Eugene Davis
CPE 381
Project Phase 1 Report
03/10/14

Project Description

Phase 1 of the project requires that a WAV file be read in from disk in a stream-like fashion, have a 2,500 sine wave added to it at half the maximum amplitude of the audio in the WAV file, and then have this modified data be written back to disk. During this process, the entire WAV file's data was not to be stored in memory, in order to be like the restrictions found in embedded hardware. It then requires a file summarizing the time taken, sampling frequency and recording length be outputted.

This project solves the requirements from above by reading in the header into a struct matching the header's data. From this, the software knows the size of the file, the sample size, and the number of channels. Using this information, a void pointer is setup to point to an array of a type appropriate to the sample size, here a char array for 8-bit and a short array for 16-bit. At the same time, function pointers are used to point to 8- or 16- bit versions of the functions which find the maximum amplitude of all the channels in a sample and add the sine wave signal to the data. Using void and function pointers here reduce the amount of repeated code within main, and make the code easier to maintain and extend in the future.

With the sample arrays and function pointers setup, the header for the modified WAV file is written to disk by copying the memory space of the header struct back to disk. After this, the code loops through all the samples to find the maximum amplitude of the data in the file. With the amplitude found, it resets the file pointer to the start of the data in the WAV file, and loops through all the samples again, this time adding a sine wave of half the maximum amplitude of the original signal, then saving it to the new WAV file.

Finally a summary text file was written, listing the file names involved, as well as the sampling frequency, recording length and processing time required. Unlike the other file access portions, this using ofstream rather than fread and fwrite, since it makes the code more readable and is outside the scope of the timer.

WAV File Format

The WAV file format is a container which is flexible in that it can contain audio of many structures and encoding types. The most common (and supported in this project) is PCM which is basically raw data, but it can even contain MP3 files. It is based on the RIFF (Resource Interchange File Format), which allows data to be stored in tagged chunks within the file, but most commonly for WAV has only a RIFF type chunk which provides the header information, and a data chunk containing the audio information. In the context of a WAV file, the differing chunks could be used to support multiple encoding types within the same audio file.

The header for WAV provides a variety of control fields. The name, a short description, and location (by offset) in the file are listed in Table 1.

Eugene Davis
CPE 381
Project Phase 1 Report
03/10/14

Name	Description	Location (Offset)
Subchunk{1,2}Size	Size of the subchunk following this control field	16 (sub chunk 1 – WAV format), 40 (sub chunk 2 - data)
AudioFormat	Format the audio is in, for PCM (raw) it is 1, otherwise some other compression	20
NumChannels	Number of channels in the file	22
SampleRate	Rate at which the samples are taken	24
ByteRate	Rate (in bytes) at which the samples are taken	28
BlockAlign	Number of bytes for a sample and all its channels.	32
BitsPerSample	Number of bits per sample.	34

Table 1: Control Fields in WAV header

Summary Text File

```
Input File Name: Davis_E_orig.WAV

Output File Name: Davis_E_processed.WAV

Sampling Frequency (samp/s): 44100

Recording Length (s): 59

Processing Time (s): 0.34
```

Due to the algorithm required, it is impossible for this to be done in real time. The audio source (i.e. the WAV file) must have already ended before its maximum amplitude can be found, and adding the sine wave is dependent upon knowing the maximum amplitude.

If not for this fact, the program could clearly work in real time, as processing a file of 59 seconds completed in 0.34 seconds. This also, however, may not hold true on some systems, as a bottleneck such as the storage medium, CPU, or memory may reduce the speed. For example, the summary file included here was generated on the Linux boot of a desktop system. Working from the Windows boot, on the same hard drive, the processing time required over 3 seconds. Thus real time processing, even ignoring the data dependency, also has hardware dependencies.

Eugene Davis
CPE 381
Project Phase 1 Report
03/10/14

Sources

1. WAV Header Information: https://ccrma.stanford.edu/courses/422/projects/WaveFormat/
2. WAV Header Information: http://www.sonicspot.com/guide/wavefiles.html
3. WAV Header/RIFF Information: https://en.wikipedia.org/wiki/Resource_Interchange_File_Format