and the sample resolution are shown on the left side of Figure 3-4; just as I said, the default Audacity sample settings are 32-bit resolution with a CD-quality 44.1 kHz sample frequency.

Now that you have the sample data that you'll be refining over the next few chapters, let's save your project, which you always want to do after creating any sample. This is done using the Audacity **File ➤ Save Project** menu sequence, as shown in Figure 3-5.

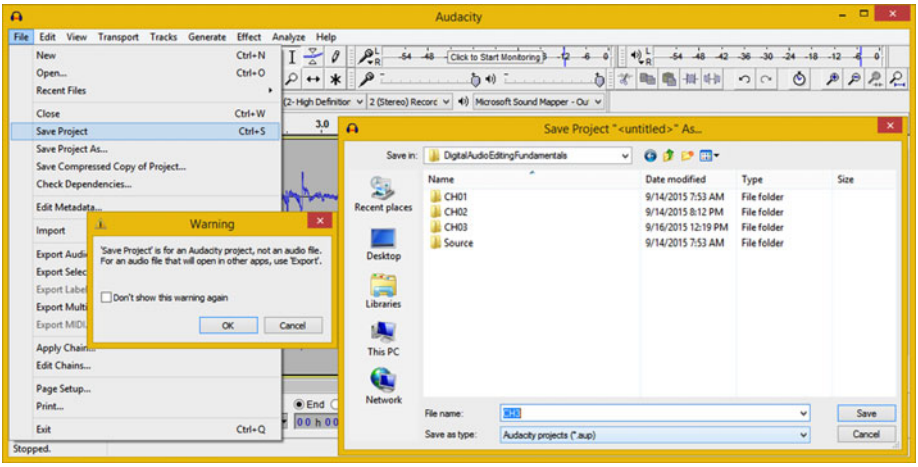

Figure 3-5. *Save you Audacity project using File ➤ Save Project*

The first thing that you see is a **Warning** dialog that informs you that an AUP file format is not an audio (playback) file format, but instead an **AU**dacity **P**roject data file format. This is shown on the left in Figure 3-5. Click the **OK** button, enter a **File name** for your Audacity project, and then click the **Save** button. I saved this sample as **CH3.AUP** in case you want to use it during Chapters 5–8.

Summary

In this chapter, you learned about digital audio data sampling concepts, techniques, and principles, as well as how to calculate sample data and how to record your own data samples using Audacity 2.1.1. You also learned about samplers and sample resolution along the y axis and sampling frequency along the x axis.

In the next chapter, you learn about **digital audio data formats** and **digital audio data transmission** principles.

CHAPTER 4

The Transmission of Digital Audio: Data Formats

Now that you understand the fundamental concepts, terms, and principles behind analog audio and how it is digitized into digital audio, it is time to explore how digital audio is compressed and stored using popular open source digital audio file formats.

You'll learn about advanced digital audio concepts, such as compression, codecs, bit rates, streaming audio, captive digital audio, and HD audio. Finally, you'll look at a number of powerful digital audio formats that are supported by open source content development platforms, such as HTML5, Java, JavaFX, and Android Studio. You can use any of the digital audio formats to deliver digital audio content for podcasts, music publishing, web site design, audio broadcasts, or multimedia applications.

Audio Compression and Data Formats

Once you sample your audio, you compress it into a digital audio file format for streaming over the Web or for captive audio file playback within an application. In this chapter, you'll look at encoding audio using bit rates, and learn about streaming and the new 24-bit HD audio standard, which is now utilized in broadcast and satellite radio. I'll also cover audio codecs and the audio file formats that they support across open platforms, such as HTML5, Java, and Android. (I cover digital audio data footprint optimization in Chapter 12, after you learn more about Audacity and digital audio editing.)

I want to make sure that you have a deep understanding of these digital audio new media assets so that you can eventually "render" them inside of your target application and attain a professional product that offers an impeccable end-user experience.

Digital Audio Codecs: Bit Rates, Streaming, and HD

Digital audio assets are compressed using something called a **codec**, which stands for "**co**de **dec**ode." The codec is an **algorithm** that applies **data compression** to digital audio samples and determines which playback rate, called a **bit rate**, it will use, as well as if it will support **streaming** or playback during network data transfer. First, let's take a look at how you use digital audio assets in your applications: Do you store audio inside an

W. Jackson, *Digital Audio Editing Fundamentals*, DOI 10.1007/978-1-4842-1648-4_4

application or do you stream it from remote servers over the Internet? After that, you'll consider the audio playback rate or data-streaming bit rate that you'll want to use. Finally, you'll learn about HD audio and see if it's appropriate for your digital audio applications. Only then will you be ready to look at the different audio file formats, which are actually codecs!

Digital Audio Transmission: Streaming Audio or Captive Audio?

Just as with digital video, which you view on the Internet every day, digital audio assets can either be **captive**, or contained within an application (for example, in an Android APK file), or they can be **streamed** using a remote data server. Similar to digital video, the upside to streaming digital audio data is that it can reduce the data footprint (size) of your application's files. The downside is reliability.

Streaming audio saves the data footprint, because you don't have to include all that data-heavy new media digital audio in your app file. Thus, if you are planning on coding a Jukebox application, you want to consider streaming digital audio data, as you would not want to pack your song library into your app's file because it would be 10 gigabytes (in a large library).

Otherwise, for application audio, such as user interface feedback sounds, game play audio, and so forth, try to optimize your digital audio data so that you can include it inside your app file as a captive asset. In this way, it is available to your application users when needed.

As you know, I'll go over optimization in Chapter 12, after digital audio editing has been covered. The reason that I want to cover this topic toward the end of the book is that the last step in the asset creation process is exporting your digital audio data using one of the formats discussed in the next section.

The downside to streaming digital audio is that if your user's connection (or your audio server) goes down, your audio file won't be present for your end users to play and listen to! The reliability and availability of a digital audio data stream is a key factor to consider on the other side of the streaming audio vs. captive digital audio decision.

Streaming Digital Audio Data: Setting Your Bit Rates Optimally

One of the primary concepts in streaming your digital audio is the bit rate of that digital audio data. Again, this is very similar to digital video, which also uses the concept of bit rates to determine the size of the data pipe that the audio data streams through. The digital audio bit rate is defined during digital audio file compression by the settings that you give to the codec.

Digital audio files that need to support a lower bit rate to accommodate slower bandwidth networks have more compression applied to the digital audio data. This results in a lower audio-quality level. However, lower playback quality isn't as noticeable in digital audio as it is in digital video.

Low bit-rate digital audio can always play back smoothly across a greater number of hardware devices. This is because if there are fewer bytes of audio data to transfer over any given data network, then there are fewer digital audio data bytes to be processed by the CPU inside that hardware device.