Math 321 Programming Project I - Spring 2024

Moorer Reverb Filter

Please submit all project parts on the Moodle page for MAT321/598. You should include all necessary files to recompile, and a working executable, all in a zipped folder (one file for upload). Time-stamp determines the submit time, due by midnight on the due-date.

Due: Friday, Feb 10

Programs should compile under g++.

Implement the filter in Chapter 14 section 2, according to the diagram on page 292, Figure 2.3. Note that each comb in that diagram should be a lowpass-comb as in Figure 2.4 on page 293.

The basic program will simply take in a mono wave file (16 bit, sample-rate 44100) of length about 5 seconds, and output a wave file with reverb algorithm applied to it. This program will have hard-coded filter parameters (which we can change later with an interface).

Use the filter parameters described in the text, with slight adjustments to accommodate for our sample rate 44100 Hz. This means we will use all-pass coefficient $a = 0.7$, and direct wave to all-pass output factor $K = 9$. The delay values L_i for the six lowpass-comb filters $i = 1, \ldots, 6$ should be computed according to the number of samples to match the suggested delay values in the text in milliseconds: 50, 56, 61, 68, 72, and 78. The g_i values, for the six lowpass-comb filters $i = 1, \ldots, 6$ should be taken as in Moorer's original paper where they are given for sampling rates 25 and 50 Khz. You should interpolate linearly between these g values to obtain the values for 44.1 Khz. The gain constants R_i for each of the lowpass-comb filters should be computed as in the text by solving for the R_i in $R_i/(1 - g_i) = 0.83$.

Also, normalize the output to -1.5 dB. Call the final program reverb.cpp.

The program should run on the command line as:

./reverb $< input.wav >$

Output should be a new wave file output.wav

Update: You may want to use JUCE from the start, rather than doing this first phase of the project as a command line program. It is pretty straight forward to use JUCE to read in a wav file, access the samples as floats, modify those and write to a buffer, write the buffer to a new wav file, and play back audio from the new wav file or from the buffer. This could be done with a minimal interface, then the interface could be enhanced for project 2. If you are new to JUCE, I recommend you start with two tutorials: Playing Sound Files, and Sine Synth.