HW7—Sound Editor

15 points

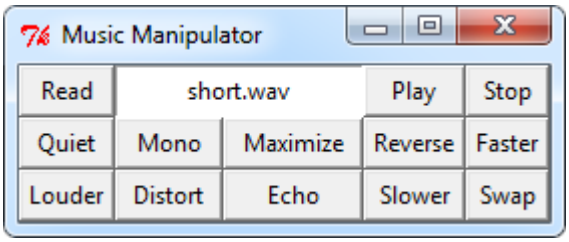
Assignment: Write a program that manipulates and plays sounds, as described below. You'll need to install the PyAudio package; see the back of this sheet.

Due: Friday 10/11, 5PM

Hand In: Submit a copy of your sources to Blackboard Vista. Do not submit your sound files.

Before you begin: You'll need to install the "PyAudio" module. On BBV, you'll find three installers: pyaudio-0.2.7.dmg for Macs, pyaudio-0.2.7.py27.exe for 32 bit PC's, and PyAudio-0.2.7.win-amd64-py2.7.exe for 64 bit PC's. Run the one appropriate for your computer. Also: put the "BCAudio.py" file in the same folder as your HW7 solution.

Write a program that performs sound editing on a .wav file, as demonstrated in class, and as shown here.



Get a copy of the "BCAudio.py" file, and the sample audio files "Vivaldi.wav", and "short.wav" from Blackboard Vista. Put them in the same folder that you put your hw7 solution into. Debug each button using "short.wav" first.

If you want to use your own audio file, I can help with .mp3 to .wav conversion, or you can do it yourself with "Audacity".

In your HW7 solution, do an "import BCAudio", and the following functions will be available to you:

BCAudio.read(filename)	Opens the audio file, and reads the content. Must be called before any of the other functions in BCAudio.
BCAudio.getLeft()	Returns a list of the samples for the left channel of the current song.
BCAudio.getRight()	Returns a list of the samples for the right channel of the current song. You are guaranteed that the left and right channels will be the same length. Each sample will be in the range of -1.0 to 1.0
BCAudio.play()	Plays the song, using the current left and right lists. The left and right lists must be the same length, and all of the values must be in the range of -1.0 to 1.0
BCAudio.stop()	Stops playing the current song.

The functions that manipulate the sound should get the left and right lists, and then make the desired changes to these lists.

- Maximize. Find the maximum of the absolute values of the samples, then divide each element of the lists by that maximum. This should set the largest sampled value to +1.0 or -1.0, which is the maximum allowed by the BCAudio play() method.
- Quiet. Divide the content of each element of the lists by 2. This should make the volume quieter. You might need to press Quiet more than once to notice a difference.
- Mono. Convert the stereo signal to mono. To do this, replace both left and right elements by the average of those two elements.
- Swap Channels. Set the right channel to be the old left channel, and vice versa.
- Reverse. Reverse the content of both the left and right lists. The song should sound backwards.
- Add Echo. Starting about 1/8 second into each list, calculate:
  `left[i] = .25*left[i] + .75*left[i-delta].`
  And likewise for the right channel. Since there are 44,100 samples/second in the .wav file, use delta = 44100/8. Hint: Which is the first element that you should modify?
- Faster. Remove every second element from each list. This should make the sound play twice as fast, though they'll also be octave higher in pitch.
- Slower. Duplicate each element of the lists. For example, if left contains [.1, .2, .3], it should be changed to [.1, .1, .2, .2, .3, .3]. (This will not exactly undo "faster", but it will sound close.)
- Louder. This should take a sound, and make it louder without changing the sample above ±1. Advertisements on TV often do this. Call your maximize function, and then replace each value from the lists with $oldValue^{0.8}$. That is, raise each value to the 0.8 power. But if the number is negative, flip the sign before raising it to a power, and then flip it back, so that it remains negative.

- Distort. Any samples with values greater than 0.2 should be set to 0.2. Likewise, any samples less than -0.2 should be set to -0.2. This is the type of distortion you get when the volume of an amplifier is set higher than the amplifier is capable of delivering. You'll be able to hear the effect best if you Maximize first.

!! Special notes for 64 bit Mac users:

If you get a Python message that says: "Please build and install the PortAudio Python bindings first.", then you're running Python in 64 bit mode. That's normally fine, but unfortunately PyAudio requires that you run Python in 32 bit mode.

To run Python in 32 bit mode, you can't use the normal Run/Run Module (F5) command in Idle. You'll need to run Python from a terminal window.

Open a terminal window (you'll probably need to use Spotlight (the magnifying glass in the upper right corner of the Mac screen) to find it). In the terminal window, you'll need to use the "cd" command to move into your hw7 folder. For example, if you have a hw7 folder on the desktop, do this:

```
cd Desktop
cd hw7
```

Then you can run your program like this:

```
python-32 hw7.py
```

The "python-32" program is a 32-bit version of Python.