Developing a Music Sequencer/Synthesizer TJHSST Senior Research Project Computer Systems Lab 2009-2010

Victor Shepardson

October 27, 2009

Abstract

A non real-time sequencer/synthesizer is being developed in Python, capable of producing audio signals made up of superposed periodic functions. Smooth pitch transition, vibrato, tremolo, envelope shaping, and polyphony have been implemented. The program performs as intended so far.

Keywords: additive synthesis, sound synthesis, music sequencing

1 Introduction

As the terms are used here, a synthesizer electronically produces sound; a sequencer allows the specification of melodic and harmonic information to a synthesizer. Together, they can allow the production of music without physical proficiency in an instrument, and the creation of unique sounds unique to the synthesizer as an instrument. The goal of this project is to produce something of creative value; the intent is not to imitate existing instruments, but to produce something which is a versatile and intersting instrument in itself.

2 Background

Currently the only language in use is Python. Knowledge of Python was preexisting, as was basic knowledge of the digital representation of sound and of synthesis by superposition. To date, areas of study have been sound file formats, scientific pitch notation, and methods of synthesis. Articles discussing granular synthesis and synthesis by cross-coupled oscillators were examined; similar methods may be implemented at some point. An interface to communicate with a composition program called MuseScore might also be developed.

3 Development

At the core of the synthesizer is an oscillator. A waveform is specified as a Python function with domain [0.0,1.0); as a time variable loops over the total audio duration incremented according to the sample rate, a parameter p is incremented such that it ranges from 0.0 to 1.0 over the duration of 1 cycle according to a frequency function of time. In this way, a discontinuous frequency function will not result in a discontinuous audio signal, preventing undesirable pops and clicks. The output from this system of frequency function and oscillator is then multiplied by an envelope function of time with range [0,1]. The resulting value is stored as a decimal floating point value. After the main loop terminates, the stored audio signal output is normalized, converted to binary integers, and printed to an output file.

Features will be/are being/have been implemented as follows, and revisited or expanded as necessary:

- core of synthesizer: oscillator, loop over time, amplitude frequency and wave weighting functions; enables smooth pitch change, vibrato and tremolo effects.
- file write: converting floating points to binary integers, writing uncompressed wav file header
- core of sequencer: note class, the note matrix, build-from-matrix amplitude and frequency functions
- interface: implementation of musical notation, user input
- optimization: revisiting algorithms to improve speed
- additional features: varied attack, GUI/integration with MuseScore, post-synthesis processing, computer composition, conversion to real-time synthesis

4 Testing

Testing is primarily by ear; files are played to detect inconsitencies or affirm behavior. The spectral analysis feature in Audacity has been employed to examine noise levels. Okteta has been used to examine file headers. Speed testing will be done using the time module in Python.

5 Results

The synthesizer is producing wav files with the intended content; smooth pitch transitions and properly shaped envelopes are evident. Polyphony is functioning but may still exhibit issues related to phase of voices in unison. Noise is not excessive and seems to be properly linked to sample depth of output files. Currently, speed lags behind real-time, and is probably nowhere close to optimal.

6 Conclusion

The employed methods of sound synthesis are capable of producing a wide range of subjectively interesting tones.

References

[1] 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.

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