Coneko: A Simple Convolution Reverb Plugin

MUMT-307 Final Project Report

Eto Sun
eto.sun@mail.mcgill.ca

April 20, 2022

1 Introduction

Convolution reverb is a widely used audio technique to simulate the reverberation characteristics of an acoustic space. The convolution reverb calculates the convolution between the input audio signal and impulse response (IR) from a linear time-invariant (LTI) system. The direct convolution has a heavy computation in the time domain and it's nearly impossible to be computed in real-time. One solution is to use the fast Fourier transform (FFT) algorithm with some specific optimization to calculate in the frequency domain, which drastically improves performance.
Therefore, the main objective of this project is to implement a convolution reverb plugin using JUCE[1], a partially open-source and cross-platform C++ application framework for developing audio applications and plugins. This report discusses the process structure of the plugin, and the personal experiences while implementing the program.

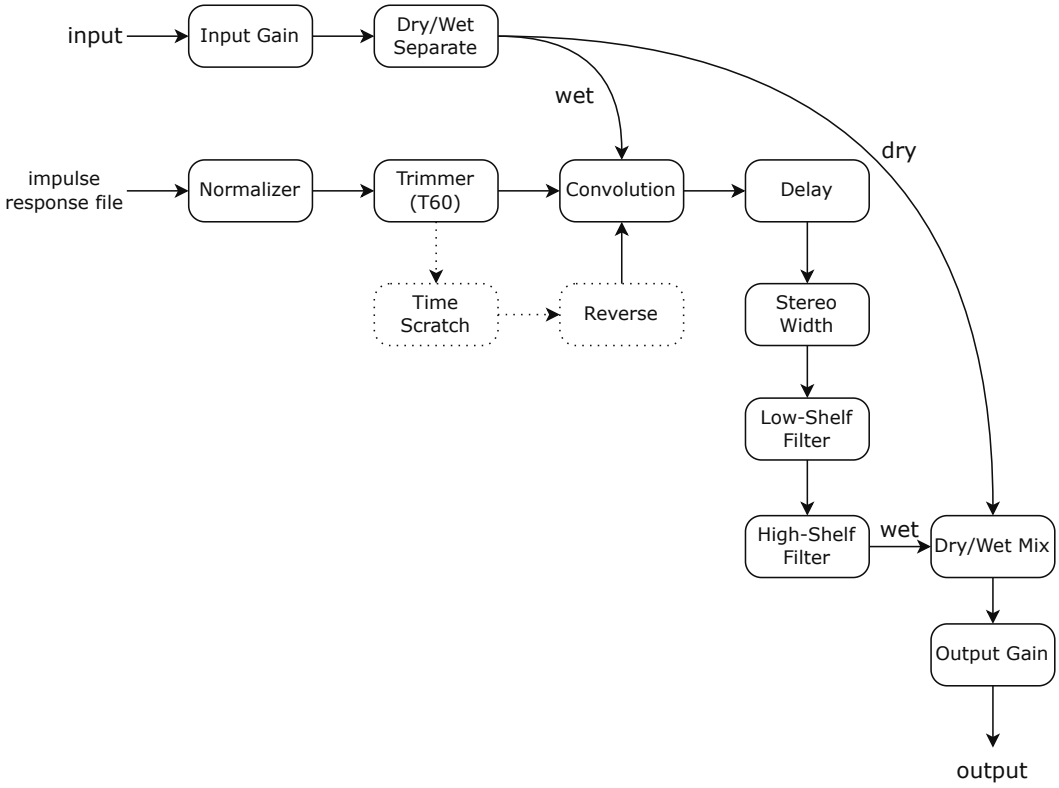

Figure 1: Signal flow diagram of the audio plugin

1. https://juce.com/

2 Plugin Structure

The plugin consists of three parts, and the signal flow diagram is shown in figure 1. Figure 2 shows a screenshot of the user interface of the plugin. The code of this project is hosted on GitHub.[2]

2.1 Input and output signal control part

Input gain (in dB), output gain (in dB), and dry/wet signal mixing ratio can all be controlled. Setting the dry-wet signal mixing ratio to x% means that x% of the input signal will be processed by the reverb and after-reverb effects.

2.2 Convolution reverb control part

A button is provided for loading an impulse response audio file (in WAV or AIFF format) from the computer. The plugin will first normalize the amplitude of the IR to the range of -1 to 1. Then trim the IR based on the time range between the amplitude of IR reaching 0.001 (-60 dB) for the first time and for the last time (T60 time, the time to decay by 60 dB). The IR name and amplitude plot will be displayed in the interface. Then the user can modify the decay time (the T60 time, in seconds), the pre-delay time of the reverb (in milliseconds), and the stereo width. There is also a checkbox for reversing the IR for special sound effects.
I implement the decay time control by using a time-scretch library named SoundTouch,[3] a C++ library for changing the speed of the audio without modifying the pitch. And I convert left/right stereo signals into mid/side signals for controlling the stereo width.

2.3 After-reverb effects part

There are a low-shelf filter and a high-shelf filter in this plugin. Users could change the gain and cutoff frequency of these two filters. The quality factors of shelving filters are fixed at Q = 0.7.

3 Personal Experiences

There are two reasons for me to choose JUCE for developing this plugin. Firstly, it is not easy to convolve a long impulse response (several seconds long) in real-time, even if using FFT. JUCE provides a stereo uniform partitioned algorithm to optimize the performance of convolution. There exist some non-uniform partitioned convolution algorithms for better performance. However, they are usually used in commercial software and are not well documented. Secondly, JUCE provides a convenient way to create a GUI for the audio plugin that does not require preprocessing of user inputs.
The main challenge of implementing the plugin is to learn JUCE from scratch. JUCE has a steep learning curve because of hundreds of classes and fragmented documents and tutorials. Once I understand how JUCE handles audio signals and processings, developing audio effects becomes much easier.

2. https://github.com/etosphere/coneko/
3. https://www.surina.net/soundtouch/index.html

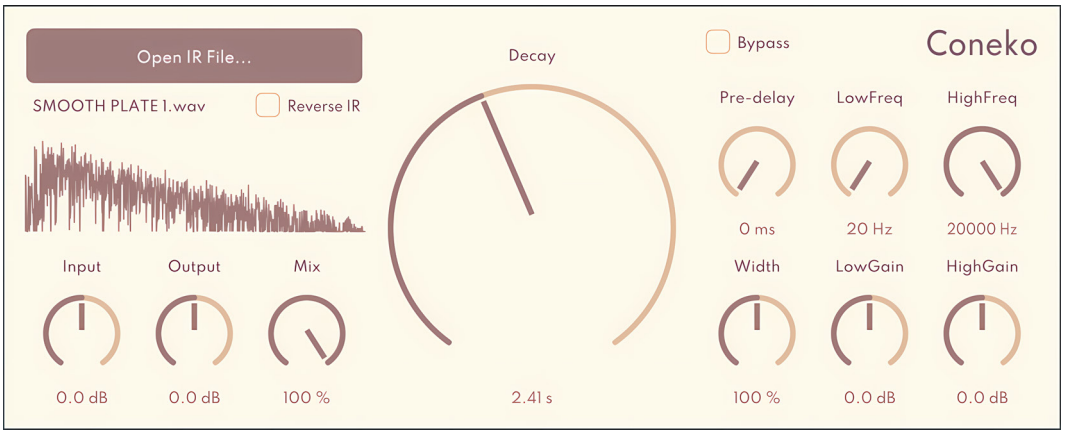

Figure 2: User interface of the audio plugin