October 30, 2017

CMSC 178.1: Introduction to Multimedia

Programming Exercise 1 Specifications

I. Program Features

The program must be able to:

1. Generate a pure tone (sine wave) of a specific frequency and save it to a WAVE (.wav) file
2. Generate a sequence of pure tones that follows the Just Intonation or 12-Tone Equal Temperament System and save the result to a WAVE file.

Program Flow:

The program must be a console program written in C++. The library to be used for writing WAVE files is at: https://github.com/adamstark/AudioFile. How to use the library is documented on the Github page. I suggest you read and try out the basic features first. For the whole exercise, writing 44.1 kHz/16 bits (signed integer) WAVE files is the only feature we'll use.

When launching the program for the first time, it must prompt the user for two (2) options:

```
1. Generate a Pure Tone
2. Generate Tuning Sequence
Choose a feature:
```

If the user chooses feature no. 1, the program must ask the user to specify the frequency of the pure tone:

```
Frequency of the pure tone in Hertz:
```

The allowed input must be strictly a **positive integer greater than 0**. After the user confirms the frequency value, the program must generate a pure tone (sine wave) of that frequency 10

seconds in length and save the generated audio to a WAVE file named "**sine-<frequency-in-hertz>.wav**".

For example, if the user specifies 1000 as the frequency, then the program generates a 1000 Hertz (1 kHz) pure tone 10 seconds in length and save it to "**sine-1000.wav**" in the same directory as the program (.exe). **No need for the program to play back the WAVE file.**

The output WAVE file must be a standard, uncompressed data which the C++ library above writes by default. The sampling rate and bit depth of the output WAVE files must be 44.1 kHz and 16 bits signed integer, respectively.

After successfully saving the pure tone to the WAVE file, the program must inform the user:

```
Pure tone generated and saved to "sine-1000.wav"
Press Enter key to exit…
```

The program simply quits when the user closes it using the "X" button of the program window or when the user presses the Enter key. Done!

If the user chooses feature no. 2, the program must prompt the user for another set of options:

```
1. Just Intonation
2. 12-Tone Equal Temperament
Choose a Tuning System:
```

If the user chooses **Just Intonation**, the program must prompt the user for a starting frequency:

```
Frequency of "Do" in Hertz:
```

The frequency must be a **positive integer equal to or greater than 20 Hz**. This frequency will be the lower *Do* in the sequence *Do-Re-Mi-Fa-So-La-Ti-Do*. The goal is to generate all the

other pitches starting from this frequency. Remember that the pitches are related to each other via frequency ratios relative to the lower *Do.*

Do-Re-Mi Name	Musical Interval Name	Ratio	Distance from Unison	Sample lower *Do* = 100 Hz
Do	Unison	1	1.00	100
Re	Major Second	9/8	1.125	113
Mi	Major Third	5/4	1.25	125
Fa	Perfect Fourth	4/3	1.33	133
So	Perfect Fifth	3/2	1.50	150
La	Major Sixth	5/3	1.67	167
Ti	Major Seventh	15/8	1.88	188
Do	Octave	2	2.00	200

NOTE: The frequencies resulting from the calculations will have to be rounded off to the nearest whole number values. This is to make things simpler.

What the program must do is generate pure tones 1-second long each starting from lower *Do* (the frequency specified by the user), then a 1-second pause (silence), then *Re*, then a 1-second pause (silence), then *Mi,* etc.

The end result must be a WAVE file containing a 15-second long audio that goes like this:

Do - pause - Re - pause - Mi - pause - Fa - pause - So - pause - La - pause - Ti - pause - Do

If the user chooses the **12-Tone Equal Temperament (12-TET)**, the program must prompt the user for a starting frequency:

```
Frequency of Unison in Hertz:
```

The frequency must be a **positive integer equal to or greater than 20 Hz**. This frequency will be the unison. Unlike in Just Intonation, the individual notes except the unison (starting pitch)

and octave in Equal Temperament do not have definite pitch names. This is because they are not exactly equal to their musical counterparts (again, except the unison and octave).

The goal is to create fill in the notes between the unison (the frequency specified by the user) and the octave (twice the unison). In the 12-TET system, there are 12 unique notes in an octave. 13 if you include the octave itself.

The 12 TET system goes like this:

1st / Unison -> **2nd** -> **3rd** -> **4th** -> **5th** -> **6th** -> **7th** -> **8th** -> **9th** -> **10th** -> **11th** -> **12th** -> **13th / Octave**

Each of these note are *half-steps* apart in musical terms. The 8th / octave is, of course, 2x the frequency of the 1st / unison.

To calculate the frequencies of each of the notes starting from the unison, follow the formula :

$f_n = f_0 * (a)^n$

f_n is the frequency you're trying to calculate. So, if you're calculating the 2nd, this should be f_1
f_0 is the frequency of the unison, which is specified by the user
a = twelfth root of 2, which in our case, will be equal to **1.059463**
n = the index of the frequency you're trying to calculate. Again, if 2nd, this should be 1 since unison is index 0

Same as in Just Intonation, each of the notes must be 1-second long, and separated by 1-second pauses.

For Just Intonation, the output WAVE file must be named "just-intonation-<frequency-of-lower-Do-in-Hertz>.wav". For 12-TET, the WAVE file must be named "12-TET-frequency-of-Unison-in-Hertz>.wav". Same as in Feature 1, after the generated audio is successfully saved to a WAVE file, the program must inform the user:

```
Tuning sequence saved to "just-intonation-100.wav"
Press Enter key to exit…
```