Audio Scavenger Hunt

Instructions

- Download **Resources → Project3 → mystery.wav** from Sakai. Optional: also download **wavCode.cpp**, **wavCode.h,** and **main.cpp** if you want to use them. The .wav file contains a hidden message embedded in a series of 7 tones produced using the DTMF definition of dial tones. Your task is to decode the message by identifying the tones using a FFT function that you write and utilize.

Overview

- The first task is to write a driver program to read the sound amplitude values from *mystery.wav*. You're welcome to do this from scratch but I have provided a C/C++ program named *wavCode.cpp* that will read a file and return an array of integers containing the amplitudes of the data. I've also given you *main.cpp* that illustrates how to use the code. The main() function also prints out information from the .wav file *header* and returns the original SampleRate through a reference parameter. The function I have provided assumes that the .wav file contains 16 bit numbers in little-endian format.
- Your second task is to write a FFT function that will receive an array of N (Complex) numbers and return an array of Fourier coefficients. Note1: Assume that N is a power of 2. Note2: As is always true using FFT, only the first N/2 Fourier coefficients returned are useful, as the values will be symmetric around the middle value. Note3: It is actually easier to do the FFT "in place" than to make copies of the values inside the FFT function. That means that you can return the Fourier coefficients in the same array that you send the values into the FFT function.
- The next goal is – for each of the 7 tones – identify the two frequencies associated with the highest values returned in the Fourier coefficient array. That will allow you to identify the phone key used to generate each of the tones.
- The final goal – the least important one – is to figure out the original message.

Details

- The data was sampled at 8000 Hz, a fact that you can recover from the .wav file header. Each tone was 0.1 sec in duration with a 0.1 sec gap between tones.
- Your FFT function will work on arrays whose size is a power of 2, hence you will want to send to your FFT function a "window" of each tone's values whose size is a power of 2.
- Some of your code will need to work with complex numbers. Fortunately, C++ has complex data types that you can learn about or you can simply use two parallel arrays to store complex values, one for the real part and another for the imaginary part. You should only need to add, multiply, and compute the magnitude of complex values.
- The values returned from the FFT function will be complex numbers. The magnitude of these values is called the power spectrum of the signal. You'll want to compute the magnitude of the values returned from the FFT function in order to compute the power spectrum and identify its two peaks.
- You can look up the DTMF frequency values on Google.

Grading

- 50% - Correct implementation of the FFT on a 1-d array of numbers.
- 10% - Documentation & readability
- 35% - Can use your FFT code to identify the two predominant frequencies in each of the 7 tones and to then identify the 7 phone keys associated with those tones.
- 5% - Identify the encoded mystery message.