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

Victor Shepardson

October 26, 2009

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.

2 Background

Currently the only language in use is Python; speed might necessitate the use of C. An interface to communicate with a composition program called MuseScore might also be developed. Knowledge of Python was preexisting, as was basic knowledge of sound 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.

3 Goal

The goal of this project is to produce something of creative value; the intent is not imitate existing instruments, but to produce a versatile and intersting instrument in itself. To be considered sucessful, a finished program will enable smooth pitch change, harmonic variability, and dynamic control as well as sequencing and input using some kind of standard notation. Expansions to the project, time allowing, may include GUI, computer music composition, or additional audio processing functions.

4 Procedure

Features are being implemented as follows, and revisited or expanded as necessary:

- core of synthesizer: oscillator, loop over time, amplitude frequency and wave weighting functions
- 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

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.

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.