COS 116
The Computational Universe
Laboratory 4: Digital Sound and Music

In this lab you will learn about digital representations of sound and music, especially focusing on the role played by frequency (or pitch). You will learn about frequency spectra of sound and about digital sampling, which will give you a concrete insight into everyday technologies such as MP3 and music compression (crucial for Internet and the iPod). One experiment will give you an introduction to the challenges of automatic speech recognition.

If you get stuck at any point, feel free to discuss the problem with another student or a TA. However, you are not allowed to copy another student's answers.

Hand in your lab report at the beginning of lecture on Tuesday, March 4. Include responses to questions printed in bold (number them by Experiment and Step), as well as the Additional Questions at the end.

Before you begin:

1) You'll need to download the following files:
   beethoven.wav
   flutesolo.wav
   grunge.wav
   octaves.sgp

from **http://www.cs.princeton.edu/courses/archive/spring08/cos116/lab4_files/**. Save the files to any convenient location on your computer.

2) Your TA will give you a headset/microphone at the beginning of the lab. This is*not* yours to keep; please return it when you are done with the lab. Plug the headset into the computer's audio jacks, and turn up the volume on the computer. (The laptops in Friend 007 have *two* volume controls – the usual Windows icon on the taskbar, and also physical buttons on the laptops themselves. You may need to fiddle with both to get the volume right.)

3) Most experiments below will use an audio recording and manipulation program called Goldwave. Start Goldwave from the Start menu (the 'Goldwave' program in the 'Goldwave' group).

- To record and replay sounds, use the buttons at the top of the Control window (on the right side of the screen). Click the record button (a red circle) to begin recording; when you are finished, click the stop button (a red square). To play the sound, click the green arrow button.

- Whenever you want to create or record a new sound file, start by clicking the “New” button on the toolbar. When asked, use the following settings: 1 channel, 44100 Hz sampling rate, 10 seconds length. When you save a sound file, accept the default options for “Save as type” and “Attributes”, unless instructed otherwise.

Experiment 1: Recording and Pitch — Meet your “twin” of the opposite sex

If you had a twin of the opposite sex what would he or she sound like?

1. **Click the New button to start a new song file. Click the record button to begin recording. Sing “Row, Row, Row Your Boat” for 10 seconds. Save the file as “song.wav”, and then listen to the result. Repeat if needed.**

2. **Click the Effect menu and click Pitch. If you are a man, increase the pitch by setting Scale to 150%; if you are a woman, decrease it by setting Scale to 70%. Be sure to check the box labeled “Preserve tempo.”**

3. **Play the altered sound and comment in your notes about what you hear.**

Experiment 2: Pure tones

You learned in class that Sine waves are perceived by our ears as "pure tones." Let us listen to them.

1. **Start a new sound in GoldWave. (**Click the New button on the toolbar.)

2. **Open the Expression Evaluator.** (Tools menu, Expression Evaluator)
   The Expression Evaluator creates a sound from a mathematical function you enter. For instance, to create a sound at a single frequency, use the sine function. Enter sin(2*pi**f**t+pi/2), replacing *f* with the desired frequency.

3. **Create a sine wave with a frequency of 1000 Hz, and save as "sine1000.wav".**
   Enter the expression sin(2*pi*1000*t+pi/2) and click OK.

4. **Repeat 1-3 to create a sine wave with a frequency of 500 Hz and one with a frequency of 2000 Hz. Save these as "sine500.wav" and "sine2000.wav".**

5. **Listen to all three sine waves, and comment on what you hear (or don't). What is the relationship between frequency and the sound of a tone?**
   Open each in a separate window for easier comparison.

6. **Try zooming in to the waves and compare them visually.**
   GoldWave displays your recording as a graph. The x-axis represents time, and the y-axis represents the air pressure of the sound at each instant. This is called a "time-domain representation" of the sound. Press Shift+1 to view the waveform at full size (1 sample per screen pixel). Can you see distinct peaks and valleys in the graph? A sound at a frequency of 1000 Hz will have 1000 evenly spaced peaks (and 1000 valleys between them) over 1 second.

7. **Synthesize a sound with two frequencies instead of just one. Use the expression (sin(2*pi*1000*t) + sin(2*pi*2000*t))/2 (the average of the expressions for the two individual sine waves).** Such a wave is called a "superposition."

8. **Listen to your two-tone sound and predict what the two-tone wave will look like. Examine it by zooming in (Shift+1), and sketch a picture in your notes.**

9. **Use the Expression Evaluator to create a *square wave* with a frequency of 1000 Hz. Use the expression int(2*t*1000)%2*2-1. Save as "square1000.wav".**
   While a sine wave smoothly transitions between peaks and valleys, a square wave shifts instantaneously from maximum to minimum value.

10. **View both "sine1000.wav" and "square1000.wav" at full size (Shift+1) and compare them visually. Listen to them and compare what you hear. Which sounds more like a "pure" tone to you?**

Experiment 3: Sampling Rates and Quantization Noise

1. **Reopen the file "sine1000.wav" that you created in Experiment 2.**

2. **You created the file at a sample rate of 44,100 Hz. Now resample it at a rate of 2000 Hz and save it as "sine1000-2k.wav"**
   On the Effect menu, click Resample. Enter a rate of 2000 Hz.

3. **Open the original "sine1000.wav" again and resample it at a rate of 4000 Hz. Save the result as "sine1000-4k.wav".**

4. **Repeat Steps 1–3 for "sine2000.wav". (Save under corresponding filenames ending in "-2k.wav" and "-4k.wav".)**

5. **Nyquist's Theorem says that to capture the bare essence of a wave, one must sample it at a rate that is no less than twice the frequency. Predict what the four resampled sine waves will sound like. Listen and compare your results.**

6. **Create a new sound file and record yourself speaking the numbers "one, two, three, four, five, six." Save the recording as "numbers.wav".**

7. **Resample "numbers.wav" at 8000 Hz and 1000 Hz. Save the results as "numbers-8k.wav" and "numbers-1k.wav".**

8. **Predict what the resampled files will sound like, then listen to the files and compare the results. What does this imply about the range of frequencies that are important for understanding human speech?**

9. **Open the original "numbers.wav" file again. From the File menu, click Save As. Name the new file "numbers-8bit.wav"; for Attributes, select "PCM unsigned 8-bit, mono."**

When you spoke into the microphone, your voice created a fluctuating electric signal. This signal was analog—a smooth wave. The computer converted it into a digital sound file where each sample is represented by 16 bits of data, so each analog sample had to be rounded to one of 65,536 (2^{16}) different values. This process, known as *quantization*, replaced the smooth wave with a stepped one, causing a loss of fidelity.

It's difficult to hear the quality loss in the original recording because of the large number of values available for each sample. However, the copy that you just saved represents each sample in only 8 bits of data, so each sample was rounded to one of only 256 (2^8) values. The result is a much coarser approximation. In the lecture this was termed *quantization noise.*

10. **Close and re-open "numbers-8bit.wav." Predict what the file will sound like, and then listen to it. Describe the effect of the coarser quantization.**

Experiment 4: Spectral analysis of Speech and Music

Up to this point you focused on the time-domain representation of sound. Another way to represent sound is in the frequency domain. Every wave can be viewed as the superposition of sine waves of different frequencies. This fact is called Fourier's Theorem and it yields an alternative way to represent sound called *frequency domain representation* or "spectrogram." As mentioned in the lecture, the human ear perceives sound according to the precise mix of frequencies in it, and how this mix changes over time. As in the time-domain representation, the x-axis in the spectrogram indicates time, but now the y-axis indicates frequency rather than amplitude. Each point is assigned a color that indicates the intensity of that frequency at that instant.

1. **Turn on the spectrogram display.**
   GoldWave's Control window (on the right side of the screen) displays two different visualizations of the audio as its playing. Right-click on of the top half of the Control window, and select "Spectrogram" from the menu that pops up. If the spectrogram does not have scales for the x and y axes, right click again and click Properties; then check the box labeled "Show axis."

2. **Listen to the file "sine1000.wav" that you created in Experiment 2 while watching the spectrogram display. Notice how the 1000 Hz frequency appears in the spectrogram. Pay attention to the y-axis (frequency) scale.**

3. **Listen to the file "square1000.wav" and observe its spectrogram.**

4. **How does the spectrogram of the sine wave compare to the spectrogram of the square wave? How does this relate to what you hear in each file?**

5. **Open "numbers.wav" and display its spectrogram. Compare the spectrograms of the numbers "four" and "six". How does the spectrogram relate to your findings in Experiment 3, Question 8 (resampling).**

6. **Open the file "flutesolo.wav" and observe its spectrogram. How are the pitches of the notes reflected in the display?**

7. **Open the file "grunge.wav" and observe its spectrogram. Comment on the appearance and sound of the two different sections (0-7 seconds, vs. 7-14 seconds).**

Experiment 5: Phonemes and how to recognize them

Automatic speech recognition (transforming spoken speech into written text) is a difficult problem. This experiment explores some reasons why.

Phonemes are the smallest meaningful units of sound in spoken language. For example, the word "glad" is made up of the phonemes /g/, /l/, /æ/, and /d/. One would naively think that all that the computer needs to do is examine the sound file of a person speaking and identify the individual phonemes. Assuming the human speaker pauses between words a little, this ought to give a good approximation to the individual words. Unfortunately, spotting phonemes in natural speech is a difficult task.

1. **Create a new sound file and record yourself speaking the phrase, "show me the money."**
   Speak as you normally would; don't add extra pauses.

2. **Try to select each individual word without selecting any sound from the previous or next words. Is this always possible?**
   To select part of the sound, click and drag the mouse from left to right over the target region. Click the green play button to listen to the selection.

3. **When we speak, do we produce phonemes one at a time (*linearly*), or blended together? Try to select only the "th" sound in the word "the"; try to select only the "uh" sound. Is this possible?**
   This is called the problem of *non-linearity.*

4. **Select part of the "ee" sound in the word "me" and listen to it while examining its spectrogram. Do the same for the "ee" sound in the word "money". Does a particular phoneme always correspond to the same mix of frequencies in your speech?**
   This is called the problem of *variance*.

Experiment 6: Understanding the Scribbler's sound generator, octaves, and pitch

1. **Use Scribbler Control Panel to open the file "octaves.scp." Examine the pseudocode and load it onto the Scribbler.**
   This program plays a series of sounds that cover three octaves on the musical scale.

2. **Start the program on the Scribbler, and listen to the tones. How does the pitch of a note correspond to the frequency specified in the pseudocode? (This is easy.)**

3. **Create a new sound file in GoldWave, and record the sounds produced by the Scribbler program.**
   Position the microphone about six inches above the Scribbler's speaker. Start recording, and then start the program on the Scribbler.

4. **Select one of the tones in GoldWave (drag the mouse from left to right over the wave) and zoom in to see the waveform. Does the waveform look like a sine wave or a square wave?**

5. **It is reasonable to assume that Scribbler, since it contains a microprocessor, is capable in principle of generating a sine wave. In your notes, speculate on why the recorded wave is not a pure wave like the one you generated in Experiment 2.**

6. **Identify pairs of tones that are the same note in different octaves, and examine them using the spectrogram. In terms of frequency, what does it mean for them to be the same note? What is an octave?**

Experiment 7: Compression

1. **Open the file "beethoven.wav".**

2. **Note the length of the recording (in seconds). GoldWave displays this in the lower left hand corner of the window, to the right of the word "stereo" or "mono."**

Files in .wav format are not compressed, so you can calculate the size of the file (in bytes) from its length and recording parameters. This file was recorded in near-CD quality: 1 channel (monophonic), 44,100 samples per second, and 2 bytes (16 bits) per sample per channel. Below you are asked to do some calculations in your report which you then verify by experiment. If you are short of time, just do the experiments and finish the calculations later.

3. **Calculate how many bytes the uncompressed audio format needs to record one second of sound.**

4. **Calculate the size of "beethoven.wav" in bytes. Check your prediction by comparing it to the actual size of the file. (It should not differ by much.)**
   To find out the real size of the file, click the File menu and click Open. Right-click on the file and click Properties. Look at "Size", not "Size on disk".

5. **Save a copy of the file using MP3 compression and a data rate of 128 kbps (kilobits per second). Name the copy "beet128.mp3".**
   In the Save As dialog box, set "Save as type" to "MPEG Audio (*.mp3)" and set "Attributes" to "Layer-3, 44100 Hz, 128 kbps, mono."

At a data rate of 128 kbps, the compressed file uses 128 kilobits to store each second of sound. There are 1000 bits in a kilobit, and 8 bits in a byte.

6. **Calculate for a 128 kbps MP3 file how many bytes are needed to store one second of sound.**

7. **Calculate the size of "beet128.mp3", and check your answer by comparing it to the actual size of the file.**

8. **Repeat steps 5-7, but create an MP3 compressed file with a data rate of 48 kbps, and save it as "beet48.mp3".**
   This time, set "Attributes" to "Layer-3, 44100 Hz, 48 kbps, mono."

9. **What is the compression ratio for the 128 kbps MP3 file? For the 48 kbps MP3 file?** (The compression ratio is the uncompressed file size divided by the compressed file size.)

We say MP3 compression is "lossy" because some information about the audio signal is lost when the compression is applied. Check whether you are able to notice the absence of this lost information.

10. **Compare the uncompressed, 128 kbps MP3, and 48 kbps MP3 versions of "beethoven" by listening to them with headphones. Do you notice any distortion in any of the compressed clips? Why do you suppose MP3 is such a popular music format? If you did hear some distortion, what did it sound like, e.g. which instruments were most affected?**

Additional Questions

1. **The telephone system uses a sampling rate of 8000 Hz. Why don't you think they use a 1000 Hz sample rate?**

2. **If you had a 20 Gigabyte iPod, how many minutes of music could you store on it, if it were compressed using the 192 kbps MP3 encoding setting?**

3. **Explain Nyquist's Theorem at an intuitive level in your report, using the sine wave as an example. What happens if the sampling rate becomes less than twice the frequency?**

4. **Given what you know about the Scribbler's capabilities, can the robot be programmed to say "Hello"? Speculate.**