# 5

# Getting Them Started: I Didn’t Know You Could Do That With a Computer

## Just What *Is* a Computer, Anyway?

“What’s an iPod?” “What’s an iPhone?” “What’s an Android?” Pose those questions to any gen-Xer or millennial and we guarantee you that there’s one answer you *won’t* get: “a small, hand-held computer.” We’d also be shocked if their answers included anything like “a communication device capable of connecting to a wireless or cellular network.” You’re more likely to hear: “It’s like, a thing I use to chat with my friends, watch videos, and listen to music.” They can tell you what these devices are *for*, but they would have trouble telling you what they really *are*.

Is this bad? No, not in and of itself. The computer has truly become an appliance, and some think about it no more deeply than they think about their toaster. Let’s credit the geeks in Silicon Valley for making complex devices so easy to use that, as they say about driving a car, “any fool can do it, and many do.” If your neighbor has a problem connecting to the Internet and comes to you for help and you ask what browser he or she uses, you just might get a blank stare. If you then try to break the ice by asking: “When you connect to the Internet (or World Wide Web) to look up something with Google or read your email, what program do you use?” An answer we commonly hear is: “I don’t know. I just click on the little picture that says ‘Connect to the Internet’ (or ‘Read Your Email,’ etc.).” This assumes, of course, that your neighbor knows what you mean by “program.” If not, it’s probably easiest to just ask: “Tell me the steps you follow to open your email so that you can read and send new messages.”

Even though we feel that everyone should know a bit more about computers than this, we applaud the developments that have made computers everyday devices. All of us drive cars, but the vast majority of us have no desire whatsoever to really understand what’s going on under the hood or the physics involved in controlling a 2-ton vehicle traveling at 65 miles per hour. However, understanding a bit about those things can make us better drivers, and understanding a bit about how computers work can make students better users. To begin with, recognizing that one’s iPhone is a hand-held computer connected to a complex network can alleviate a lot of frustration when something goes wrong. And learning how to back up one’s personal files can alleviate a lot more than that when a disk goes bad.

But how do we interest gen-Xers and millennials in learning how computers work? When Jesse asked his 13-year-old niece — who uses both handheld and desktop computers constantly for communication, gaming, and doing her homework — whether she had any interest in learning to program those machines to make them do new and interesting things, she not only replied “No,” but she added that she didn’t know any of her friends who had such an interest, either. That doesn’t bode well for our efforts to help students learn to do computational thinking.

Many children start taking music lessons very early in their lives. It’s not uncommon for preschoolers to already be enrolled in music classes, and YouTube hosts an uncountable number of videos of children singing, dancing, conducting, and doing other musical things even if they’ve never had any training at all. They’re “jes doin’ a-watcha comes natcherly” [[4](#_ENREF_4)]. Ideally, we want young musicians engaging first with actual musical materials through sound and kinesthetic explorations before introducing them to theory or the structure of notation embodied in staves, notes, and accidentals. Likewise, we want new computing students — particularly those turned off by mathematical formalities — engaging first with interesting applications before introducing them to theory and the structure of computer languages.

Our approach is therefore to start them off by getting them to use a computer to manipulate something they’re familiar with, surreptitiously showing them that they can *control* a computer and get it to do what *they* want it to. We do this by getting them to create digital mash-ups and variations of their favorite songs.

## Audacity: The Standard in Free Music Editing

We love Audacity [[2](#_ENREF_2)]. It’s free, it’s easy to use, it has more than adequate functionality for our purposes, and it runs on both Macintosh and Windows systems. Admittedly, it’s not a professional quality editor like Avid’s Pro Tools [[3](#_ENREF_3)] or even a “prosumer”[[1]](#footnote-1) [[7](#_ENREF_7)] quality editor like Sony’s Sound Forge [[8](#_ENREF_8)] or Adobe’s Audition [[1](#_ENREF_1)]. However, free cost, ease of use, reasonable functionality, and cross-platform availability are critical characteristics that allow all students (and teachers) to download and run Audacity on their own systems, thus allowing them to do their assignments outside of class. These characteristics are paramount in virtually all decisions we make about what software to use in our courses.

Audacity’s home page is http://audacity.sourceforge.net. You can download the software for Windows, Macintosh, and Linux systems from there. Once the program is installed, you can import and manipulate lots of different file types, including MP3s, but there’s a bit of an issue with regard exporting out MP3 files. The Appendix at the end of this chapter provides instructions to address this issue that augment those on the Audacity website itself. If you don’t yet have Audacity up and running on your system with the LAME MP3 encoder, please see that appendix.

***Important Note:*** All descriptions of Audacity in this book refer to Audacity version 1.3.13 (beta), released on April 11, 2011. Almost all screen shots were made on a Windows 7 system, but in most cases, the Macintosh version looks only slightly different.

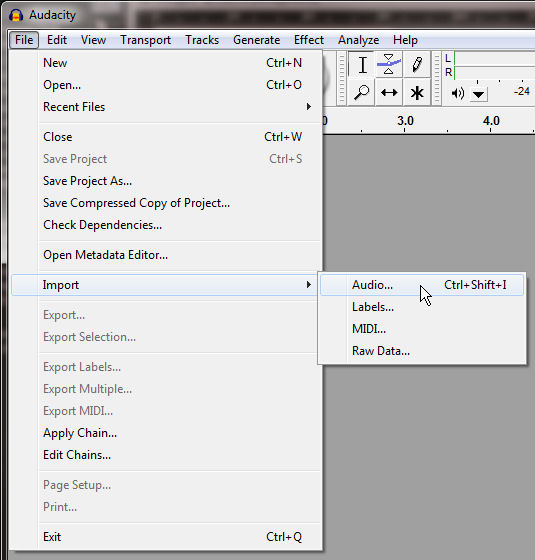
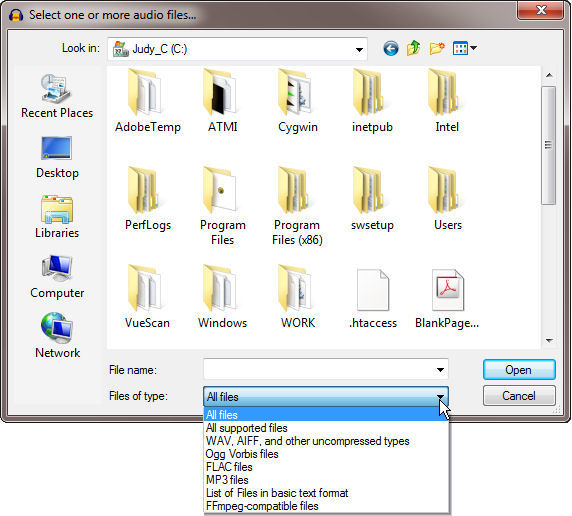
## Getting Music Into and Out of Audacity

There are many terrific things that students can do with Audacity, but we begin by building on the song flowchart work discussed in a previous chapter. Virtually all students have MP3 versions of their favorite songs, and if they do not, they can usually find an MP3 of a song they’re willing to work with or make their own MP3 using a variety of tools, including Audacity. If they find a song they want to work with on YouTube, they can download that song as an MP3 using the free YouTube to MP3 converter available from DVDVideoSoft.com.[[2]](#footnote-2)

The types of audio files that Audacity can import include (see Figure -):

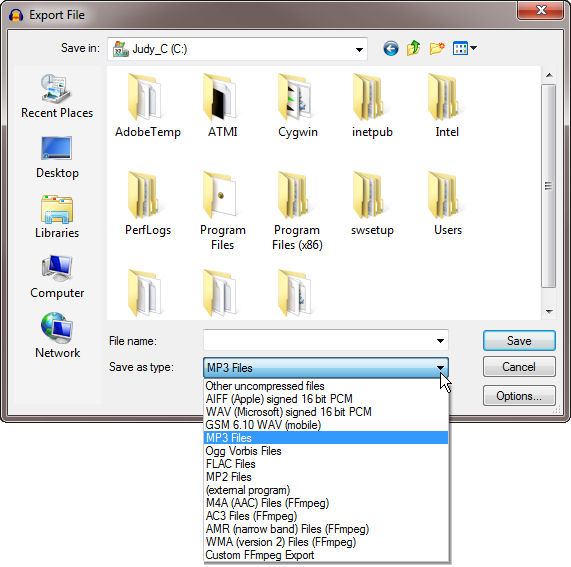
* WAV, AIFF, and other uncompressed file types, most of which are playable in the Windows Media Player
* Ogg Vorbis files, which are playable *without* a plug-in in the Firefox browser, but *not* in most other browsers
* FLAC (Free Lossless Audio Codec) files, which are “similar to MP3s, but lossless, meaning that audio is compressed in FLAC without any loss in quality” [[6](#_ENREF_6)]
* MP3 files, which are playable *without* a plug-in in the Chrome, Opera, and Safari browsers, in the Internet Explorer browser with the QuickTime plug-in installed, and in QuickTime, iTunes, and the Windows Media Player
* FFmpeg-compatible files, such as MPEG and FLV (Flash), which are playable in many applications, including games

That about covers the field, so students should be able to import music in just about any file format they happen to have.

*Figure 5-1.* Importing an audio file into Audacity and supported import file types.

*Exporting* music is only slightly different. The basic process is the same in that you select Export... from the File menu (once you have something to export and the Export... option is not grayed out indicating that it is disabled), but the list of available file formats is longer (see Figure -). The only issue here is that the MP3 files entry will not appear in the list of available export file formats unless the MP3 encoder is installed, as mentioned previously and discussed in the Appendix to this chapter.



*Figure 5-2.* Supported Audacity export file types.

## Gaining Computational Thinking Skills Through Audacity

It may at first seem strange to go into so much detail on Audacity import and export file formats in a book on computational thinking. We don’t think so. As a matter of fact, understanding how to work with Audacity at more than a rudimentary level is exactly the type of thing that we believe *should* be covered in an interdisciplinary course or project on computing + music. This does *not* mean that we get into the mathematics of compression or the algorithms that allow Audacity to shape and manipulate sounds. It means that we want students to understand things like Audacity’s file structure to the point that they can backup their work and move projects from one system to another. Just as musical instruments produce very different results depending upon the skill and experience of the artist who plays them, computer tools such as Audacity produce very different results — and levels of frustration — depending upon the skill and experience of the person who uses them. We don’t try to make our GenEd students programmers, but we certainly do try to make them skilled, experienced, and sophisticated users who practice computational thinking (CT).

So just what CT skills can students gain by working with Audacity? Before answering this question, it’s worthwhile to review Jeanette Wing’s definition of CT once again:

Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science. [[10](#_ENREF_10)]

Also, remember that in the context of our interdisciplinary work we define CT very broadly, encompassing a wide range of problem solving skills. Our reason for this stance is that we see problem solving as the essence of CT for students in non-technical fields. These students are just as smart as any other, of course, and they can use computer applications as well as anyone *when things go according to plan*. The issue is their ability to solve problems *when things go wrong*. Do they analyze the situation using CT, or do they throw up their hands and say “I can’t do this”? Our goal is obviously to get them to do the former.

## How Could You Have the *Audacity* To Do That to My Song?!?

Rather than just listing and describing the CT behaviors that can be gained by using Audacity, let’s explore them in the context of an assignment that might be used in a course like our *Sound Thinking*. The assignment itself is conceptually simple, and it builds directly on the song flowchart project discussed in previously. Here it is in a nutshell:

*Part I*. Load the song that you used to create your flowchart into Audacity. Using your song flowchart as a guide, break the song into chunks and store each as a separate MP3 file. Reload the MP3 files you created into a new Audacity project so that each is in a separate track. Recombine the tracks in various ways to make a new “song” from the component chunks. [*There are several ways to do this, which we would cover in class.*] Save your Audacity project so that you can come back to it later and save your composition as a new MP3 file.

*Part II.* Reload you composition into Audacity as a new project. Experiment with some of the sound manipulation tools provided on the Effect menu by applying them to all or part of your composition to see what they do. Remember that you can type Control-Z (Ctrl-Z) on Windows or Command-Z (‑Z) on Macintosh to “undo” any change that you don’t like. Save your Audacity project and save this second version of your composition in such a way that you now have both your original and your manipulated versions.

Do students learn CT skills by doing assignments such as these? Yes, and they have fun while they do. OK, so just what do they learn? “Let me count the ways...” [[5](#_ENREF_5)]

1. ***Students learn about file types and data storage.*** They learn that Audacity, like many other programs, has its own unique way of storing data so that users can close the program and reopen it at a later time and continue from where they left off. They learn that MP3 files are not the same as AU files.[[3]](#footnote-3) They learn about hierarchical folder structures, because Audacity’s critical AU files exist in a subfolder of the main folder. For example, Figure - shows an Audacity folder hierarchy in the first column, with the contents of each folder in this hierarchy expanded in the remaining four columns.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Folder  Hierarchy** | **Contents of  AudacityTestsEtc** | **Contents of OneToFive\_data** | **Contents of e00** | **Contents of d00** |
| DirectoryStructure.jpg | DirectoryContents-Level1.jpg | DirectoryContents-Level2.jpg | DirectoryContents-Level3.jpg | DirectoryContents-Level4.jpg |

*Figure 5-3.* Audacity folder hierarchy and typical contents of subfolders.  
*Icon Legend:* Icon-Folder.jpg = Folder (or subfolder). Icon-MP3file.jpg = MP3 file (.mp3).   
Icon-AudacityProjectFile.jpg = Audacity project file (.aup). Icon-AUfile.jpg = Audacity sound file (.au).

1. ***Students learn about the importance of saving intermediate versions of their work.***Audio editing can be complex, involving many steps. It is really disheartening to work on something for an hour or more and then to be unable to save it or to have a computer glitch wipe out one’s work. It has happened to all of us, even the most techno-savvy geeks. It can be devastating to students who are already predisposed to dislike using computers.

To avoid such crises, we teach students to do two things. First, we caution them to save the entire Audacity project every 10-15 minutes and/or at critical points in their work using File🠞Save Project As... rather than just File🠞Save Project..., adding -v01, -v02, -v03, etc. to the project name each time they save it. This allows them to go back to a good, working, previous version of their project if they — or the computer — really screws something up. Second, we entreat them to take notes as they go along so that they can recall what they did to get “from here to there.” One Music student learned these lessons very well, with a valuable side-effect. He wrote in his reflection on a related assignment:

I had stayed up till about 2:30 in the morning [working on this project]. I went to export the audio from my Audacity project and received nothing but silence. I troubleshot until about 4:00AM and then threw in the towel. Once again the computer had won. I can’t explain how much I despise giving in to an inanimate object. I had no choice but to postpone the project.

About four days later I tried a couple of last ditch efforts and then realized I would have to start the project from scratch. Luckily, having taken the advice of my knowledgeable professors, I had taken notes on the original project. It was an interesting process to repeat my steps again. Not only was I able to reproduce my project, but I was also able to create a better version. I found better ways to do things and had more creative ideas as I reworked my project.

1. ***They learn techniques for and the importance of making backups.*** They learn that if they want to transfer an Audacity project to another system, they must copy the entire folder hierarchy, not just the .aup project file in the top-level folder (a common mistake). This is an important concept, since hierarchical file structures are used by most other audio and video editing and many other types of programs, too. Backing up also involves dealing with the large sizes of the data files, which in the case of Audacity are the .au files in the last “leaf” of the file structure tree. Each of these is typically 1 megabyte in size. For large projects, especially those in which a lot of editing has taken place, there can literally be dozens of .au files that one has to copy. Therefore, if students are going to backup their projects from school systems to their personal ones, they have to plan to bring a flash or thumb drive big enough to hold all those files and make sure they have enough time to copy them before they have to run to their next classes. These may seem like trivial matters to experienced computer users, but they are anything *but* trivial to students whose only computer experience involves using a device that they don’t even realize is a computer.
2. ***They learn to explore and experiment.*** Audacity is a rich environment. It provides a wealth of manipulations that students can perform on imported audio. For starters, the number of effects that one can apply is huge, as shown in Figure ‑6.

(*Note:* These same effects are available on Macintosh systems, but the menu looks somewhat different because they are grouped into submenus as shown in Figure ‑.) It is not only fun to figure out what each of these effects does, but also very instructive from both a CT and a music point of view.

|  |
| --- |
| *Figure 5-6.* Audacity Effect  options menu (on Windows).  *Note that the effects are  not grouped.* |

To us, exploration and experimen­tation (with the ability to undo changes), as well as wading through sometimes sparse documentation, are as much CT skills as writing program code. While students learn these skills they also deepen their understanding of music structure and sound effects and get to exercise a bit of creativity.

|  |
| --- |
| Menu-Effects-Expanded-Macintosh-isolated.jpg  *Figure 5-6.* Audacity Effect options menu  (on Macintosh). *Note the difference between  the Mac and Windows versions of this menu:  on the Mac, the effects are grouped.* |

The exploratory aspect of the project can produce some wonderfully serendipitous results. A Music student wrote:

I experienced something while working on this project that I was not expecting. As musicians and consumers of music we are all familiar with the potential music has to move people. There is a phenomenon known as a “skin response,” which is the very thing I live for in music. There is simply nothing that compares.

So there I was ... just throwing in music and experimenting, when out of nowhere I felt this shutter. These four songs lined up at this one point at 2:30 in the morning. That was an amazing moment. I never would have expected to nor have I experienced this in “digital music.” ... This project was worth that single moment.

1. We could go on and on, but let’s end on this one: ***students learn to solve problems.*** This is perhaps the penultimate CT goal, and it certainly involves using (in Wing’s words [[10](#_ENREF_10)]) a “range of mental tools” that are part of “the breadth of the field of computer science.” The types of problems students encounter range from the mechanics of getting *this* chunk to go *there* to getting the mash-up to sound the way they want it to. The technical students in the class can help the arts students at the mechanical end of that spectrum, while the arts students can help the technical ones at the aesthetic end. It’s a win-win situation that exemplifies the best characteristic of interdisciplinary work: students learn *from each other*, not just from the professors. As a matter of fact, it is common for students to come up with a trick or two that the professors don’t know. The teacher-student roles are then reversed, creating one of those magical moments that make teaching so fulfilling.

## An Example from Our Course: The Audio-Ethnography Project

The first time we taught *Sound Thinking* we used a slightly different assignment than the one just described because we did not do the flowchart project (described in an earlier chapter) that year. However, the assignment we made was still a mash-up. We presented it to students as follows.

**Audio-Ethnography: The Soundtrack of Your Life**

***What This Assignment Is About***

The next project is a play on the term “autoethnography.” According to Wikipedia, autoethnography is “a form of autobiographical personal narrative that explores the writer’s experience of life” [[9](#_ENREF_9)]. Autoethnography focuses on the writer’s subjective experience, rather than the beliefs and practices of others.

Autoethnography is now becoming more widely used (though controversial) in performance studies, the sociology of new media, novels, journalism, and communication, as well as applied fields such as management studies.

***What You Are To Do***

Instead of a written narrative, you will be creating an audio narrative of who you are. Your materials will be a collection of music that describes you, reflects your interests, represents the type of music you enjoy, or anything else that will give us an idea of who you are and what makes you tick. You may also include other non-musical sound sources to enhance your presentation.

One piece of music played from start to finish will not cut it. You will need to work with at least a half dozen musical sources that you will edit, process, and layer into a cohesive musical narrative of exactly 300 seconds … not a second more or a second less.

By requiring students’ submissions to be *exactly* 300 seconds, we forced them to use CT skills to map out the timing of the songs they wanted to include rather than just including clips of arbitrary length. In most cases, one wouldn’t want to start or stop a song in the middle of a phrase, although this could be done as a creative technique if it was appropriate to a student’s vision of who he or she is. Clean, logically whole clips are usually much more preferable, and these had to be created of appropriate length so that everything added up to 300 seconds.

This assignment is challenging to do in Audacity for a number of reasons. The first is obviously the task of cutting the clips to the lengths one wants. Then there’s the need to load each clip in the right sequence, which is easier to do if one uses separate tracks rather than trying to do everything in a single track. Next is the issue of transitions. Of course sharp cuts are acceptable if that’s the effect one wants, but most of the time it’s aesthetically more pleasing to hear smooth transitions between songs. Here’s how a CS student described his thoughts on this issue and what he decided to do about transitions:

One major problem I had at first was figuring out a proper order of the piece. Some of the songs had the same tempo, while others were much faster or slower. I decided I wanted many of the clips to run smoothly together, but also to have a few drastic jumps within the piece to make certain clips stand out.

So putting the piece together involves trial and error with lots of “undoing” along the way to get things just the way one wants. And since this assignment is to create something that represents themselves, students take it pretty seriously. Students have to exercise their CT skills considerably to accomplish the mechanics of the assignment.

At the same time, the assignment gets students to listen critically to the music they want to use so that they can create and recombine clips in a way that makes sense. Here’s what one student wrote about listening:

Analyzing the lyrics and the melodies, a lot more than I initially realized could be discovered about me. There are certain qualities the listener could easily realize I possess just by listening to this piece. I would imagine this is exactly what an Audio-Ethnography piece should do.

To demonstrate how seriously some students took this assignment, consider that one student used 44 songs! This student wrote in her reflection:

I spent so much time on this project. I thought about it over and over again. I actually completed it three times. The first time I had maybe only ten songs with much longer selections, but I didn’t feel that it represented me. Then I shortened some so I could add more. I finished it again. This time about twenty five songs. It still wasn’t enough. I finally went through and shortened each clip into only the necessary part so I could add more and more. ... I cannot wait for us to post these on our new websites so when I go home for Easter I can sit down with my family and play it for them and see if they will know it is supposed to represent me.

[*Note to the Editor: At this point we will reference actual sample student projects available as MP3 files on the book website.*]

## The Value of Performance

A major aspect of almost all projects should be an opportunity for students to share their work such as having informal “performances” of the students’ creations. This is where the term *Performamatics* comes from. In the audio-ethnography project, we had students submit their creations and then we played them without telling the class who the composers were. Some of the students knew each other well and some did not. Nevertheless, we asked them to guess who had composed the pieces they heard. A Computer Science student wrote:

... the opportunity to listen to other people’s pieces without knowing who made them ... made the whole assignment come together for me, because it gave us the opportunity to see qualities of our classmates that we normally may not see. Coming from a Computer Science background and not Music, I’ve always found myself a little out of the loop in the classroom, so it was nice to give the class a better glimpse of who I am. It was also an interesting experience to listen to other people’s works and figure out who made them based on the little I know about each student. It also let me experience qualities of people I’d never normally see or music I probably wouldn’t discover on my own.

Another CS student had a somewhat different “take” on the assignment after hearing other students’ creations. He wrote:

While the musical choices themselves were important, I feel that the song order and style of transition also said significant things about the author. Some students organized their projects in a way that sounded best to them, while others tried to convey a message or idea with the order and choice of the music they included in their compositions. Some chose songs that seemed to fit and blend together, while others opted for more variety.

I think that students’ personalities show as much through the composition of their project as their song choices. These were important aspects of this project, and I tried to focus on both. I used music from my library that I felt defines me as a person and reminds me of important past experiences. This project helped me to familiarize myself with my own musical tastes, further my knowledge of Audacity, and was an interesting and fun introspective exercise.

We are strong believers in the value of “performance” as a way to get students to learn from each other. In addition to playing student compositions in class, we make them available on the web. We actually do this with all assignments. It was especially interesting for students to be able to see others’ song flowcharts, for example, since we didn’t have time in class to show and discuss all that work. With the students’ permission (they all signed waivers approved by our Institutional Review Board), we made our assignment website public. You are welcome to view their work at [*URL to be added*].

***Please Note:***We didn’t do the audio-ethnography project in 2011, so you won’t find students’ compositions for this assignment there. However, you will find the Song Flowchart assignment there. Also note that the audio players that appear next to MP3 files in most browsers and OGG files in Firefox require HTML5, so you need to be using the latest version of your favorite browser to have access to these controls.

It is interesting to note two things about posting student work on a public website. First, it completely eliminates cheating. Students can’t possibly copy other students’ work and submit it as their own when it’s all publicly accessible by everyone. Copying would stand out like a sore thumb!

Second, it fosters far more learning than can be accomplished in class, interdisciplinary or not. In regular Computer Science classes, for example, students seldom take a detailed look at other students’ code. Music students, on the other hand, are strongly encouraged and even required to listen to many different types of music, different interpretations, and different arrangements. There is something to be learned from each one. The same is true in Computer Science, but we seldom require CS students to really “get into” other programmers’ code other than to look at brief examples. By posting student work on a public website for all to see, students’ natural curiosity is piqued to the point that at least some examine the code in detail to figure out how their peers “did that.”

Developing CT skills in a collaborative environment focused on exploration and discovery can lead to many surprises. Students often come up with very different ways to approach assignments, including ways that we don’t think of ourselves. They also discover features of the software that we may not have investigated and techniques that we may not have explored. Thus, not only do the students learn from each other by seeing their classmates’ work, but we professors learn as well. When everyone is posting work for others to see, everyone is engaged. What could possibly be a better environment for learning?

## Appendix

### Downloading and Installing Audacity with the LAME MP3 Encoder

Audacity’s home page is http://audacity.sourceforge.net. To download the software, click the Download tab and then click the appropriate link for your system. This takes you to a page with instructions specific to your system (see Figure -7). Click the appropriate link to download the software for your system. Once you save the installation file on your system, execute that file to begin installation.



*Figure 5-7.* The Audacity website Download page.

http://audacity.sourceforge.net/download

There is one more thing you need to do before using Audacity. While *importing* MP3 files into Audacity is straightforward, *exporting* music created in Audacity as MP3 files requires an additional installation step: you need to install the MP3 *encoder*. Like Audacity, the encoder is available free of charge, but it is covered by a separate license, so it is not bundled with Audacity itself.

The good news is that downloading and installing the MP3 encoder is rather easy. Under Optional Downloads on the download page (see Figure ‑8), you will find a link to the LAME MP3 encoder. That link that takes you to a page entitled “How do I download and install the LAME MP3 encoder?” with detailed instructions for both Windows and Macintosh systems. You click yet another link, labeled LAME download page, to get to lame.buanzo.com.ar, which is the website from which you can actually download the required .exe file for Windows or .dmg file for Macs to install the LAME MP3 encoder.[[4]](#footnote-4) Once you have that file on your system, follow the directions back on the Audacity website and voilà!, Audacity will then be able to export MP3 files that play on just about any MP3 device that you or your students may own.



*Figure 5-8*. The Audacity website Download section for Windows systems, showing the locations of the links to download the program and the LAME MP3 encoder.

http://audacity.sourceforge.net/download/beta\_windows

## Bibliography for Chapter 5

[1] Adobe Systems (2011). *Adobe Audition CS5.5*. www.adobe.com/products/audition.html *accessed* 8/9/2011.

[2] Audacity Open Source Development Team (2011). *Audacity: The Free, Cross-Platform Audio Editor and Recorder*. audacity.sourceforge.net *accessed* 8/5/2011.

[3] Avid Technology (2011). *Pro Tools*. www.avid.com/US/products/family/pro-tools *accessed* 8/10/2011.

[4] Berlin, I. (1946). *Annie Get Your Gun.*

[5] Browning, E.B. (1850). *Poems ("New Edition," 2 vols.)*. Revision of the 1844 edition, adding *Sonnets from the Portuguese* and other poems. Sonnet No. 43. London: Chapman & Hall.

[6] Coalson, J. (2008). *FLAC - Free Lossless Audio Codec*. flac.sourceforge.net *accessed* 8/10/2011.

[7] Quinion, M. (2011). *World Wide Words: Prosumer*. www.worldwidewords.org/turnsofphrase/tp-pro4.htm *accessed* 8/10/2011.

[8] Sony Creative Software (2011). *Sound Forge Product Family Overview*. www.sonycreativesoftware. com/soundforgesoftware *accessed* 8/5/2011.

[9] Wikipedia (2010). *Autoethnography*. http://en.wikipedia.org/wiki/Autoethnography *accessed* 8/21/2011.

[10] Wing, J.M. (2009). *Computational Thinking.* Jrnl. of Computing Sciences in Colleges **24**(6):6–7.

1. “prosumer” = near professional, very high-end consumer. [↑](#footnote-ref-1)
2. http://dvdvideosoft.com. This site provides a huge number of image, audio, and video converters, editors, and associated tools, and they’re all free. We don’t know how they can afford to provide all this functionality for free, but we’re very glad they do! [↑](#footnote-ref-2)
3. AU comes from “audio,” not “Audacity.” This file type was developed by Sun Microsystems, the company that gave birth to the Java computer language. Audacity is programmed in Java, so that’s why the AU file type in used to store Audacity’s data. [↑](#footnote-ref-3)
4. Be aware that the LAME download page looks nothing like the Audacity web pages, so don’t be surprised when you click the link and end up on what appears to be a personal, rather unprofessional-looking page with lots of ads and requests for donations. Just scroll down until you see “LAME LIBRARY DOWNLOADS BELOW.” When this book was written, the current version of the LAME encoder for Windows was 3.99.3, while the one for Macs was 3.98.2. [↑](#footnote-ref-4)