly open Maya scenes (*.mb and *.ma files) in Unity. You can therefore ask, Why export FBX files into Unity at all?

There is more than one reason actually; one that is worth $4,000 is that FBX files are far more shareable than native Maya files. For example, to open a Maya scene in Unity, you must have Maya installed or the file won't open. If you expect to exchange files between people on their own PCs, you cannot assume that each one of them will have a Maya license. It will thus be safer to use formats that don't require additional costly software, as is the case with Maya (almost $4,000 per license).

That said, it is still quite useful to open Maya scenes in Unity. For example, in Unity you can immediately check the result of modifications made on a file in Maya without the need to export an FBX file with each new edit. Also, as already stated, Maya LT is going to export FBX file directly into Unity.

Configuring imported FBX files in Unity

Whenever an FBX file is imported into Unity, it is possible to edit some of its properties using Unity Inspector. Usually, these are operations that are required for setting things such as the correct scale of the model, the materials settings, as well as animations and other animation-related settings.

The following recipe provides useful hints on correctly specifying these settings using Unity Inspector.

Getting ready

For this recipe, we will basically resume from where we left the previous recipe. After having exported the model from Maya, launch Unity. By default, Unity always opens up to the last project you worked on.

How to do it...

1. From the FBX folder in your Assets directory, select the FBX file you just imported.
2. In the Inspector panel, let's begin with the Model tab. Depending on the actual unit system you set Maya with, you may need to set Scale Factor for Unity. By default, Unity scales down imported FBX files to one-hundredth of their original size. If you didn’t consider this when modeling the object in Maya, you may need to scale it up in Unity Inspector. In our case, we scale the model back to its original size, setting Scale Factor to 1.
3. Set both Normals and Tangents to Import from the drop-down menu.
4. Flag the Import Materials option.
5. From the drop-down menu, set Materials Naming to From Model's Materials.
6. For completeness, check that Material Search is set to Recursive-Up, which is the default setting.
7. Click on the Apply button on the bottom-right corner. Check the following screenshot to ensure you did everything right:

![Screenshot of Import Settings](image)

8. We can now move to the next tab, Rig. In the Avatar Definition field, select Create From This Model from the drop-down menu. This setting is important in order to animate the character. What we are stating here is that we want the Rig model for this character to be created from this actual FBX file.
9. In the **Root node** field, set the root node of your model, then click on **Apply**. Check the following screenshot for reference. As stated previously, we took care of naming our root node root earlier:

![Root Node Setting]

10. The last tab is called **Animations**. Since we are not importing the animations of our model with this specific FBX file, we will unflag the **Import Animation** option, as shown in the following screenshot:

![Animations Setting]

11. Click on **Apply** to update the setting.
How it works...

As already said with regard to the export settings in Maya, what we provided here was a default reference to set an FBX file imported in Unity. You may need to change some of these settings based on specific matters related to how you modeled your object in Maya and/or how you exported and/or planned to use it as a Unity game object.

We suggest you experiment with Inspector settings to get a wider grasp of the meaning of each setting and refer to the Unity documentation available at http://docs.unity3d.com/Manual/HOWTO-exportFBX.html when you have questions.

Exporting animations

It is very likely that models are enriched with animations to improve the quality of their behavior as game objects. In the following recipe, we will see how to export animations from Maya.

Getting ready

When exporting animations, two approaches are possible. One, which we will adopt in this book, is to keep things on separate files. With this approach, we export one FBX file for the model, the materials, and the rigging (as we did before), and one FBX file for each individual animation, such as idle, run, jump, and the like.

The other approach is to export everything on a single file. In this case, the exported FBX file will consist of the model, materials, rigging, and a timeline containing the frames for the entire animation set of the character. We will discuss this second approach later.

As usual, we took care of providing the required assets (animated Maya scenes for the model) in case you don't have any.

How to do it...

1. With the root node of the animated model selected in the outliner, navigate to File | Export Selection as usual.

2. Select FBX File Format in the exporting panel and name your file. If you want Unity to automatically read the animation name when it is imported, follow the official naming convention that requires the animation file to be named modelName@animationName (with @ before the animation name). Assuming that we are about to export the idle animation for our character, name the file modelName@idle.
3. Select the destination directory for the Animation file, which should be under Animations in your Unity project.

4. In the Options... panel, we need to make some adjustments to import the animation correctly. Under the Animation tab, flag the Animation option.

5. In the Bake Animation field, check that Bake Animation is flagged and that the starting and ending frame for the current animation are selected (this should be done automatically by Maya).

6. In the Deformed Models field, check that all the flags are selected. These settings may change, depending on the specific requirements of each individual model and animation set.

7. You don't need to change anything else with regard to the setting we defined to export the static model, so you can click on Export Selection.

How it works...

As we will see in the following recipe, where we will edit the settings of the imported animation in Unity, what we get is an FBX file containing an animation clip named idle, which represents the idle animation for our character.
As we said before, it is possible to trace the animation clips on a single file\timeline and export all character animations in one FBX file; in this case, however, additional operations are required when compared to the one-file-one animation technique.

With a single file containing all the clips, Unity is not capable, by default, to read the timeline imported from Maya and automatically detecting the individual clips, and you end up with a single, most likely long, timeline named track01 containing all the animations.

So you have to split the whole timeline by yourself, naming each clip individually and manually setting the start and end frames for each one of them in Unity Inspector.

If you think about a typical working pipeline with different people taking care of different operations, you may find your animators manually writing a text file of some sort with information regarding animation clip names and their reference frames. The animators then pass on this text file to someone responsible for taking care of importing animations in Unity. This last guy would also be responsible for typing the data found in the text file into Unity Inspector. A way of doing things that can easily lead to human errors, as you can imagine...

This is the reason why we prefer using separate FBX files for each animation clip.

It is also possible to automate the exporting process so it doesn't become a time-consuming activity, but you may need a programmer for this.

The idea is to script a piece of code in MEL to handle the job. MEL is the scripting scaffold of Maya: any operation you perform in the Maya editor has an equivalent instruction in MEL. Since performing hardcoding in the MEL scripting language would go beyond the scope of this book, we just provide a few references here for those interested. The list of MEL exporting commands is available at http://download.autodesk.com/us/fbx/20112/Maya/index.html.

An example MEL script, courtesy of James Kyle, is available at http://www.jameskyle.net/2013/03/maya-to-fbx-batch-export/.

There's more...

For those of you who are interested in automating the exporting process, there are ways, pretty elegant too, that require advanced programming skills.

One way is to create a Maya Embedded Language (MEL) script that reads the scene in Maya and exports what you need, based on the settings you define for the exporting process. MEL is the programming language behind Maya; any operation performed in Maya can be converted into a scripting instruction that will achieve exactly the same result. By using MEL, you can thus create a script that automatically exports animation clips into Unity on one or more FBX files, helping you save time (and reduce the risk of errors).
Importing 3D Models and Animations

Another option is to configure Maya to generate an XML file that describes the animation data stored in the timeline of a model (animation names, starting/ending frames, and the like), and then read this XML file from Unity to automatically create the required FBX files.

Both these approaches are very similar to using the post-processor to read custom attributes from Maya, as we discussed earlier.

If you’d like to go that way, you can refer to the following links:

Configuring imported animations in Unity Inspector

Before we end this chapter about Maya and Unity, there is one last step we must take care of configuring imported animations in Unity Inspector. This is the topic of our next recipe.

Getting ready

Again, we pick up from where we left the previous lesson. Open Unity and select the animated FBX file in the project panel.

How to do it...

1. With the animated FBX file selected in the project panel, go to Inspector and access the Model tab. Since we are only interested in the animation data stored in this file, we will basically unflag most of the options that we set when we imported the static model. Use the following screenshot as a reference and then click on Apply:
2. Now we move on to the **Rig** tab. As the animation clip stored in this file is to be used by the model we imported before, we need to set the **Avatar Definition** field as **Copy From Other Avatar**.
3. We also need to set the source avatar. Click on the small button to the right of the **Source** field and add the avatar created from the static model to it, as shown in the following screenshot:

4. Click on **Apply** and select the last tab, **Animations**.

5. In this tab, there is much to do. First of all, flag the **Import Animations** option, which will make a group of related options that depend on this option visible:
6. Scroll down the panel and look for the Root Transform Rotation. Flag Bake Into Pose and select Root Node Rotation from the drop-down menu.

7. In the Root Transform Position (Y) group, flag Bake Into Pose and check that Root Node Position (Y) is flagged. In the Root Transform Position (XZ) group, flag Bake Into Pose and ensure that Root Node Position isn't flagged.

8. Notice that this is an idle animation and, as such, it should be set on looping by flagging the Loop Time option. We didn't do this intentionally to provide a more general example.

9. Click on Apply to end configuring the animation.
How it works...

Let's begin with a few words on the topic of root motion, as Unity Inspector displays several options about it.

Root motion has to do with controlling the actual position of the mesh with regard to its collider while animations are being played. Most animations happen in place, meaning that the mesh and the collider don't actually move around as the character is animated. This may be the case for examples of walk and run animations.

There are other animations though that require the mesh and collider to actually move or rotate in the 3D world as the animation is played. Actions such as strafing, jumping, and other in-game specials require the mesh and the collider to change their position and rotation in the 3D world as the animation is performed.

By setting the properties of Root Rotation and Root Position groups, you control whether the collider should rotate or move with the mesh during specific animation clips.

In a walk animation, the start and stop root orientations and positions in world space are identical. In such cases where animations are acknowledged by Unity with a green light in Unity Inspector, it is recommended to flag the Bake into Pose option in the Root Transform Rotation group and set Root Node Rotation from the drop-down menu. Also, flag Bake Into Pose in the Root Transform Position (Y) group and set Root Node Position from the menu.

As for the Root Transform Position (XZ) group of options, the manual recommends using them for long, idle animations, where the repetition of many frames could lead the mesh to drift from its collider in the long run. In such cases, it is recommended that you bake the position on the x and z axes as well.

Latin speakers used to say "Repetita iuvant," meaning repetition helps. The settings provided here are to be intended as general, default settings that may not fit into any situation. For example, you may find yourself having to set Scale Factor in the Model tab for the animated FBX file too in order to prevent the animation from not fitting the model rig. Alternatively, you may need to tweak the Root Transform Rotation and Root Transform Position settings differently, depending on how you rigged\animated your model with specific animation clips. As usual, we recommend that you refer to the Unity documentation at http://docs.unity3d.com/Manual/AnimationsImport.html whenever you have questions.
The Rig tab in Unity Inspector displays an option that we didn't mention here: Animation Type. You configure this option to select whether you want your character to be animated as a humanoid or a generic object, as it could be with a vehicle, a ball, or whatever.

Also, when a rigged FBX file is imported into a project, Unity automatically creates a so-called avatar out of it. This task is generally performed automatically by Mecanim, the built-in tool responsible for animation setting and management; we will discuss this in detail in Chapter 3, Animating a Game Character.

Mecanim has routines that examine the skeletal configuration of a rigged model imported in Unity and then recompute it into a general template it can interpret. This template is called the avatar. Once an avatar has been created, other animation clips can be targeted to that same avatar. This is how multiple animation clips are linked to a single character model in Unity.

In the recipe, we configured the model scale and material in the Model tab only for the static version of the model; in the Rig tab, we set the Avatar Definition option to Create for this model. For animated clips, on the other hand, we didn't configure the Model tab, and we set Avatar Definition to Copy From Other Avatar with a reference to the static avatar. This assumes Mecanim succeeds in interpreting the rigged model.

There are cases where for some reason Mecanim is unable to correctly compute the avatar; alternatively, you may just want to make changes on your own. In such cases, it is possible to manually edit the avatar in Unity Inspector and check that all bones required by Mecanim are correctly named and in place.

A recipe about setting up the avatar would go beyond the scope of this book, as it has a lot to do with modeling and rigging techniques, which are beyond the scope of this book.

Refer to the Unity manual at http://docs.unity3d.com/Manual/AvatarCreationandSetup.html. Besides this, if you are interested in delving more into this matter, we suggest the many tutorials available on websites about game development and 3D graphics. For example, digital tutors offer many exceptional tutorials about Unity in general and Mecanim in particular; we recommend you check them out.
Where to buy this book
You can buy Unity 2D Game Development Cookbook 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.