HOUR 21
Audio

What You'll Learn in This Hour:

- The basics of audio in Unity
- How to use audio sources
- How to work with audio via scripts
- How to use audio mixers

In this hour, you'll learn about audio in Unity. You'll start by learning about the basics of audio. From there, you'll explore the audio source components and how they work. You'll also take a look at individual audio clips and their role in the process. You'll then learn how to manipulate audio in code before wrapping up the hour with audio mixers.

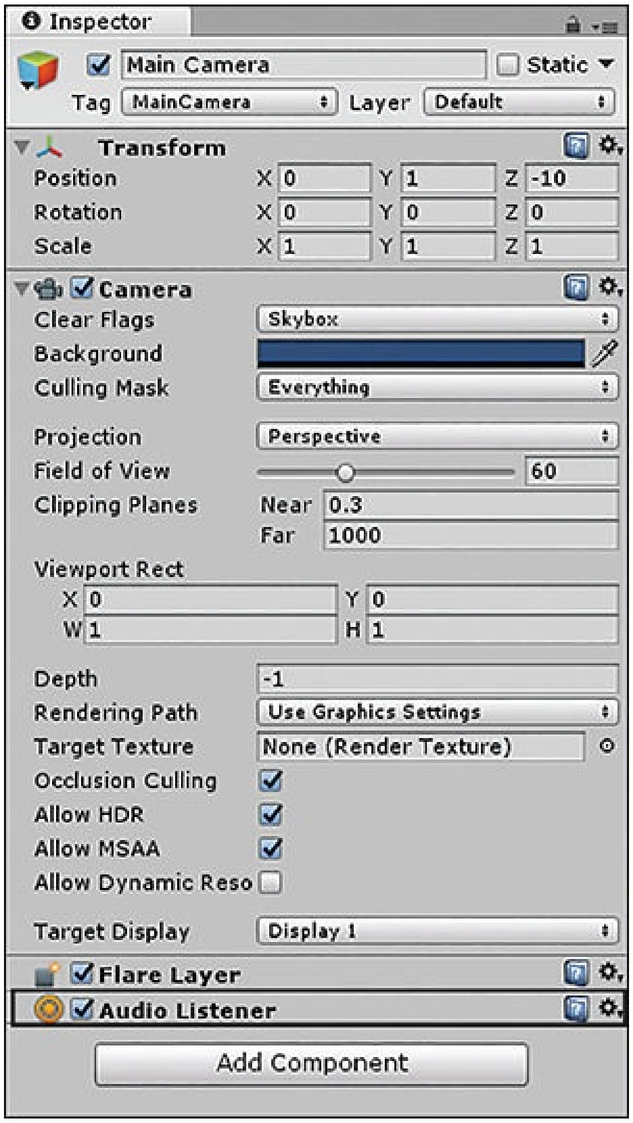
Audio Basics

A large part of any experience involves the sounds of that experience. Imagine adding a laugh track to a scary movie. All of a sudden, what should be a tense experience would become a funny one. With video games, most of the time players don't realize it, but the sound is a very large part of the overall gameplay. Audio cues like chimes mark when a player unlocks a secret. Roaring battle cannons add a touch of realism to a war simulation game. Using Unity, implementing amazing audio effects is easy.

Parts of Audio

For sounds to work in a scene, you need three things: the audio listener, the audio source, and the audio clip. The *audio listener* is the most basic component of an audio system. The listener is a simple component whose sole responsibility is “hearing” the things that are happening in a scene. Listeners are like ears in a game world. By default, every scene starts with an audio listener attached to the Main Camera (see Figure 21.1). There are no properties available for the audio listener, and there is nothing you need to do to make it work.

It is a common practice to put the audio listener on whatever game object represents the player. If you put an audio listener on any other game object, you need to remove it from the Main Camera. Only a single audio listener is allowed per scene.


Inspector
☑ Main Camera ☐ Static
Tag MainCamera Layer Default

Transform
Position X 0 Y 1 Z -10
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

☑ Camera
Clear Flags Skybox
Background
Culling Mask Everything
Projection Perspective
Field of View 60
Clipping Planes Near 0.3
Far 1000
Viewport Rect
X 0 Y 0
W 1 H 1
Depth -1
Rendering Path Use Graphics Settings
Target Texture None (Render Texture)
Occlusion Culling ☑
Allow HDR ☑
Allow MSAA ☑
Allow Dynamic Reso ☐
Target Display Display 1

☑ Flare Layer
☑ Audio Listener

Add Component

FIGURE 21.1

The audio listener

The audio listener.

The audio listener listens for sound, but it is the *audio source* that actually emits the sound. This source is a component that can be put on any object in a scene (even the object with the audio listener on it). There are many properties and settings involved with the audio source; they are covered in their own section later in this hour.

The last item required for functioning audio is the *audio clip*. Just as you would assume, the audio clip is the sound file that actually gets played by an audio source. Each clip has some properties that you can set to change the way Unity plays them. Unity supports the following audio formats: .aif, .aiff .wav, .mp3, .ogg, .mod, .it, .s3m, and .xm.

Together, these three items—audio listener, audio source, and audio clip—give your scene an audio experience.

2D and 3D Audio

One concept to be aware of with audio is the idea of 2D and 3D audio. 2D audio clips are the most basic types of audio. They play at the same volume regardless of the audio listener’s proximity to the audio source in a scene. 2D sounds are best used for menus, warnings, soundtracks, or any audio that must always be heard exactly the same way. The greatest asset of 2D sounds is also their greatest weakness. Consider if every sound in your game played at exactly the same volume, regardless of where you were. It would be chaotic and unrealistic.

3D audio solves the problems of 2D audio. These audio clips feature something called *rolloff*, which dictates how sounds get quieter or louder depending on how close the audio listener gets to the audio source. In sophisticated audio systems, like Unity’s, 3D sounds can even have a simulated Doppler effect (more on that later). If you are looking for realistic audio in a scene full of different audio sources, 3D audio is the way to go.

The dimensionality of each audio clip is managed on the audio source that plays the clip.

Audio Sources

As mentioned earlier in this hour, the audio sources are the components that actually play audio clips in a scene. The distance between these sources and the listeners determines how 3D audio clips sound. To add an audio source to a game object, select the desired object and click **Add Component > Audio > Audio Source**.

The Audio Source component has a series of properties that give you a fine level of control over how sound plays in a scene. Table 21.1 describes the various properties of the Audio Source component. In addition to these properties, the 3D Sound Settings section, covered later in this hour, has a number of settings you can apply to 3D audio clips.

TABLE 21.1 Audio Source Component Properties

Property	**Description**
Audio Clip	Specifies the actual sound file to play.
Output	(Optional) Makes it possible to output the sound clip to an audio mixer.
Mute	Determines whether the sound is muted.
Bypass Effects	Determines whether audio effects are applied to this source. Selecting this property turns off effects.
Bypass Listener Effects	Determines whether audio listener effects are applied to this source. Selecting this property turns off effects.
Bypass Reverb Zones	Determines whether reverb zone effects are applied to this source. Selecting this property turns off effects.
Play On Awake	Determines whether the audio source will begin playing the sound as soon as the scene launches.
Loop	Determines whether the audio source will restart the audio clip once it has finished playing.
Priority	Specifies the importance of the audio source. 0 is the most important, and 255 is the least important. Use 0 for music so it always plays.
Volume	Specifies the volume of the audio source, where 1 is the equivalent of 100% volume.
Pitch	Specifies the pitch of the audio source.
Stereo Pan	Sets the position in the stereo field of the 2D component of the

	sound.
Spatial Blend	Sets how much the 3D engine has an effect on the audio source. Use this to control if a sound is 2D or 3D.
Reverb Zone Mix	Sets the amount of the output signal that gets routed to the reverb zones.

NOTE

Audio Priorities

Every system has a finite number of audio channels. This number is not consistent and depends on many factors, such as the system's hardware and operating system. For this reason, most audio systems employ a priority system. In a priority system, sounds are played in the order in which they are received until the maximum number of channels are used. Once all the channels are in use, lower-priority sounds are swapped out for higher-priority sounds. Just be sure to remember that in Unity a lower-priority number means a higher actual priority!

Importing Audio Clips

Audio sources don't do anything unless they have audio to play. In Unity, importing audio is as easy as importing anything else. You just need to click and drag the files you want into the Project view to add them to your assets. The audio files used in this hour have been graciously provided by Jeremy Handel (http://handelabra.com).

▼ TRY IT YOURSELF

Testing Audio

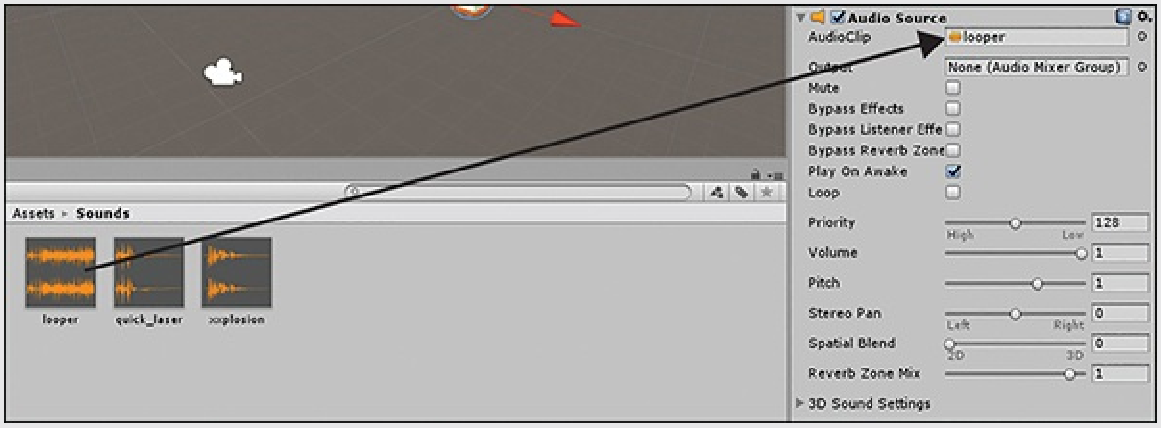
In this exercise, you'll test your audio in Unity and make sure everything works. Be sure to save this scene because you will use it again later in this hour. Follow these steps:

1. Create a new project or scene. Locate the **Sounds** folder in the book assets for Hour 21 and drag it into the Project view in Unity to import it.
2. Create a cube in your scene and position it at (0, 0, 0).

3. **3.** Add an audio source to the cube (by selecting **Add Component > Audio > Audio Source**).

4. **4.** Locate the file **looper.ogg** in the newly imported Sounds folder and drag it into the Audio Clip property of the audio source on the cube (see Figure 21.2).

5. **5.** Ensure that the **Play On Awake** property is checked, and run your scene. Notice the sound playing. The audio should stop after about 20 seconds (unless you set it to loop).



AudioClip	looper
Output	None (Audio Mixer Group)
Mute	☐
Bypass Effects	☐
Bypass Listener Effe	☐
Bypass Reverb Zone	☐
Play On Awake	✓
Loop	☐
Priority (High–Low)	128
Volume	1
Pitch	1
Stereo Pan (Left–Right)	0
Spatial Blend (2D–3D)	0
Reverb Zone Mix	1
▸ 3D Sound Settings	

FIGURE 21.2
Adding a clip to a source.

NOTE

Mute Audio Button

At the top of the Game window is a button called Mute Audio (between Maximize on Play and Stats). If you don't hear anything when your game is playing, ensure that this button is not pressed in.

Testing Audio in the Scene View

It would get a bit taxing if you needed to run a scene every time you wanted to test your audio. Not only would you need to start up the scene, you would also need to navigate to the sound in the world. That is not always easy—or even possible. Instead, you can test your audio in the Scene view.

To test audio in the Scene view, you need to turn on scene audio. Do this by clicking the scene audio toggle (see Figure 21.3). When it is selected, an imaginary audio listener is used. This listener is positioned on your frame of reference in the Scene view (not on the position of the actual audio listener component).

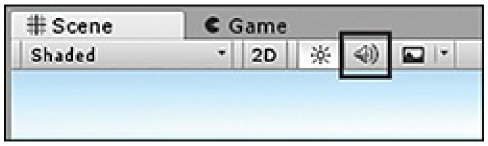


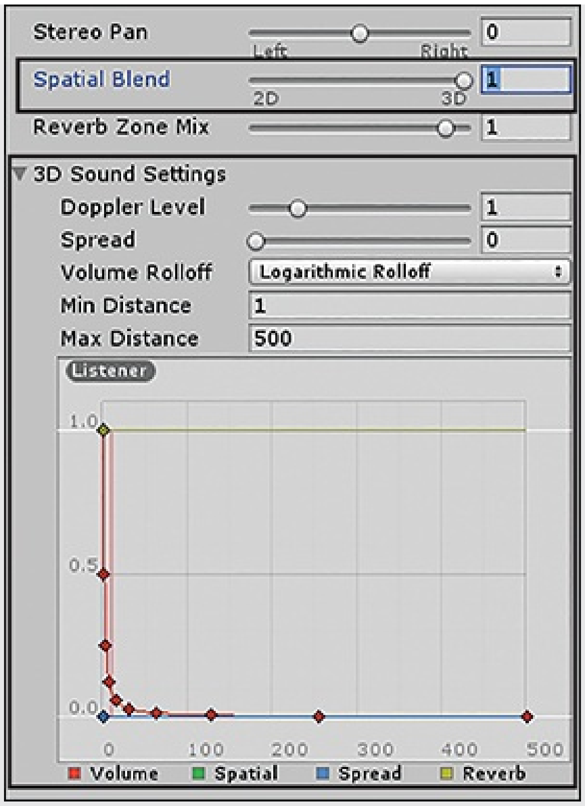
FIGURE 21.3
The audio toggle.

▼ TRY IT YOURSELF

Adding Audio in the Scene View

This exercise shows you how to test your audio in the Scene view. It uses the scene created in the Try It Yourself “Testing Audio.” If you have not completed that scene yet, do so before continuing. Follow these steps:

1. Open the scene you created in the Try It Yourself “Testing Audio.”
2. Turn on the scene audio toggle (refer to Figure 21.3).
3. Move around the Scene view. Notice that the sound stays the same volume, regardless of your distance from the cube emitting the sound. By default, all sound sources default to 2D.
4. Drag the Spatial Blend slider over to 3D (see Figure 21.4). Now try moving around in Scene view again. Note that the sound now gets quieter as you get further away.



Stereo Pan 0
Spatial Blend 1
Reverb Zone Mix 1
3D Sound Settings
Doppler Level 1
Spread 0
Volume Rolloff Logarithmic Rolloff
Min Distance 1
Max Distance 500

FIGURE 21.4
The 3D audio settings.

3D Audio

As mentioned earlier in this hour, all audio is set to be fully 2D by default. It is easy to change it to fully 3D by setting the Spatial Blend slider to **1**. This means all audio will be subject to the 3D audio effects that are distance and movement based. These effects are modified by the 3D properties of the audio component (refer to Figure 21.4).

Table 21.2 describes the various 3D audio properties.

TIP

Using the Graph

As you play with the options under 3D Sound Settings, look at the graph. It will tell you how loud the audio will be at various distances and is a great way of visualizing the effect of the controls. In this case, a picture really does say a thousand words!

TABLE 21.2 3D Audio Properties

Property	Description
Doppler Level	Determines how much Doppler effect (how sound is distorted while you are traveling toward or away from it) is applied to the audio. A setting of 0 means no effect will be applied.
Spread	Specifies how spread out the various speakers of a system are. A setting of 0 means that all speakers are at the same position and that the signal is essentially a mono signal. Leave this alone unless you understand more about audio systems.
Volume Rolloff	Determines how the change in sound volume over distance is applied. Logarithmic is set by default. You can also choose Linear or set your own curve with a custom rolloff.
Min Distance	Specifies the distance you can be away from the source and still receive 100% volume. The higher the number, the farther the distance.
Max Distance	Specifies the farthest you can be from the source and still hear some volume.

2D Audio

Sometimes you want some audio to play regardless of its position in the scene. The most common example of this is background music. To switch an audio source from 3D to 2D, drag the Spatial Blend slider over to 2D (this is the default; refer to Figure 21.4). Notice that you can also have a blend between 2D and 3D, meaning the audio will always be heard at least some amount, regardless of your distance from it.

The settings above the 3D Sound Settings section, such as Priority, Volume, Pitch, and so on, apply to both the 2D and the 3D portions of the sound. The settings in the 3D Sound Settings section obviously apply only to the 3D portion.

Audio Scripting

Playing audio from an audio source when it is created is nice, assuming that's the functionality that you want. If you want to wait and play a sound at a certain time, or if you want to play different sounds from the same source, however, you need to use scripting. Luckily, it isn't too difficult to manage your audio through code. Most of it works just like any audio player you're used to: Just pick a song and click **Play**. All audio scripting is done using variables and methods that are a part of the class `Audio`.

Starting and Stopping Audio

When dealing with audio in script, the first thing you need to do is get a reference to the Audio Source component. Use the following code to do this:

Click here to view code image

```
AudioSource audioSource;

void Start ()
{
    // Find the audio source component on the cube
    audioSource = GetComponent<AudioSource> ();
}
```

Now that you have a reference stored in the variable `audioSource`, you can start calling methods on it. The most basic functionality you could want is simply starting and stopping an audio clip. These actions are controlled by two methods simply named `Start()` and `Stop()`. Using these methods looks like this:

Click here to view code image

```
audioSource.Start(); // Starts a clip
audioSource.Stop();  // Stops a clip
```

This code will play the clip specified by the Audio Clip property of the Audio Source component. You also have the ability to start a clip after a delay. To do that, you use the method `PlayDelayed()`, which takes in a single parameter that is the time in seconds to wait before playing the clip and looks like this:

Click here to view code image

```
audioSource.PlayDelayed(<some time in seconds>);
```

You can tell whether a clip is currently playing by checking the `isPlaying` variable, which is part of the `audioSource` object. To access this variable, and thus see if the clip is playing, you could type the following:

Click here to view code image

```
if(audioSource.isPlaying)
if(audioSource.isPlaying)
{
   // The track is playing
}
```

As the name implies, this variable is true if the audio is currently playing and false if it is not.

▼ TRY IT YOURSELF

Starting and Stopping Audio

Follow these steps to see how to use scripts to start and stop an audio clip:

1. Open the scene you created in the Try It Yourself “Adding Audio in the Scene View.”
2. On the Cube game object created previously, locate the Audio Source component. Uncheck the **Play On Wake** property and check the **Loop** property.
3. Create a new folder named Scripts and create a new script in it called AudioScript. Attach the script to the cube. Change the entire script code to the following:

Click here to view code image

```
using UnityEngine;

public class AudioScript : MonoBehaviour
{
    AudioSource audioSource;

    void Start()
    {
        audioSource = GetComponent<AudioSource>();
    }

    void Update()
    {
        if (Input.GetButtonDown("Jump"))
        {
            if (audioSource.isPlaying == true)
                audioSource.Stop();
            else
                audioSource.Play();
        }
    }
}
```

4. Play the scene. You can start and stop the audio by pressing the spacebar. Notice that the audio clip starts over every time you play the audio.

TIP

Unmentioned Properties

All the properties of the audio source that are listed in the Inspector are also available via scripting. For instance, the Loop property is accessed in code with the `audioSource.loop` variable. As mentioned earlier in this hour, all these variables are used in conjunction with the audio object. See how many you can find!

Changing Audio Clips

You can easily control which audio clips to play via scripts. The key is to change the Audio Clip property in the code before using the `Play()` method to play the clip. Always be sure to stop the current audio clip before switching to a new one; otherwise, the clip won't switch.

To change the audio clip of an audio source, assign a variable of type `AudioClip` to the `clip` variable of the `audioSource` object. For example, if you had an audio clip called `newClip`, you could assign it to an audio source and play it by using the following code:

Click here to view code image

```
audioSource.clip = newClip;
audioSoure.Play();
```

You can easily create a collection of audio clips and switch them out in this manner. You will do this in the exercise at the end of this hour.

Audio Mixers

So far you've seen how to play audio from a source and have it heard by a listener. That process has been very straightforward and easy to use, but you are using it "in a vacuum." That is, you are playing only a single audio clip—or a small number of them—at a time. Difficulties arise when you need to start balancing audio volumes and effects with each other. It could be quite a pain to constantly need to find and modify each audio source in a scene or prefab. This is where audio mixers come in. *Audio mixers* are assets that act as mixing boards, allowing a fine level of control when balancing audio.

Creating Audio Mixers

Audio mixer assets are very easy to create and use. To make one, simply right-click in the Project view and select **Create > Audio Mixer**. Once an audio mixer asset is created, you can double-click it to open the Audio Mixer view (see Figure 21.5). Alternatively, you can select **Window > Audio Mixer** to open the Audio Mixer view.

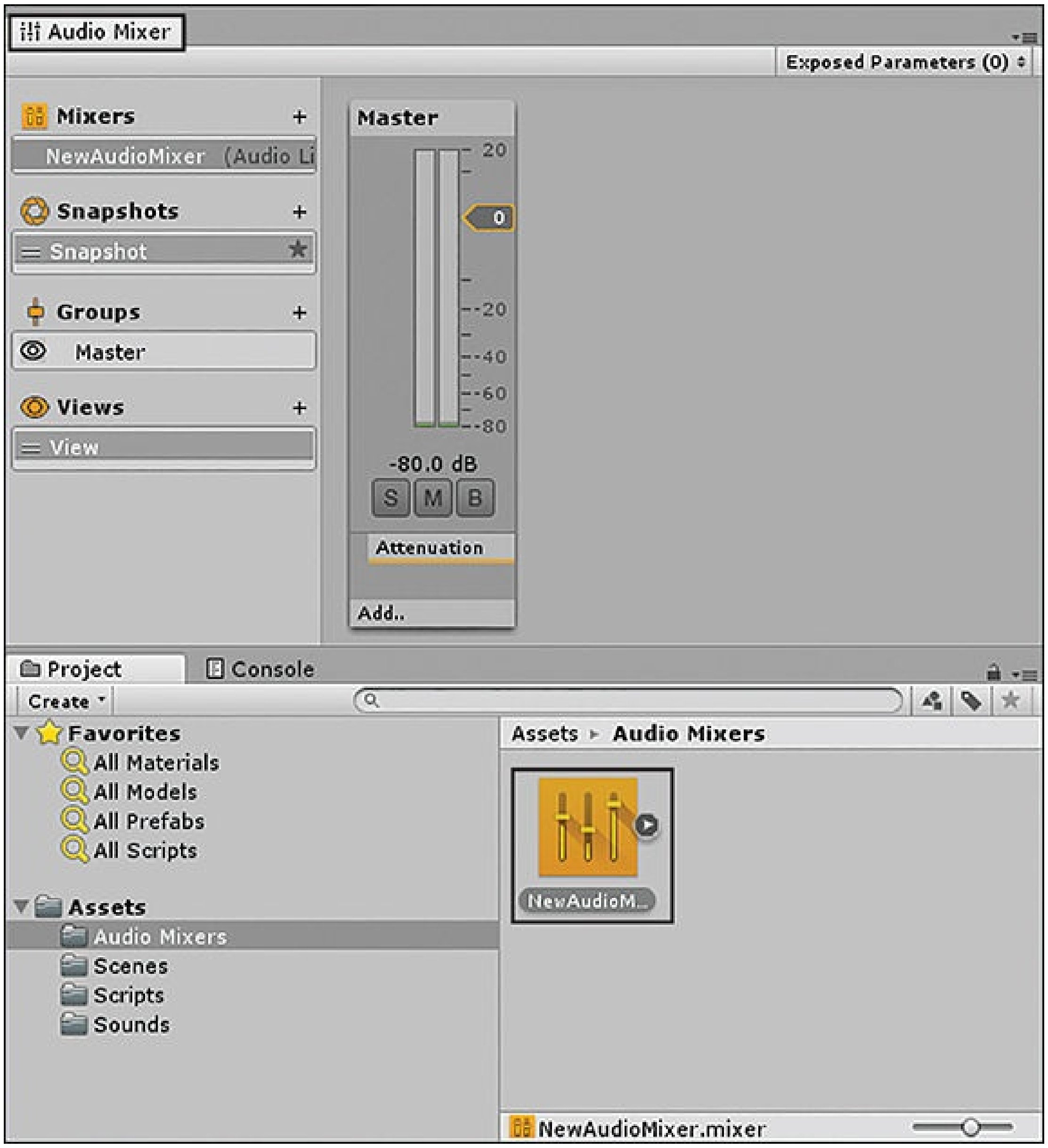


FIGURE 21.5
The Audio Mixer view.

Sending Audio to Mixers

Once you have an audio mixer, you need to route sounds through it. Routing audio through a mixer requires setting the output of an audio source to one of the audio mixer's groups. By default, an audio mixer has only a single group, called Master. You can add more groups to make organizing your audio easier (see Figure 21.6). Groups can even represent other audio mixers for a very modular approach to audio management.

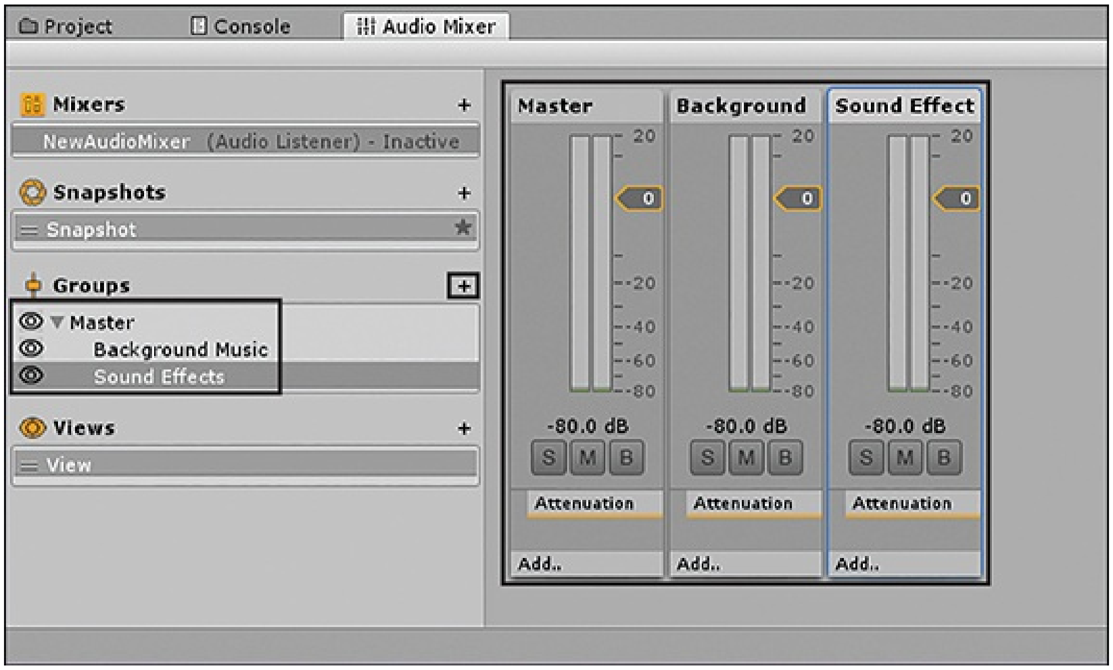


FIGURE 21.6
Adding groups.

Once you've created the desired audio groups, simply set the output property of an Audio Source component to your group (see Figure 21.7). Doing so allows the audio mixer to control the volume and effects of the audio clip. Furthermore, it overwrites the audio source's Spatial Blend property and treats the audio like a 2D audio source. By using the audio mixer, you can control the volume and effects for an entire collection of audio sources at once (see Figure 21.7).

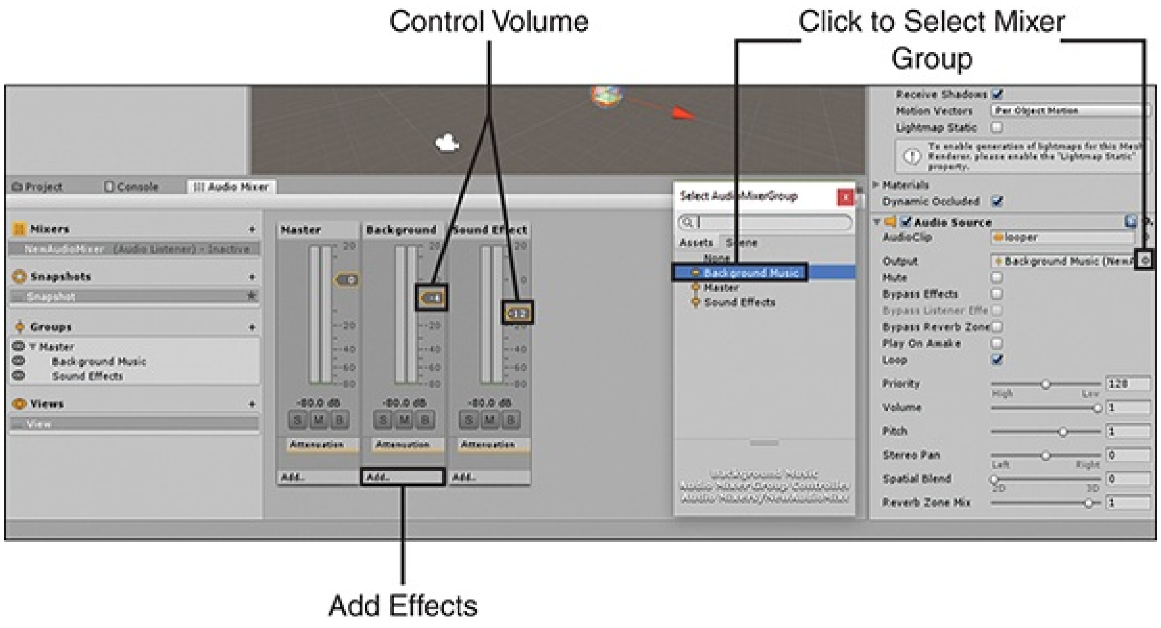


FIGURE 21.7
Routing audio.

Summary

In this hour, you’ve learned about using audio in Unity. You started by learning about the basics of audio and the components required to make it work. From there, you explored the Audio Source component. You learned how to test audio in the Scene view and how to use 2D and 3D audio clips. You finished the hour by learning to manipulate audio through scripts and exploring the use of audio mixers.

Q&A

Q. How many audio channels does a system have, on average?

A. It truly varies for every system. Most modern gaming platforms can simultaneously play dozens or hundreds of audio clips at the same time. The key is to know your target platform and use the priority system well.

Workshop

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

Quiz

1. What items are needed for working audio?
2. True or False: 3D sounds play at the same volume, regardless of the listener’s distance from the source?
3. What method allows you to play an audio clip after a delay?

Answers

1. Audio listener, audio source, and audio clip
2. False. 2D sounds play at the same volume, regardless of the listener’s distance from the source.
3. `PlayDelayed()`

Exercise

In this exercise, you’ll create a basic sound board. This sound board will allow you to play one of three sounds. You will also have the ability to start and stop the sounds and to turn looping on and off.

1. Create a new project or scene. Add a cube to the scene at position (0, 0, -10) and add an audio source to the cube. Be sure to uncheck the **Play On Awake** property. Locate the **Sounds** folder in the book assets for Hour 21 and drag it into the Assets folder.
2. Create a new folder called **Scripts** and create a new script named **AudioScript** in it. Attach the script to the cube. Replace the contents of the script with the following:

Click here to view code image

```
using UnityEngine;

public class AudioScript : MonoBehaviour
{
    public AudioClip clip1;
    public AudioClip clip2;
    public AudioClip clip3;

    AudioSource audioSource;

    void Start()
    {
        audioSource = GetComponent<AudioSource>();
        audioSource.clip = clip1;
    }

    void Update()
    {
        if (Input.GetButtonDown("Jump"))
        {
            if (audioSource.isPlaying == true)
                audioSource.Stop();
            else
            audioSource.Play();
        }

        if (Input.GetKeyDown(KeyCode.L))
        {
            audioSource.loop = !audioSource.loop; // toggles lopping
        }

        if (Input.GetKeyDown(KeyCode.Alpha1))
        if (Input.GetKeyDown(KeyCode.Alpha1))
        {
            audioSource.Stop();
            audioSource.clip = clip1; audioSource.Play();
        }
        else if (Input.GetKeyDown(KeyCode.Alpha2))
        {
            audioSource.Stop();
            audioSource.clip = clip2;
            audioSource.Play();
        }
        else if (Input.GetKeyDown(KeyCode.Alpha3))
        {
            audioSource.Stop();
            audioSource.clip = clip3;
            audioSource.Play();
        }
    }
}
```

3. In the Unity editor, select the cube in your scene. Drag the audio files **looper.ogg**, **quick_laser.ogg**, and **xxplosion.ogg** from the Sounds folder onto the Clip1, Clip2, and Clip3 properties of the audio script.

4. Run your scene. Notice how you can change your audio clips with the keys **1**, **2**, and **3**. You can also start and stop the audio with the **spacebar**. Finally, you can toggle looping with the **L** key.