# Developing a Versatile Audio Synthesizer
# Victor Shepardson
# Computer Systems Lab 2009-2010

## Abstract

A software audio synthesizer is being implemented in C++, capable of taking musical and non musical input information and using additive and FM methods of synthesis to achieve rich spectra, vibrato, tremolo, and smooth pitch change effects.

## Background

Electronic sound synthesis has been of interest to musicians, electrical engineers and computer scientists for as long as it has been practical. The goal of this project is to create an easy-to-use piece of software for exploring multiple methods of sound synthesis using digital oscillators.

**Additive Synthesis**

An audio signal which behaves periodically over long enough time domains can be represented by a collection of sine waves with different phase, frequency and amplitude called a spectrum (Moore). Additive synthesis creates audio signals by superimposing waveforms or by using Fourier Transforms to convert spectra to audio signals.

**FM Synthesis**

Frequency Modulation synthesis uses one signal to modulate the the frequency of another, producing an harmonically complex signal with just one oscillator. In FM synthesis, the carrier and modulating frequencies are both in the audio band; the result is an output signal which contains the carrier frequency as well as many audible sideband frequencies.

**Cross Coupled Oscillators**

In the case of cross coupled oscillators, two oscillators are linked together, the output of each modulating either the amplitude or frequency of the other. This can produce many kinds of temporally varying spectra, from insect-like buzzing sounds to running water to unpredictably shifting noise (Miranda).

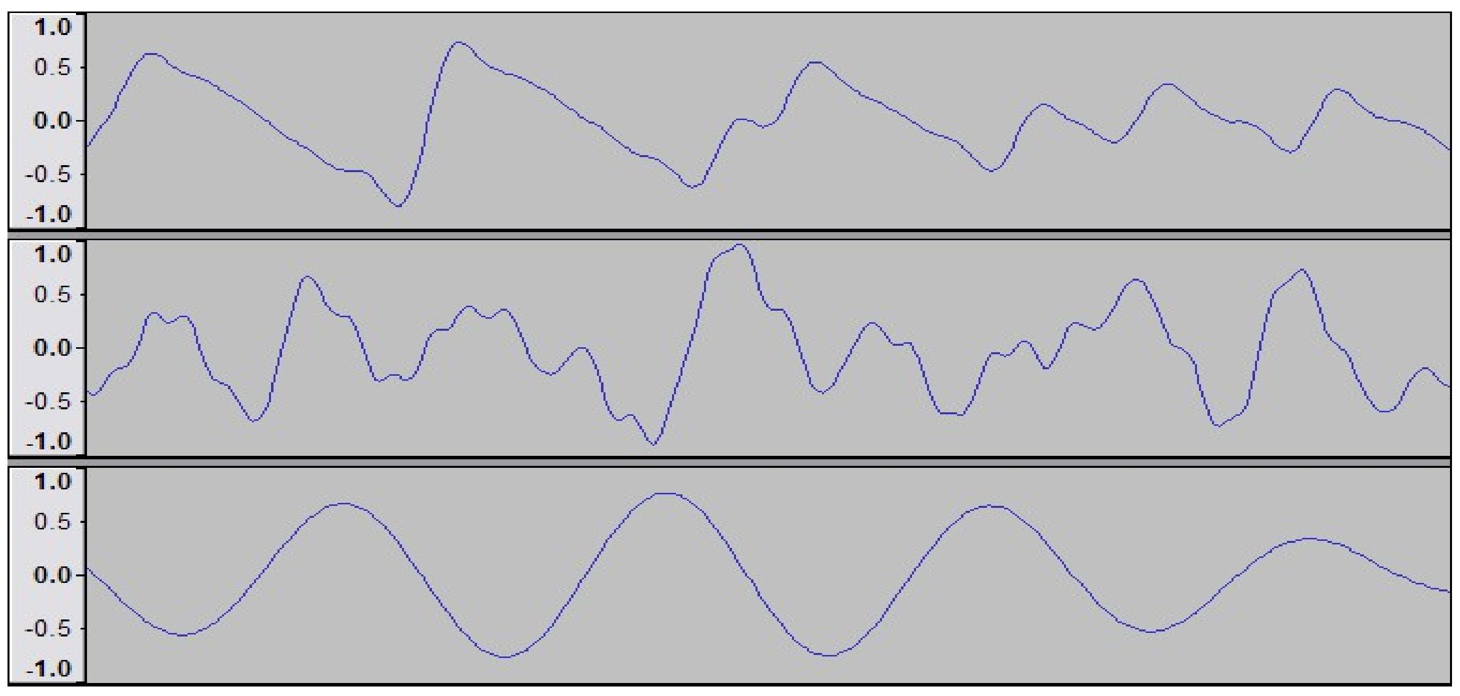

Fig 1: waveforms generated using additive synthesis

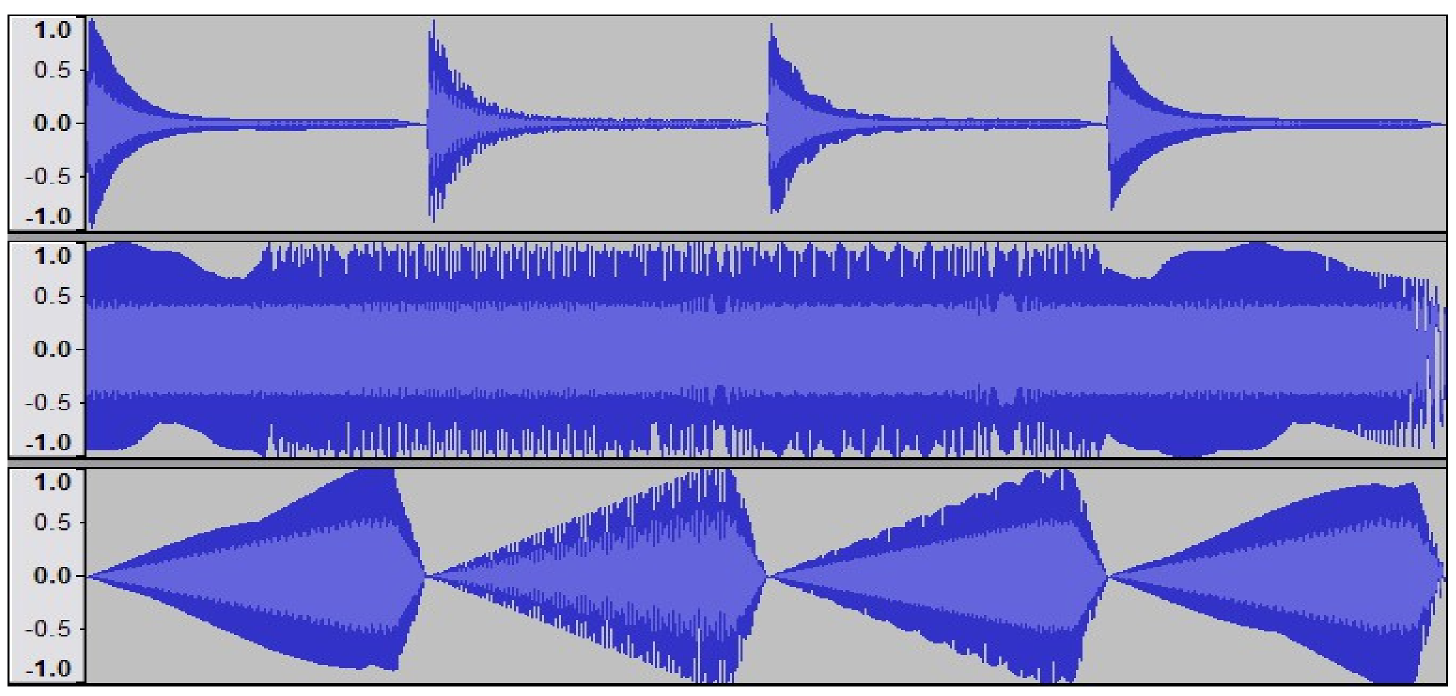

Fig 2: the signals in fig 1 on a longer time scale

## Development

**Previous Iterations**

Previous versions were implemented in Python and rely on hardcoding in values and sequencing statements within the program as methods of input. A later version uses a text based UI to allow external control through a terminal. These versions compose frequency, envelope, and waveform functions to generate audio. Later Python versions also have notation functions, capable of generating other functions from a string of musical notation.

**Current Version**

The current version was built from the ground up in C++. The basic method of synthesis is the same, but trades a relatively small amount of memory use for a drastic speed increase. Rather than using a collection of functions which must be stitched together in the body of the program, this version uses a collection of synthesizer element objects, instances of which can be created, altered and connected using a graphic interface.

These objects all inherit from an abstract class Element; Each Element has one output and some number of inputs as well as some number of constant parameters. Elements can be linked together by setting the inputs Elements to the outputs of other Elements. A few basic Elements are: Oscillator, Constant, and Mix_Sum.

By creating and linking Elements, additive synthesis, FM synthesis, and feedback/cross couple oscillators can all be implemented.

**Oscillator**
Oscillators take amplitude and frequency inputs and output a discrete waveform with those parameters.
**Constant**
Constants take no input and output a single stored parameter value.
**Mix_Sum**
This typer of mixer takes a variable number of inputs and outputs their sum, multiplied by a gain parameter.

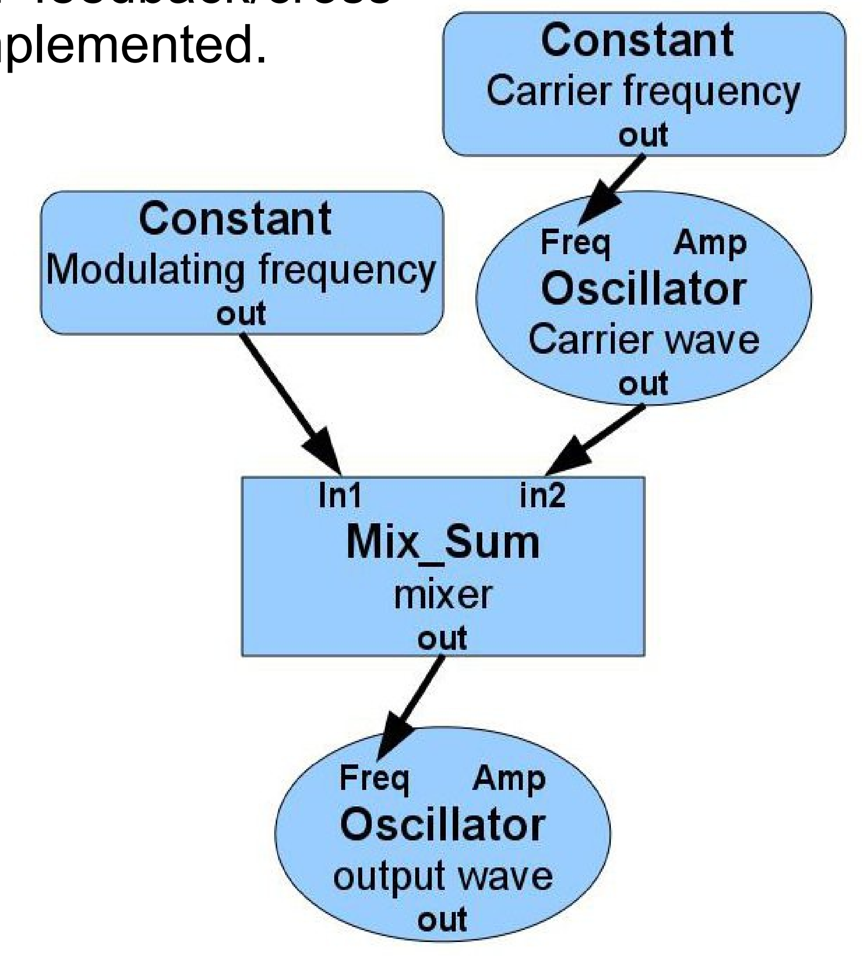

Fig 3: FM implemented with Elements

## Testing

Testing has been primarily by ear. This has been sufficient to confirm that the correct audio is being produced. Speed testing was difficult to implement in Python versions without impacting performance; testing C++ versus Python versions has not been attempted rigorously, however, the newest version appears significantly faster and produces audio as expected.

## Results and Conclusions

Those Elements implemented work properly, though the GUI cannot effectively control them yet. The core modular synthesizer, however, is capable of AM, FM, additive synthesis, and feedback.

The goal was to produce a creatively useful piece of software. At present, versions of the synthesizer are usable by the author, and can produce music in a range of timbres.