Waveform Reverberation: The Reverb Effect

Another popular digital audio effect that is often applied with algorithmic processing is **reverb**. Reverb simulates being inside an enclosure, whether a small room or a large amphitheater. For this reason, a primary setting in the Reverb dialog (see Figure 8-8) is the **Room Size**, which defaults to 75%. There's also a millisecond pre-delay data value that you can provide, as well as **Reverberance**, **Damping**, and **Stereo Width** percentage values. To fine-tune the effect, you can reduce the percentage of low tone and high tone frequencies that get through the filter, and set custom **Wet Gain** or **Dry Gain** values in decibels.

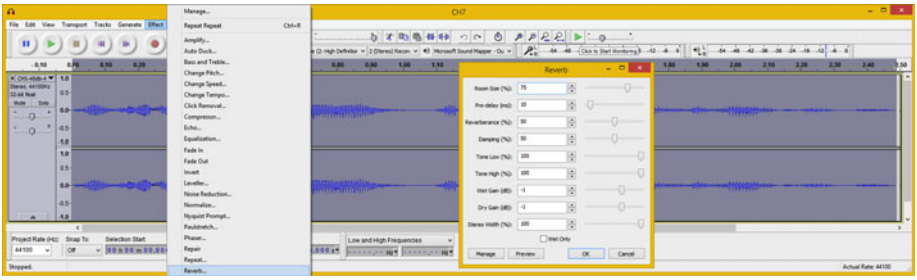

Figure 8-8. *Select sample, and select Effect ➤ Reverb algorithm*

I decided to try the default settings, which provide a standard reverb processing result, to see what the algorithmic processing is going to do to a sample waveform.

As you can see in Figure 8-9, the sample waveform is far more complex than it was before. This means that an effect like reverb is going to influence your data footprint, because there is more data to compress. The areas that were silent (thin lines) have room echo data in them, as you can see over the duration of the entire data sample.

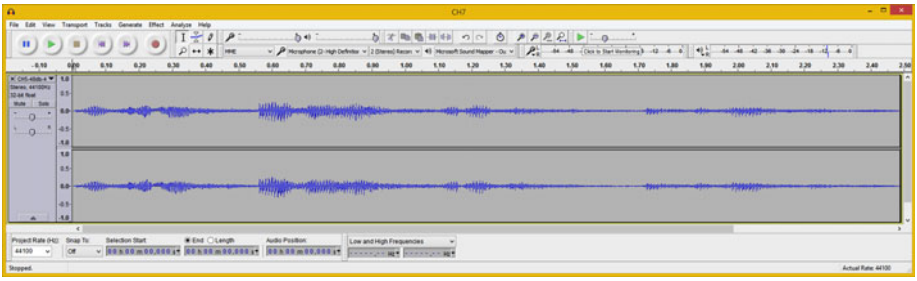

Figure 8-9. *Frequency x axis and length of sample is compressed*

Therefore, if your application doesn't require reverb as a special effect, or more importantly, if the platform that you will deliver on can add that effect to the waveform (like Java and Android), then you do not want to "hard-code" the data inside of your basic sample waveform. This approach allows you to have more flexibility in manipulating audio inside of an application. The core effects that you're looking at in this chapter are available in many open source new-media platform API packages, such as those in Java, JavaFX, and Android Studio.

Next, let's look at another effect that provides the echo processing framework to audio waveforms, the **Delay** effect.

Waveform Echo Chamber: The Delay Effect

Whereas the Reverb effect is a subtle type of echo effect, you can create more pronounced echo effects that are more like the long distance echo effects you get in canyons, for instance, by using the Delay effect. There are several types of delay, such as a regular bouncing ball or a reverse bouncing ball, in the Delay drop-down menu (see Figure 8-10). You can set the delay level and the delay time, and even set a pitch change effect from Pitch/Tempo to Pitch Shift. You can fine-tune this effect by specifying the pitch change for each echo using semitones (as you did in the Change Pitch effect) and the number of echoes to produce.

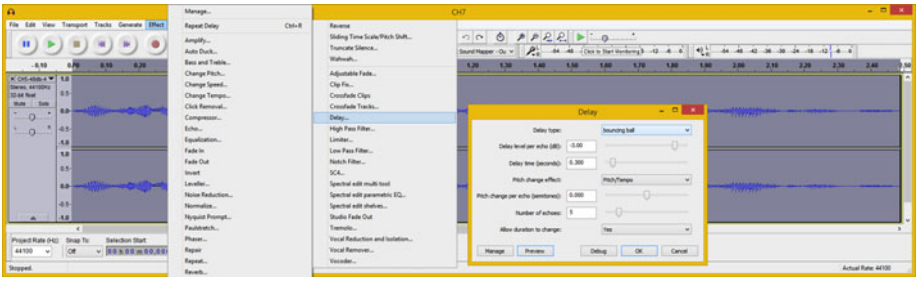

Figure 8-10. *Select sample, and select Effect ➤ Delay algorithm*

As you can see, I selected the bouncing ball delay type, because I like the delay signature that I hear when I bounce a ball. As you might imagine, this delay effect extends your sample duration because it is adding significant echo data (see Figure 8-11). The black vertical line on the right is the end of what was the original sample length. The more delay time and the higher the number of echoes that you specify, the longer the new data sample (time) length becomes.

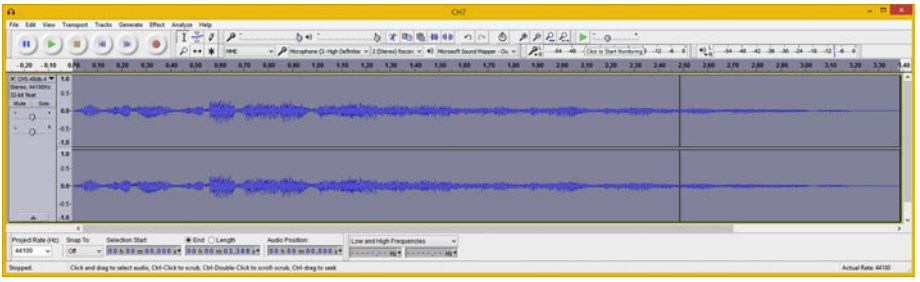

Figure 8-11. *The Delay effect may extend the data sample length*

The Delay effect can add more to the data footprint of a sample than the Reverb effect, so make sure that you need to add echo into your sample; otherwise, find a way to process it "externally" by using Java code, or even OS or hardware capabilities that may provide EQ, reverb, and echo.

Next, let's take a look at how to "shave" frequencies off of your data sample. This is usually done with the Low Pass and the High Pass Filters. You'll use the Low Pass Filter to remove the last remnants of those chirping artifacts you've worked with in the past few chapters. If that doesn't work, there's also a Notch Filter that allows you to target (notch) individual frequencies that you want to filter out or remove from the sample frequency spectrum.

Waveform Shaving: The Low Pass Filter Effect

The High Pass Filter and the Low Pass Filter remove unnecessary frequency data, or frequency data that's undesirable or causing problems with your intended aural result. It can be a good data footprint optimization tool, because even a frequency that cannot be heard needs to be encoded as data, which means that these filters can be used to remove sample data that is not needed (or heard), and thus reduce the data footprint.

As you can see in Figure 8-12, this dialog allows you to specify the **Cutoff frequency**, where everything lower (Low Pass) or higher (High Pass) is "shaved" or removed from the frequency spectrum by the filter's algorithm. There's also the **Rolloff** setting, which specifies the number of decibels per octave that the filter is going to affect. This specifies a magnitude for the application of your cut-off frequency value.

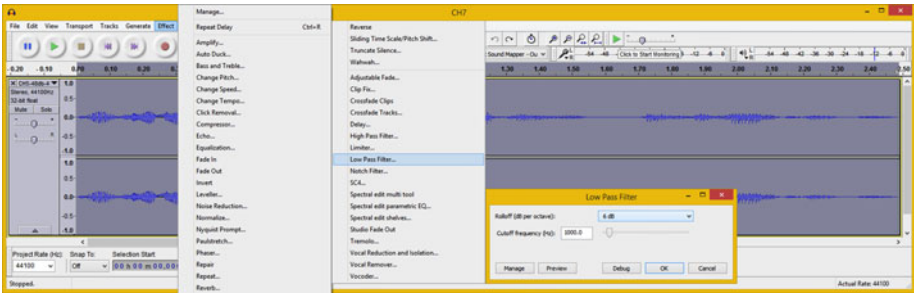

Figure 8-12. *Select sample, and select Effect ➤ Low Pass Filter*

My process for trying to use the Low Pass Filter to remove the last remnants of the chirp artifact was to first use Audacity's default Low Pass Filter settings of 6 dB and 1000 Hz.

This affected the vocal sample and made it duller, so I increased the default cut-off frequency to brighten the vocal sample back up to where it was, while lessening the final remnants of the chirp artifact.

This value turned out to be a value of 2000 Hz, because higher values permitted the remaining chirp artifact to make it through this tonal filter.

After I ascertained the optimal cut-off frequency setting to use, I selected a 12 dB Rolloff option. This provided chirp removal, but it did not affect the vocal data sample quality much.

Finally, I tried a setting of 24 dB, which affected the quality of the vocal part of my data sample too much; so, the final settings for the Low Pass Filter effect were 12 dB and 2000 Hz.

Next, let's take a look at the more surgical Notch Filter, which allows you to specifically target any frequency.

Tunneling Into Your Waveform: The Notch Filter

Instead of cutting off the top or bottom sample waveform frequencies, the Notch Filter effect allows you to surgically "tunnel" into any section of your sample to remove a specific frequency or a range surrounding that frequency. As you can see in Figure 8-13, the **Notch Filter** dialog allows you to specify your target **Frequency** and **Q value**, which is the tunnel width around that frequency. Lower Q values make this tunnel smaller, and so a very small Q value essentially targets the frequency with more surgical precision. The default settings are 60 Hz and a 1.0 Q factor, which I need to change to find the frequencies of the remaining chirp artifacts.

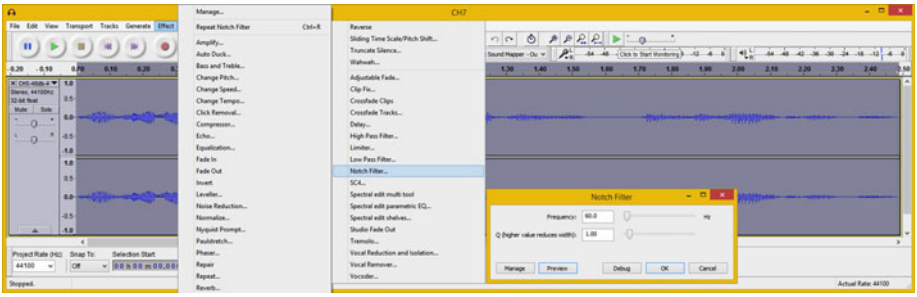

Figure 8-13. *Select sample, and select Effect ➤ Notch Filter*

My process for finding the frequency was to use the Preview button, first at 60 Hz and then at the maximum of 10000 Hz, where the artifact was still audible. Then I used a halfway mark of 5000 Hz, and then backed off that in increments of 1000 until I hit two frequencies that the artifacts were not audible at. These were 3000 Hz and 4000 Hz. I decided to use 4000 Hz because the result sounded better. I left the 100% tunnel size, as you can see in Figure 8-14, and clicked the **OK** button to apply.

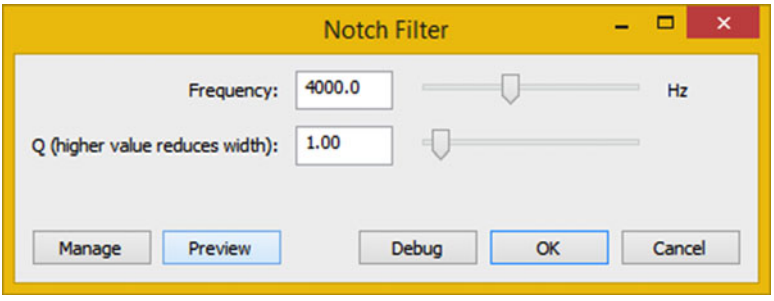

Figure 8-14. *Setting the Notch Filter to a Frequency of 4000 Hz*

To see the amount of data that the Notch Filter removed, I targeted the chirp artifact. Then I saved the data sample using the FLAC file format. As you can see in Figure 8-15, the file size is now 104 KB, whereas before it was 112 KB. So, the Notch Filter removed 8 KB worth of artifact data from the sample. The file size is now more than 300% less (104 KB × 3 = 312 KB). The original unedited baseline FLAC file was 316 KB.

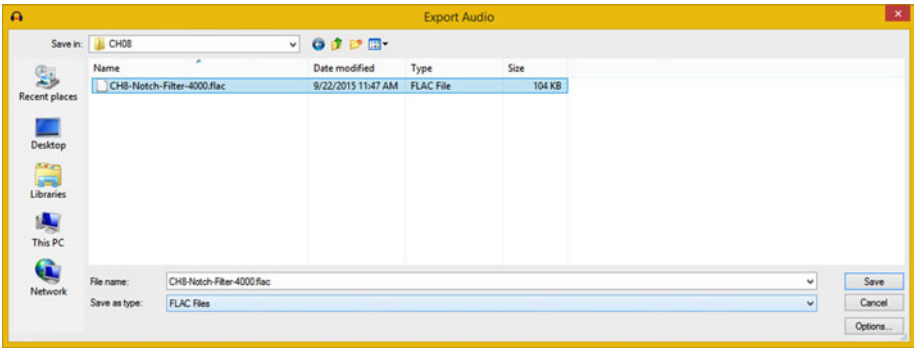

Figure 8-15. *This sample now has over 300% less data footprint*

If the FLAC format is getting a 104 KB data footprint for this audio data sample, other data formats will provide an even smaller data footprint, as you will see in chapter covering data footprint optimization.

Summary

In this chapter, you looked at digital audio processing using algorithms found in the Audacity Effect menu, where internal effects and plug-in filters can be accessed. You looked at many of the mainstream digital audio editing effects that are applied when editing and sweetening digital audio samples. These effects include Amplify, Pitch Shifting, Sample Speed, Equalization, Reverb, Delay, Low Pass Filter, and Notch Filter. You also looked at the **Effect ➤ Manage** menu sequence, which accesses a dialog that allows you to see the other 80 plug-in filters that are not on the Effect menu. Be sure to familiarize yourself with all 128 plug-in filters on the Audacity Effect menu if you want to become a digital audio editing professional.

In the next chapter, you learn about **digital audio data visualization**, by using **spectral analysis** concepts, terms, and principles, along with the Audacity 2.1.1 Analyze menu.