HOUR 17

Animations

What You'll Learn in This Hour:

- The requirements for animation
- The different types of animations
- How to create animations in Unity

In this hour, you'll learn about animations in Unity. You'll start by learning exactly what animations are and what is required for them to work. After that, you'll look at the different types of animations. From there, you'll learn how to create your own custom animations with Unity's animation tools.

Animation Basics

Animations are premade sets of visual motions. In a 2D game, an animation involves several sequential images that can be flipped through very quickly to give the appearance of movement (much like an old-fashioned flip book). Animation in a 3D world is much different. In 3D games, you use models to represent game entities. You cannot simply switch between models to give the illusion of motion. Instead, you have to actually move the parts of the model. Doing so requires both a rig and an animation. Furthermore, animations can also be thought of as "automations"; that is, you can use animations to automate object changes such as the size of colliders, the value of script variables, or even the color of materials.

The Rig

Achieving complex animated actions, such as walking, without a rig is impossible (or impossibly difficult). Without a rig, the computer has no way of knowing which parts of a model are supposed to move and how they are supposed to move. So, what exactly is a rig? Much like a human skeleton (see Figure 17.1), a rig dictates the parts of a model that are rigid, which are often called *bones*. It also dictates which parts can bend; these bendable parts are called *joints*.

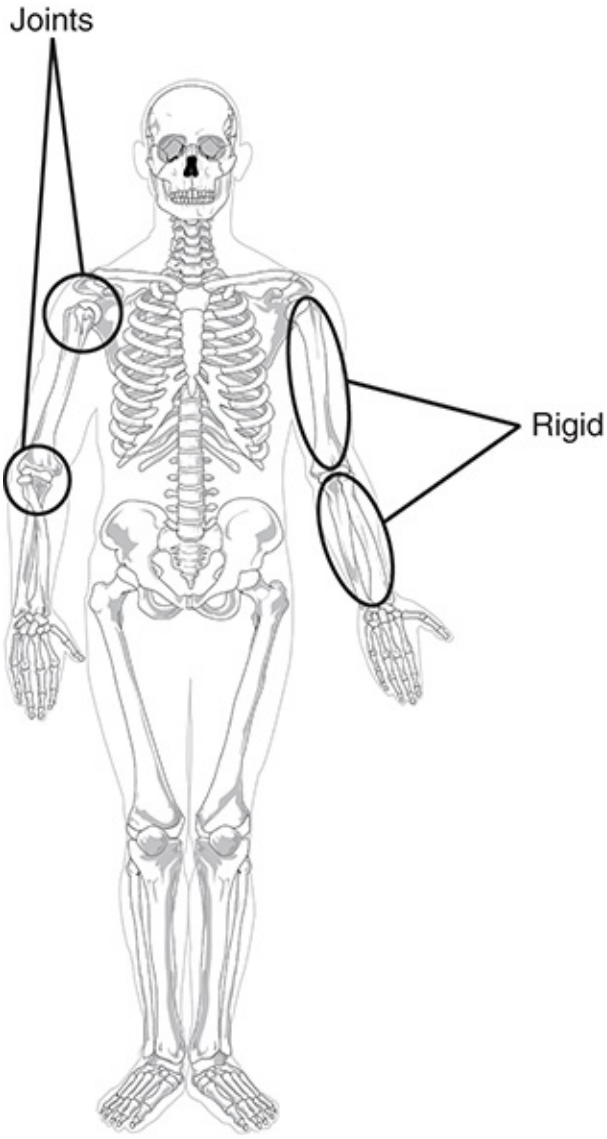

FIGURE 17.1
The skeleton as a rig.

The bones and joints work together to define a physical structure for a model. It is this structure that is used to actually animate the model. It is worth noting that 2D animations, simple animations, and animations on simple objects do not

2D animations, simple animations, and animations on simple objects do not require any particular or complex rig.

The Animation

Once a model has a rig (or not, in the case of simple animations), it can be given an animation. On a technical level, an animation is just a series of instructions for a property or a rig. These instructions can be played just like a movie. They can even be paused, stepped through, or reversed. Furthermore, with a proper rig, changing the action of a model is as simple as changing the animation. The best part of all is that if you have two completely different models that have the same rigging (or different but similar rigging, as you will discover in Hour 18, “Animators”), you can apply the same animations to each of them identically. Thus, an orc, a human, a giant, and a werewolf can all perform exactly the same dance.

NOTE

3D Artists Wanted

The truth about animation is that most of the work is done outside programs like Unity. Generally speaking, modeling, texturing, rigging, and animations are all created by professionals, known as 3D artists, in programs such as Blender, Maya, and 3ds Max. Creating these assets requires a significant amount of skill and practice. Therefore, their creation is not covered in this text. Instead, this book shows you how to build interactive experiences in Unity by putting together already made assets. Remember that there is more to making a game than simply putting together pieces. You may make a game, but artists make a game look good!

Animation Types

So far you’ve read about things like animations, rigs, and automations. These terms may seem a bit nonsensical at this point, and you may be wondering how they relate and what exactly you need in order to use animations in a game. This section looks at the various types of animation and helps you understand how they work so that you can begin making them.

2D Animations

In a sense, 2D animations are the simplest type of animations. As explained

In a sense, 2D animations are the simplest type of animations. As explained earlier in this hour, a 2D animation functions very much like a flip book (or an animated cartoon or even a film-based movies). The idea behind a 2D animation is that images are presented in sequential order at a very fast pace, tricking the eye into seeing motion.

Setting up 2D animations within Unity is very easy, but modifying them is more difficult. The reason is that 2D animations required art assets (the images being shown) in order to work. Any changes to an animation require you (or an artist) to make changes to the images themselves in other software, such as Photoshop or Gimp. It isn't possible to make changes to the images themselves within Unity.

▼ TRY IT YOURSELF

Slicing a Sprite Sheet for Animation

In this exercise, you will prepare a sprite sheet for animation. The project created in this exercise will be used later, so be sure to save it. Follow these steps:

1. Create a new 2D project.
2. Import the RobotBoyRunSprite.png image from the 2D assets package. (While this image has already been prepared for animation—it is an animated character in the 2D assets package—you examine it in this exercise as a review.) You can do this by selecting **Assets > Import Package > 2D** and importing *only* the RobotBoyRunSprite.png asset (see Figure 17.2). Alternatively, you can find the RobotBoyRunSprite.png asset in the book files for Hour 17.

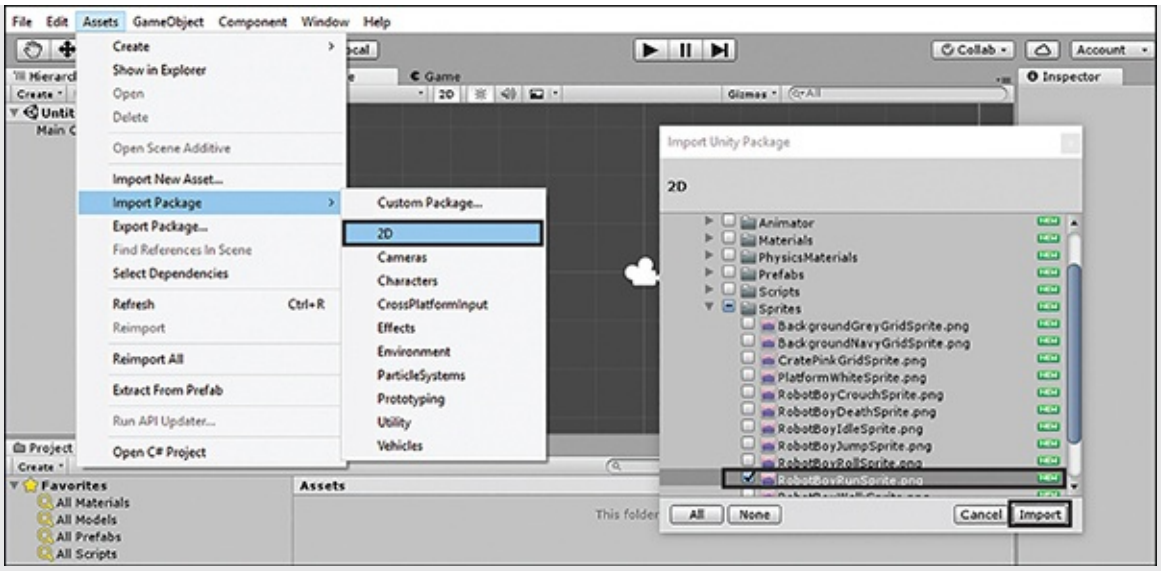

FIGURE 17.2
Importing a sprite sheet.

3. Select the newly imported RobotBoyRunSprite.png file in the Project view.

4. In the Inspector view, ensure that Sprite Mode is set to Multiple and then click **Sprite Editor**. In the upper-left corner, click **Slice** and then choose **Grid by Cell Size** as the type. Notice the grid sizes and the resulting sprite slices (see Figure 17.3).

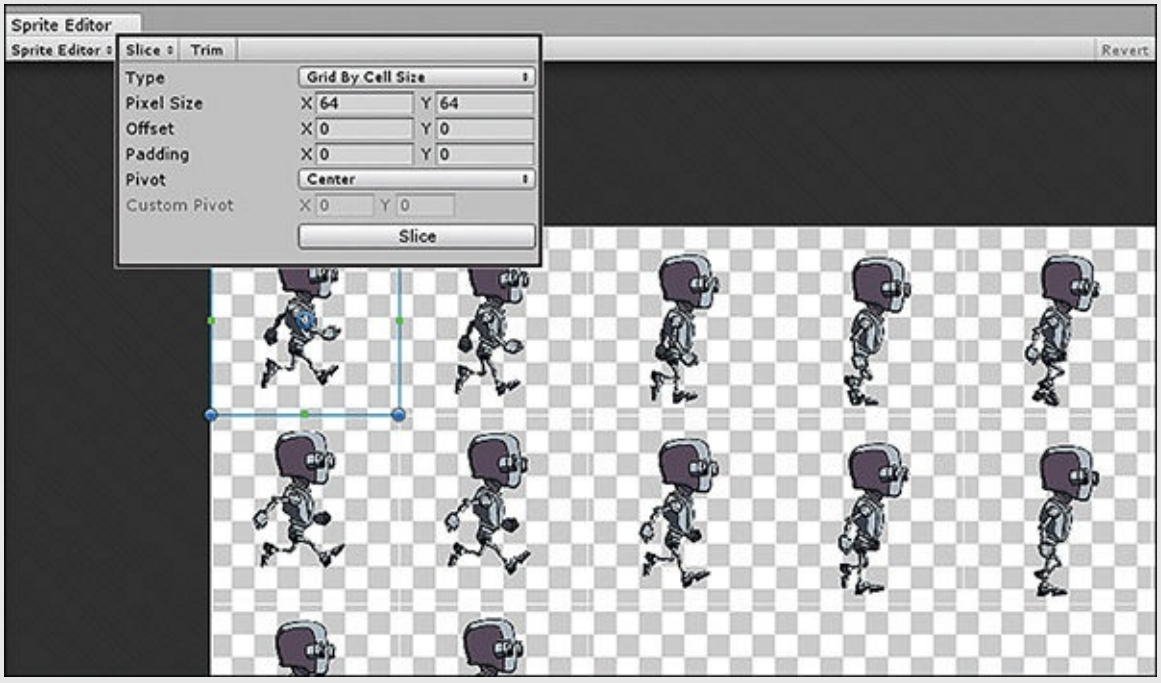

FIGURE 17.3

FIGURE 17.3
Slicing the sprite sheet.

5. Close the Sprite Editor window.

Now that you have a series of sprites, you can turn them into an animation.

NOTE

Reinventing the Wheel

This hour you are using assets from Unity's 2D asset package. You may have noticed that these assets are already animated, which means you are doing work that has already been done. This way, you can see the work that originally went into animating these characters. Better yet, you can explore the completed assets in the 2D assets package to figure out how they are put together and to see how more complex examples are achieved.

Creating an Animation

You've prepared your assets, so you are now ready to turn them into an animation. There is a simple way to do this, and there is also a complicated way to accomplish this. The complicated way involves creating an animation asset, specifying sprite renderer properties, adding key frames, and providing values. Because you haven't learned how to do all that yet (although you will in the next section), let's go with the simple route. Unity has a strong automated workflow, and you will tap into that to create your animation.

▼ TRY IT YOURSELF

Creating an Animation

Use the project you created in the Try It Yourself "Slicing a Sprite Sheet for Animation" and follow these steps to make an animation:

1. Open the project you created in the Try It Yourself "Slicing a Sprite Sheet for Animation."
2. Locate the RobotBoyRunSprite asset in the Project view. Expand the sprite drawer (by clicking the small arrow on the right side of the sprite) to see all the subsprites.

3. Select all the sprites from that sprite sheet by selecting the first sprite and then selecting the last sprite while holding the **Shift** key. Then drag all the frames into the Scene view (or Hierarchy view; either works) and let go (see Figure 17.4).

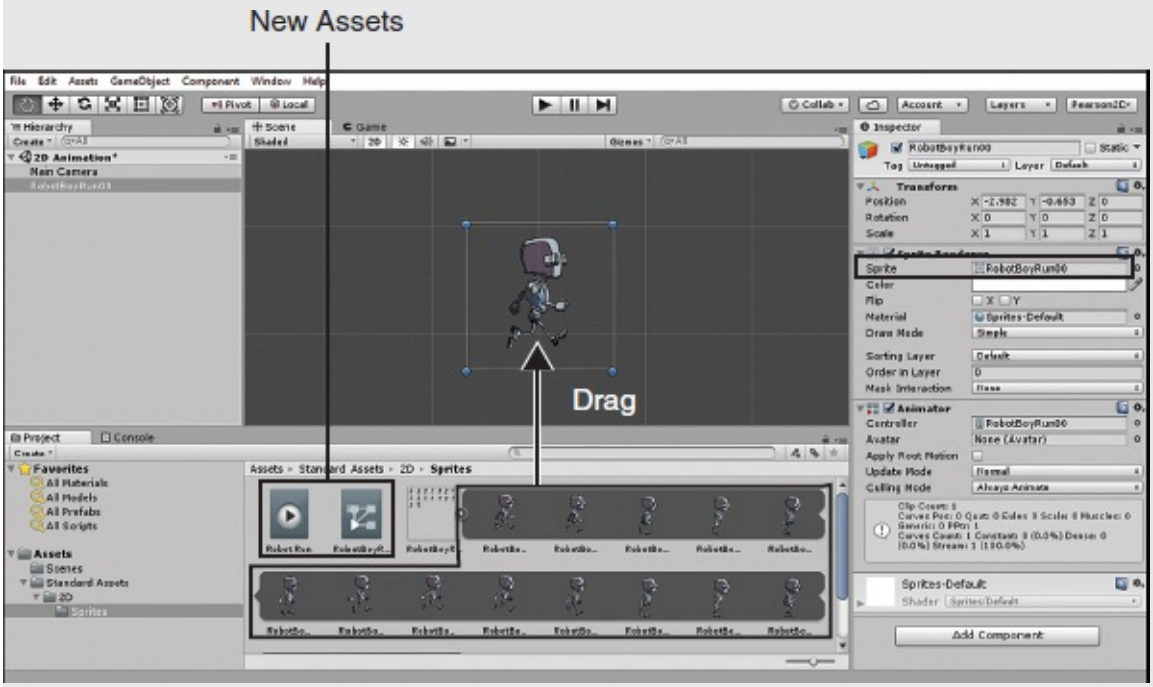

FIGURE 17.4
Creating the animation.

4. If you are prompted with a Save As dialog, choose a name and location for your new animation. If you aren't prompted, two new assets are created in the same folder as the sprite sheet. Either way, Unity has now automated the process of creating an animated sprite character in your scene. The two assets created (the animation asset and another asset called an animator controller) are covered in greater detail in Hour 18.

5. Run your scene, and you see the animated robot boy playing its running sequence. In the Inspector view, you can see the Sprite property of the Sprite Renderer component cycling through the various frames of the animation.

That's it! 2D animations really are very easy to create.

NOTE

NOTE

More Animations

You now know how to make a single 2D animation, but what if you want a series of animations (such as walk, run, idle, jump, and so on) that all work together? Luckily, the concepts you've just learned continue to work in more complex scenarios. Getting the animations to work together, however, requires greater understanding of Unity's animation system. You will learn that system in great detail in Hour 18, where you will be using imported 3D animations. Just remember that anywhere you can use a 3D animation, you can also use any other type of animation. Thus, the concepts you'll learn in Hour 18 are equally applicable to 2D and custom animations.

Animation Tools

Unity has a set of animation tools built in that you can use to create and modify animations without leaving the editor. You already used them unknowingly when you made a 2D animation, and now it is time to dig in and see just what you can do.

Animation Window

To begin using Unity's animation tools, you need to open the Animation window. You can do this by clicking **Window > Animation** (not Animator). This opens a new view that you can resize and dock like any other Unity window. Generally, it is a good idea to dock this window so you can use it and other parts of the editor without their overlapping. Figure 17.5 shows the Animation window and its various elements.

Note that for the purpose of this figure, the 2D animated sprite from the previous exercise is selected. See if you can identify how Unity used the sprites you dragged into the scene to make the animation you saw when you ran your project. Table 17.1 runs through some of the most important parts of the Animation window.

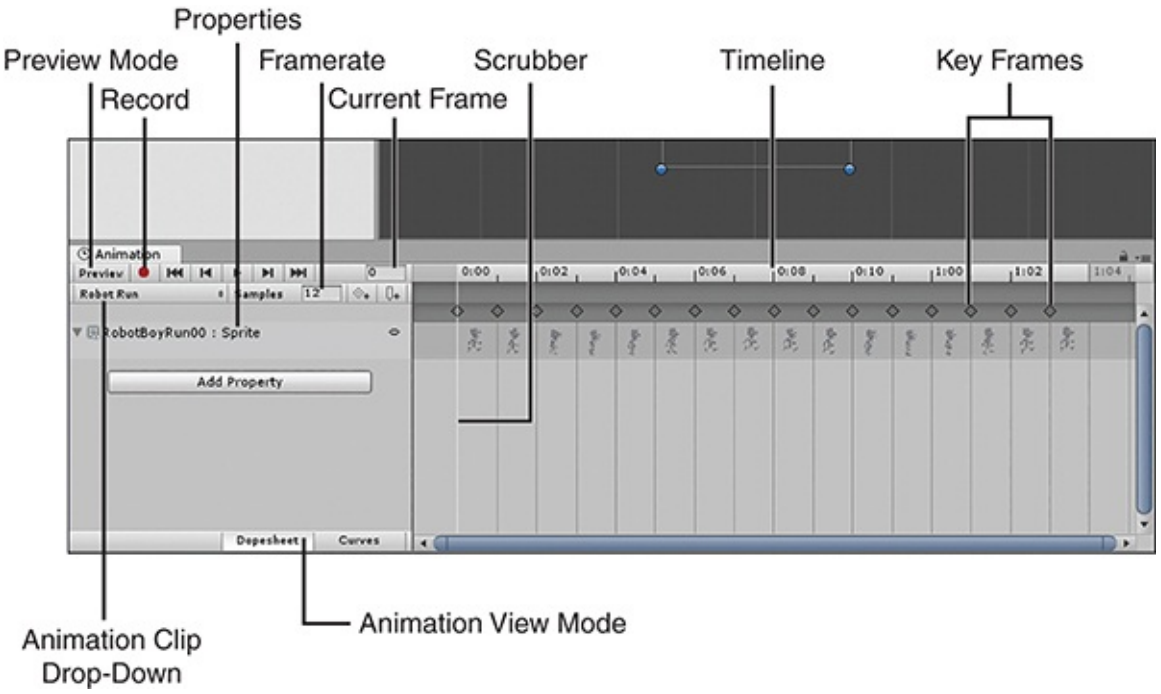

FIGURE 17.5
The Animation window.

TABLE 17.1 Important Parts of the Animation Window

Component	Description
Properties	The list of the component properties modified by this animation. Expanding a property allows you to see the value of that property based on the value of the scrubber on the timeline.
Samples	The number of animation frames available in one second of animation. This is otherwise known as the *framerate*.
Timeline	The visual representation of changes to properties over time. The timeline allows you to specify when values change.
Key frames	Special points on the timeline. A key frame allows you to specify the desired property value at a particular point of time. Regular frames (not shown) also contain values, but you cannot directly control them.
Add key frame	A button that allows you to add a key frame to the timeline for the selected property at the position of the scrubber.
Scrubber	The red line, which allows you to choose the point on the timeline that you would like to edit. (*Note:* If you aren't in

	Record mode, the scrubber line is white.)
Current frame	An indicator of what frame the scrubber is currently on.
Preview	A toggle that allows you to preview the animation. This is enabled automatically if you play the animation using the play controls in the Animation view.
Record Mode	A toggle that puts you into and out of Record mode. In Record mode, any changes you make to the selected object are recorded into the animation. Use with caution!
Animation clip drop-down	A menu that allows you to change between the various animations associated with the selected object as well as create new animation clips.

Hopefully, looking at this window gives you more insight into how your 2D animation works. Figure 17.5 shows that an animation called Robot Run was created. This animation has a sample rate of 12 frames per second. Each frame of the animation contains a key frame, which sets the Sprite property of the object's Sprite Renderer to a different image, thus causing your character to appear to change. All this (and more) was done automatically.

Creating a New Animation

Now that you're familiar with the animation tools, you're ready to put them to use. Creating animations involves placing key frames and then choosing values for them. The values of the frames in between all the key frames are calculated for you to smoothly transition between them—for example, to make an object move up and down. You could easily achieve this by choosing to modify the position of the object's transform and adding three key frames. The first would have a "low" y axis value, the second would be higher, and the third would be the same as the first. As a result, the object would bounce up and down. This is all fairly conceptual, so work through the following example to get a better feel for the process.

▼ TRY IT YOURSELF

Making an Object Spin

In this exercise you are going to make an object spin. You will be doing

In this exercise you are going to make an object spin. You will be doing this to a simple cube, but in reality you could apply this animation to any object you wanted, and the result would be the same. Be sure to save the project you create here for later use:

1. Create a new 3D project.
2. Add a cube to the scene and ensure that it is positioned at (0, 0, 0).
3. Open the Animation window (by selecting **Window > Animation**) and dock it in your editor.
4. With the cube selected, you should see a Create button right in the middle of the Animation view. Click it, and when you are prompted to save your animation, do so and name it **ObjectSpinAnim**.
5. Click the **Add Property** button in the Animation view and then click the + icon next to Transform > Rotation (see Figure 17.6).

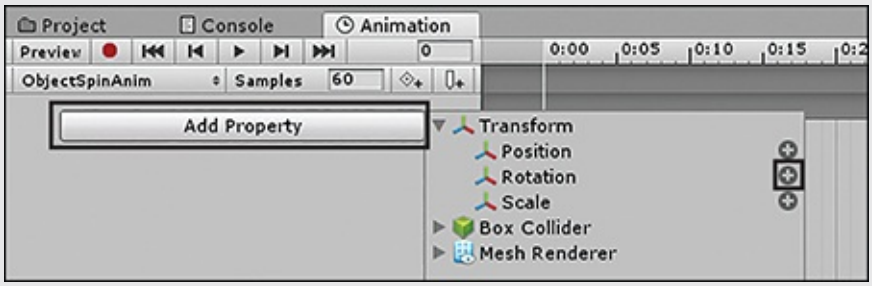

FIGURE 17.6
Adding the Rotation property.

You should now have two key frames added to your animation: one at frame 0 and another at frame 60 (or “1 second”; see the following tip for more info about the timing). If you expand the Rotation property, you can see the properties of the individual axes. Furthermore, selecting a key frame allows you to see and adjust the values of that key frame.

6. Move the scrubber bar over the ending key frame (by clicking and dragging on the timeline) and then set the value of the Rotation.y property to **360** (see Figure 17.7). Even though the starting value of 0 and ending value of 360 are technically the same, doing this causes the cube to rotate. Play your scene to see the animation.

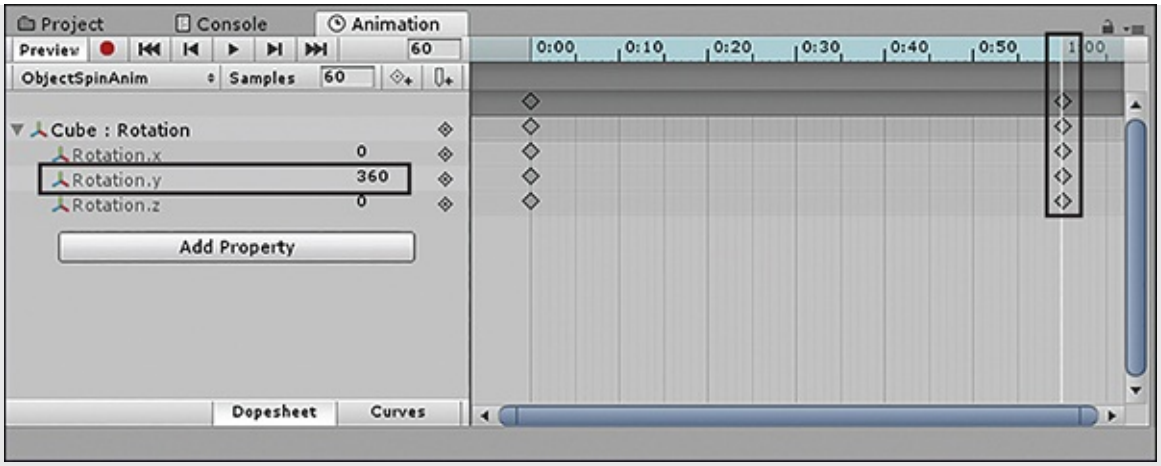

FIGURE 17.7
Adding the Rotation.y value.

7. Save your scene because you will be using it later.

TIP

Animation Timing

The values on the timeline might seem a bit odd at first glance. In fact, reading the timeline helps you understand the sample rate of the given animation clip. Based on a default sample rate of 60 frames per second, the timeline would count frames 0 to 59 and then, instead of frame 60, it has 1:00 (for 1 second). Therefore, a time of 1:30 would mean 1 second and 30 frames. This is easy when you have a 60-frames-per-second animation, but what happens with a 12-frames-per-second animation? Then the counting would be "1, 2, 3,…11, 12, one second." To put it another way, you could see "0:10, 0:11, 1:00, 1:01, 1:02,…1:10, 1:11, 2:00,…" The most important thing to take away from this is that the number preceding the colon is the seconds, and the number following it is the frames.

TIP

Moving the Timeline

If you would like to zoom out or pan into the timeline to see more frames or look at different frames, you can do so easily. The window uses the same navigation style as a Scene view in 2D mode. That is, scrolling the mouse wheel zooms the timeline in and out, while holding **Alt** (**Option** on Mac) and

dragging pans around.

Record Mode

Although the tools you have used so far have been very easy to use, there are even easier ways to work with animations. An important element in the animations tool is *Record mode* (refer to Figure 17.5 for its location). In Record mode, any changes you make to an object are recorded into the animation. This can be a very powerful way to make quick and accurate additions to an animation. It can also be very dangerous. Consider what would happen if you forgot you were in Record mode and made a bunch of changes to an object. Those changes would be recorded into the animation and repeated over and over whenever it was played. It is therefore generally advisable to always ensure that you are not in Record mode when you don't want to use it.

This is quite a scary warning for a tool you haven't even used yet, but don't worry. It really isn't that bad (and it is super powerful to boot). With a little discipline, it is a fantastic tool. In the worst-case scenario, if Unity records something for you that you don't want, you can manually delete it from the animation yourself.

TRY IT YOURSELF

Using Record Mode

Follow these steps to make the cube you created in the Try It Yourself "Making an Object Spin" change color while it spins:

1. Open the scene with the spinning cube from the Try It Yourself "Making an Object Spin." Create a new material called CubeColor and apply it to the cube (so you can change the color).
2. Ensure that the cube is selected, and then open the Animation window. You should see the rotation animation you previously created; if you don't, make sure the cube is selected.
3. Enter Record mode by clicking the Record Mode button in the Animation window. The Rotation property in the Inspector view turns red, signifying that the rotation values are driven by the animation. Click and drag along the timeline to move the scrubber and position the scrubber at frame 0:00 (see Figure 17.8).

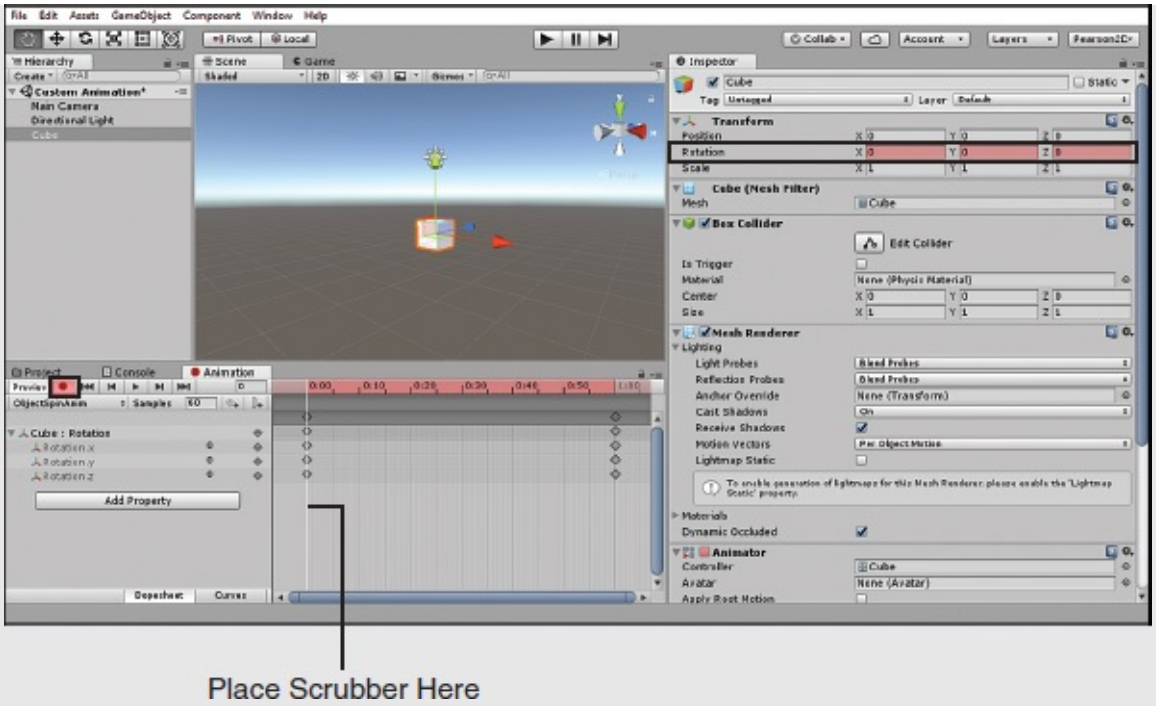

FIGURE 17.8
Getting ready to record.

4. In the Inspector view, locate the CubeColor material on the cube and change its Albedo setting to red. Notice that a new property and key frame have been added to the Animation window. If you expand the new property, you see the r, g, b, and a values of the color at that key frame (1, 0, 0, 1).

5. Move the scrubber further down the timeline and change the color in the Inspector again. Repeat this step as many times as you'd like. If you'd like the animation to loop smoothly, be sure that the last key frame (at position 1:00, so it synchronized with the rotation as well) is the same color as the first key frame (see Figure 17.9).

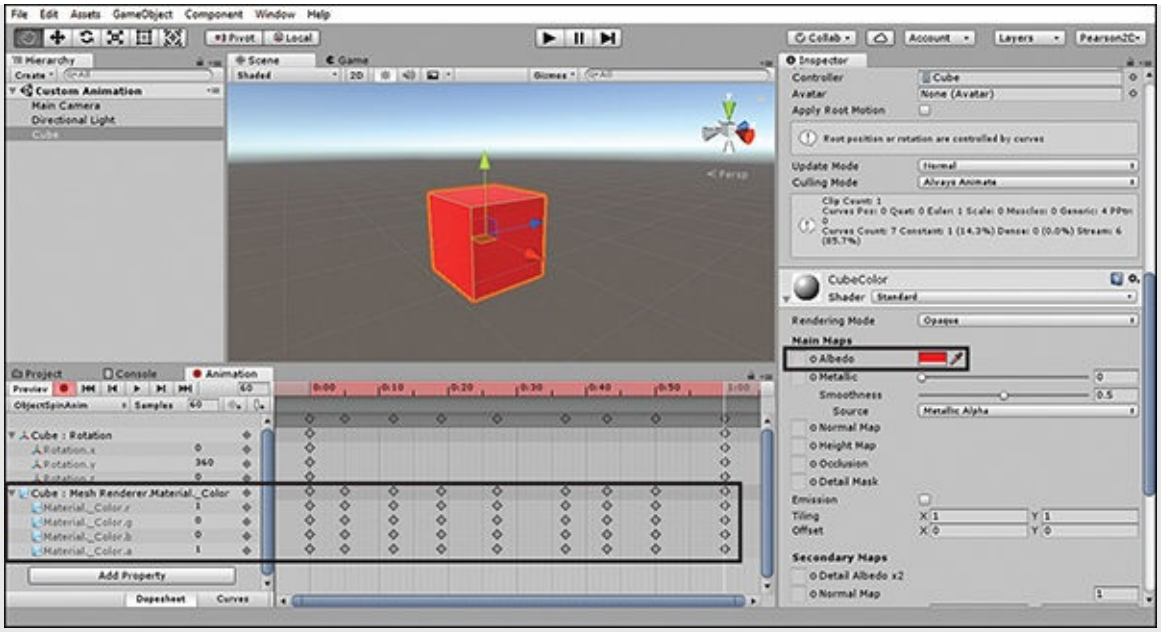

FIGURE 17.9
Recording the color values.

6. Play your scene to see your animation run in the Game view. Note that playing a scene brings you out of Record mode (which is handy because you are done using it). You should now have a spinning multicolored cube in your scene.

The Curves Editor

The last tool you are going to look at in this hour is the Curves Editor. So far, you have been in the Dopesheet view, which is the view with the key frames listed and itemized in a flat fashion. You may have noticed that while you drive the values of the key frames, you don't control the values in between. If you wondered how the values are determined, wonder no more. Unity blends the values (in a process called *interpolation*) between key frames to create smooth transition. In the Curves Editor, you can see what that looks like. To enter the Curves Editor, simply click the button labeled **Curves** at the bottom of the Animation view (see Figure 17.10).

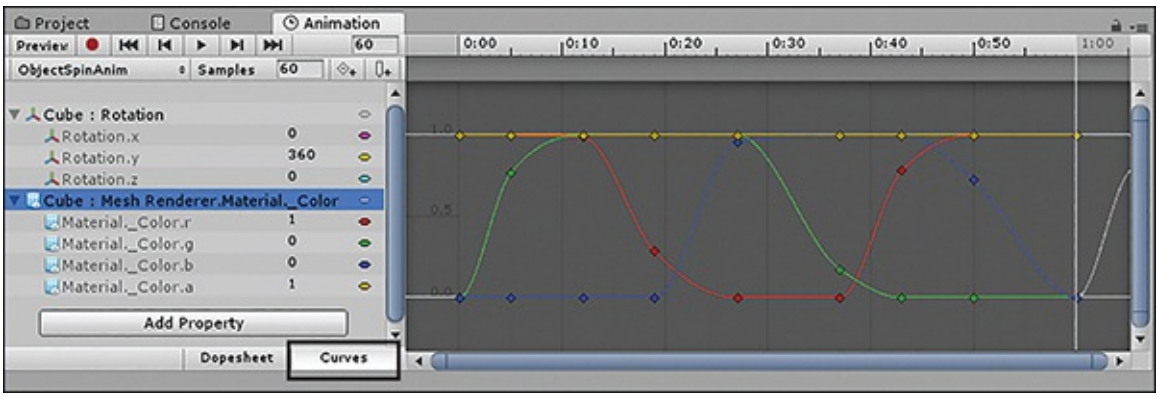

FIGURE 17.10
The Curves Editor.

In this mode, you can toggle which values you want to see by clicking their properties on the left. In the Curves Editor, you can see exactly how values are being transitioned between the key frames. You can drag a key frame to change its value or even double-click the curve to create a new key frame at the point where you clicked. If you don't like how Unity is generating the value in between the key frames, you can right-click a key frame and choose **Free Smooth**. Unity then gives you two handles that you can move to change how values are smoothed. Feel free to play around and see what sort of craziness you can create with the curves.

▼ TRY IT YOURSELF

Using the Curves Editor

You may have noticed that the cube from the Try It Yourself "Using Record Mode" doesn't spin smoothly and instead has a slow start-and-stop motion. In this exercise you will modify the animation to give the cube a smooth spinning motion. Follow these steps:

1. Open the scene with your spinning cube from the Try It Yourself "Using Record Mode." In the Animation view, click the **Curves** button to switch to the Curves Editor (refer to Figure 17.10).
2. Click the **Rotation.y** property to show its curve in the timeline. If the curve is small or doesn't fit in the window, simply move your mouse cursor over the timeline and press the **F** key.
3. Straightening out the rotation curve will give your cube a nice smooth animation (see Figure 17.11), so right-click the first keyframe (at time

0:00) and select **Auto**. Do the same for the last keyframe.

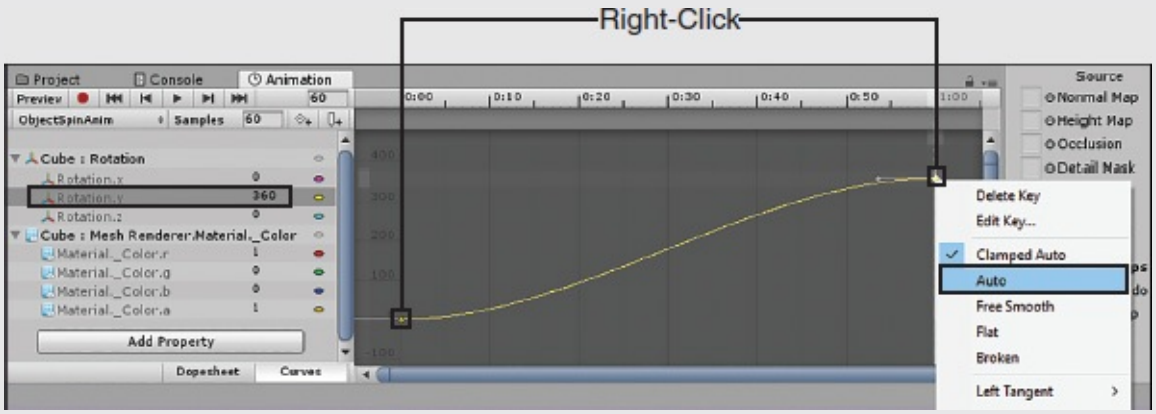

FIGURE 17.11
Modifying the rotation curve.

4. Now that the curve is straightened out, enter Play mode to see the modified cube animation.

Summary

In this hour you were introduced to animations in Unity. You started by looking at the basics of animations, including rigging. From there, you learned about the various types of animations in Unity. After that, you began creating animations, starting with 2D animations. Then you created custom animations both manually and in Record mode.

Q&A

Q. Can animations be blended?

A. Yes, they can. You can blend animations with the Unity Mecanim system, as discussed in Hour 18.

Q. Can any animation be applied to any model?

A. Only if they are rigged exactly the same or if the animation is a simple one that doesn't require a rig. Otherwise, the animations may behave very strangely or just may not work at all.

Q. Can a model be rerigged in Unity?

A. Sort of, as you will see in Hour 18.

Workshop

Take some time to work through the questions here to ensure that you have a firm grasp of the material.

Quiz

1. The “skeleton” of a model is known as what?
2. Which animation type flips through images quickly?
3. What is the name for a frame of animation that has an explicit value?

Answers

1. Rig or rigging
2. 2D animation
3. Key frame

Exercise

This exercise is a sandbox type of exercise. Feel free to take some time and get used to creating animations. Unity has a very powerful toolset, and it certainly pays to be familiar with it. Try completing the following:

- Make an object fly around a scene in a large arc.
- Make an object flicker by cycling its renderer on and off.
- Change the properties of an object’s scale and material to make it appear to morph.