Stairway to Hamershlag

Electrical and Computer Engineering Department
Carnegie Mellon University
Matt Kasper, Stephen He, Joseph Kim
{mkasper, she2, youngchk}@andrew.cmu.edu

Overview

This project introduces a software tool aimed at simulating guitar effects against both live and recorded audio sources. We implement digital effect blocks to emulate common pedal types such as fuzz, reverb, delay, and distortion pedals. We also implement a circuit simulator capable of performing real-time transient analysis on audio signals as they pass through circuits composed of resistors, capacitors, inductors, and diodes.

Ultimately, this suite of tools is packaged into a user-friendly application that allows anyone from musicians to experienced audio engineers to seamlessly experiment with new sounds and effects.

Motivation

Guitar pedals are circuits that one can insert in between a guitar and a speaker to alter the sound in some unique way. To produce these pedals, designers build their circuit designs with physical components and test them by playing audio through the circuit. To adjust the sound of the pedal, they have to recreate and change the circuit physically, leading to a slow and costly process.

Our Stairway to Hamershlag application allows a pedal designer to design and iterate guitar pedals more efficiently by letting the designer simulate their circuit designs on live or recorded audio in software.

System Architecture

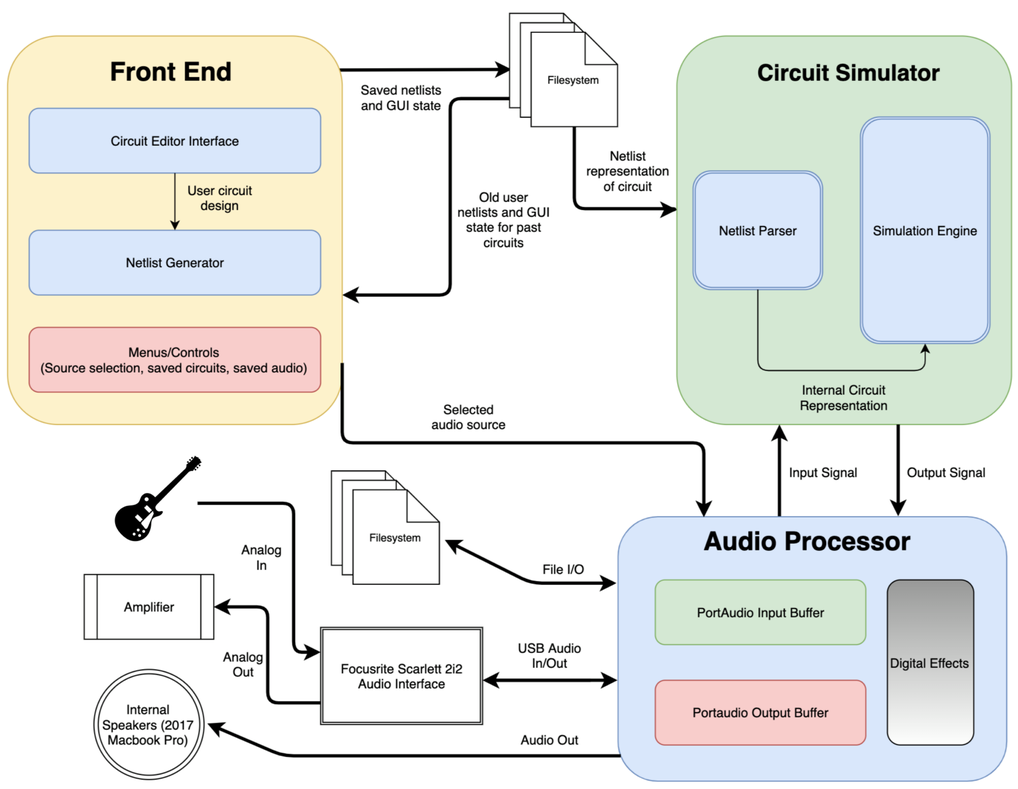
The system architecture is composed of three main modules:

- Frontend (NodeJS / Electron)
- Circuit Simulator (C++)
- Audio Processor (C++)

The **frontend** allows the user to capture user circuit designs and select input and output sources. It manages projects and generates the files required by the backend.

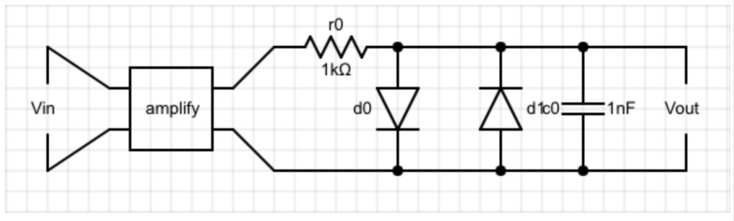
The **circuit simulator** uses the circuit received from the frontend to perform transient analysis on the circuit. It uses Newton's method to approximate solutions to the KCL equations it derives.

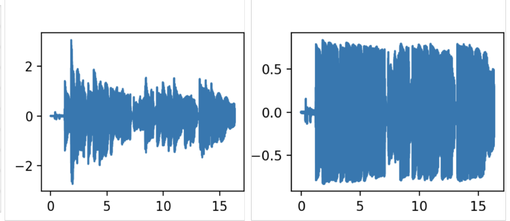
The **audio processor** is the gateway for all sounds going in and out of the system. Audio is routed to the circuit simulator and the results are received back, after which it is saved and/or played. The audio processor also applies any digital effects, which are applied before sending it to the circuit simulator.



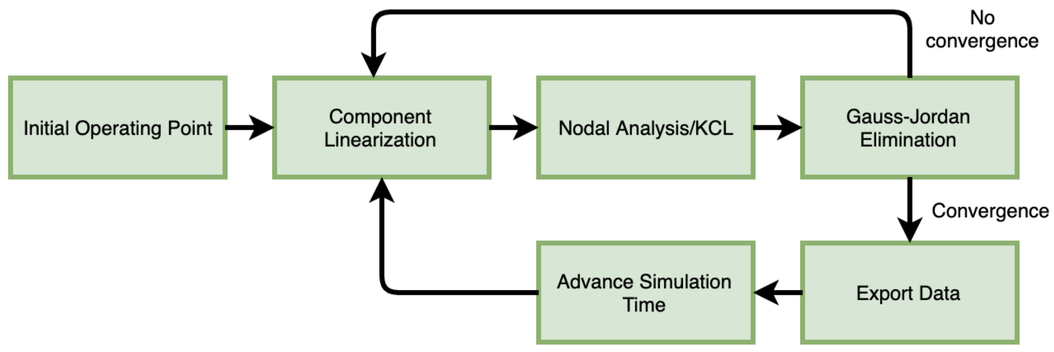
Approach

Frontend: The frontend application is built in Node.js using Electron as it's framework. It parses the user circuit description into a netlist for the backend to process.





Circuit Simulator: The circuit simulator obtains a stream of voltages from the audio processor. For each voltage, KCL is performed at each node to produce a system of equations, which solved using Gauss-Jordan Elimination.



Audio Processor: The audio processor uses Portaudio to interact with the audio hardware and libsndfile to parse various audio files. For live audio, samples received and to be sent are buffered to account for processing time. Any requested digital effects are instantiated and applied to the samples in the order specified before sending it to the simulation.

Evaluation

Though it varies from run to run, a fairly low average latency of 23.2ms was achieved with certain circuits.
The following results were obtained from a user study:

Survey Question	Average Score [1,10]
Overall Satisfaction	8.6
Ease of Use	7.4
Unobtrusiveness of Latency	9.0

Carnegie Mellon

Electrical & Computer ENGINEERING