Developing an Audio Synthesizer
TJHSST Senior Research Project
Computer Systems Lab 2009-2010

Victor Shepardson

January 22, 2010

Abstract

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

Keywords: additive synthesis, FM synthesis, digital oscillator

1 Introduction

Electronic sound synthesis has been of interest to musicians, electrical engineers and computer scientists for as long as it has been practical. Since the 1970s, synthesizers have evolved from primitive analog machines to sophisticated computer programs. Today, methods such as additive synthesis utilizing Fourier transforms, sampling, granular synthesis, physical modeling, and FM synthesis can be implemented or emulated using software. 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.

2 Background

Three methods are particularly relevant to this project: additive synthesis, FM synthesis, and synthesis by cross-coupled oscillators.

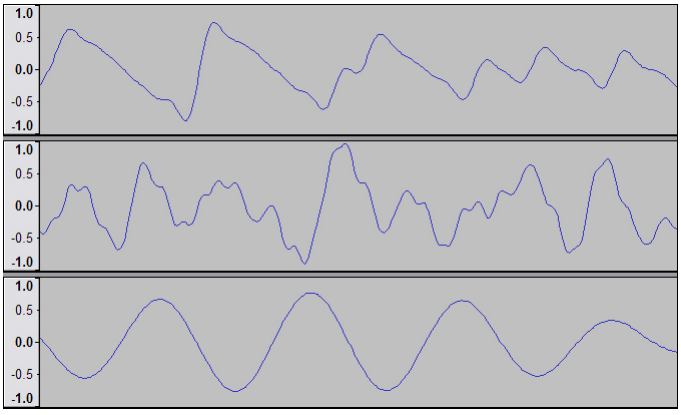

Figure 1: Waveforms produced by additive synthesis

2.1 Additive Synthesis

All periodic functions can be decomposed into sine waves; an audio signal which behaves periodically over long enough time domains, therefore, can be represented by a collection of sine waves with different phase, frequency and amplitude called a spectrum (Moore). Additive synthesis exploits this fact to create audio signals by summing together sine waves or by using Fourier Transforms to turn spectra in audio signals. One implementation of additive synthesis–the one used in this project–is to use multiple digital oscillators to generate harmonic tones, and to sum their outputs.

2.2 FM Synthesis

Frequency Modulation (FM) synthesis can produce a rich spectrum from just one tone by using it to modulate the the frequency of a second oscillator. This is the same method used to for radio transmission (where the carrier frequency is very high) and vibrato effects (where the modulating frequency is very low. 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 sideband frequencies. By varying the harmonic relationship between the modulating frequency and carrier frequency, and the amplitude of the modulating signal, output signals which are spectrally rich and dynamic over time can be produced (Chowning).

2.3 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).

3 Development

The purpose of this project is to produce software of some creative value. The final program should be capable of producing a wide variety of sounds given either musical or non musical input, and should be easy to manipulate for a user familiar with some of the underlying theory.

All versions have been implemented in Python. Early versions relied on hardcoding in values and sequencing statements within the program as methods of input. A current version uses a text based UI to allow external control through a terminal. A final version will consist of a GUI allowing the creation and interconnection of several elements: oscillators, waveforms, amplitude/frequency functions, mixing blocks, and note matrices. Oscillators repeat waveforms periodically given frequency inputs, and then apply amplitude inputs. Mixing blocks can take two or more input signals and a control signal and output some function of those signals, for example, an average of two inputs weighted by the control signal, or one input scaled by a control signal and added to a second input. Note matrices are representations of musical notation which can be used to generate functions. Using a simple notation developed for this project, a user can input musical information; a note matrix can then produce piecewise frequency functions corresponding to the specified pitches, and piecewise amplitude functions corresponding to information about envelope shape and note durations.

Any number of oscillators can be created, allowing additive synthesis by the addition of pure tones, or implicitly by the use of different waveforms. Sub-audio band frequency and amplitude functions can create vibrato and tremolo effects; note matrices allow notes to be played legato or staccato, or to swell in or fade out. Envelope shapes are visible in Figure 2. By linking the outputs of oscillators to the frequency or amplitude inputs of other oscillators FM and AM can be implemented; Figure 3 shows an implementation of FM synthesis for constant carrier and modulating frequencies.

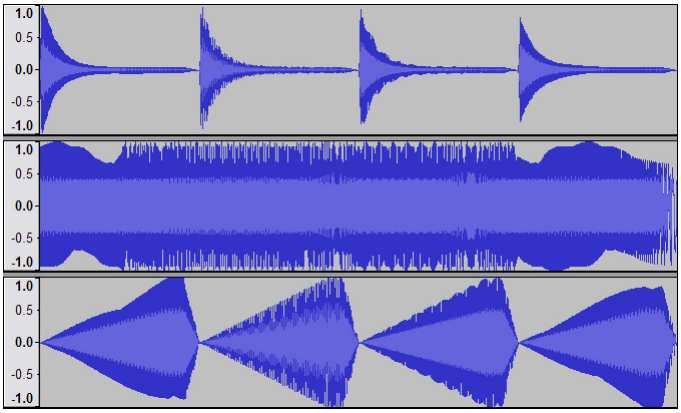

Figure 2: Audio output over several seconds

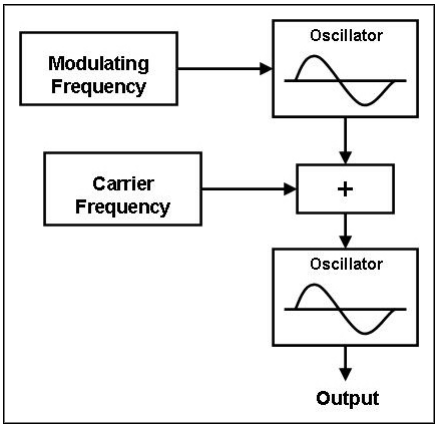

Figure 3: Simple FM synthesis

Once a system of these interconnected elements has been constructed, the user can choose to "run" it, which will loop over a specified time duration and write the output of a specified mixing block to a WAV file. The program computes using floating point values close to 0; the final audio signal is normalized to a maximum amplitude of 1, then converted to 32-bit signed integers. Audio is represented using Pulse Code Modulation (PCM) and is uncompressed. The file can then be played back by an separate media player or audio processing program.

4 Testing

Testing has been primarily by ear. This has been sufficient to confirm that the correct audio is being produced. To a smaller extent, visual and spectral analysis of output has been done using Audacity. When debugging file write, Okteta was used to examine file headers. Time testing has been done using Python's time module to compare complexity of input and operations to runtime; however, the insertion of timing statements into loops has a significant impact on performance, making it difficult to determine the relative time consumption of different processes. It appears, however, that oscillators and note matrices account for similar proportions of the total runtime, and that normalization and file write make a smaller but significant contribution. Overall, the program runs in about half of the audio duration for a single oscillator and a note matrix generating frequency and amplitude functions. Currently, sound is not produced in real time but is written to a file.

5 Results

The features described have been implemented successfully, with the exception of GUI, mixing blocks, and feedback from oscillator outputs to other elements; additive synthesis and note matrices are operational, but FM and cross coupled oscillators are not.

6 Conclusion

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

References

[1] Chowning, J., "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation". *Journal of the Audio Engineering Society 21(7)*, pp. 526-534, 1973.

[2] Miranda, E. R., "At the Crossroads of Evolutionary Computation and Music: Self-Programming Synthesizers, Swarm Orchestras and the Origins of Melody", *Evolutionary Computation 12(2)* pp. 137-158, 2004.

[3] Moore, R., *Elements of Computer Music*, Prentice Hall, Englewood Cliffs, NJ, 1990.

[4] Pachet, F., "Description-Based Design of Melodies", *Computer Music Journal 33(4)*, pp. 56-68, 2009.

[5] Valsamakis, N. and Miranda, E. R., "Iterative sound synthesis by means of cross-coupled digital oscillators", *Digital Creativity 16(2)*, pp. 79-92, 2005.

[6] Valsamakis, N. and Miranda, E. R., "Extended waveform segment synthesis, a nonstandard synthesis model for microsound composition", *Proceedings of Sound and Music Computing 05, Salerno (Italy)*, 2005.