UNIVERSITY OF CALIFORNIA

SANTA CRUZ

MAKING MUSIC IN TIME: A REAL TIME INTERACTIVE SYSTEM FOR MUSICAL COMPOSITION AND PERFORMANCE

A Thesis submitted in partial satisfaction of the requirements for the degree of

MASTER OF ART

In

MUSIC

By

Philip Lamperski

June 2010

The Thesis of Philip Lamperski is approved

_____
Professor David Evan Jones, Chair

_____
Lecturer Peter Elsea

_____
Professor Paul Nauert

_____
Tyrus Miller
Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

ABSTRACT

MAKING MUSIC IN TIME:
A REAL TIME INTERACTIVE SYSTEM
FOR MUSICAL COMPOSITION AND PERFORMANCE

By

PHILIP LAMPERSKI

This thesis presents an interactive system designed and programmed by the author to enable a performer to interact in real time with resynthesized and processed versions of the input sounds he/she produces. Specifically, it looks at how the author uses a modified electric-acoustic guitar with the interactive system to compose and perform music in real time. Technical details of both hardware and software are explained. Several musical applications of the program are described including composition of a studio piece, and live real time performance. The thesis focuses upon the ways in which the interactive system motivates and rewards performative musicianship.

Introduction

This thesis addresses and responds to research done by Miller Puckette in his paper "Patch For Guitar". Specifically, Puckette observes that:

> *Much more work needs to be done in finding intelligent ways to vary the processing parameters as a function of instrumental phrasing…*
>
> *Visits to a club or concert hall reveal that the computer is mostly used as a recording and or sequencing device, rather than as a musical instrument…*
>
> *The main payoff of computing in music performance has been to reduce and replace the role of musicianship.* [1]

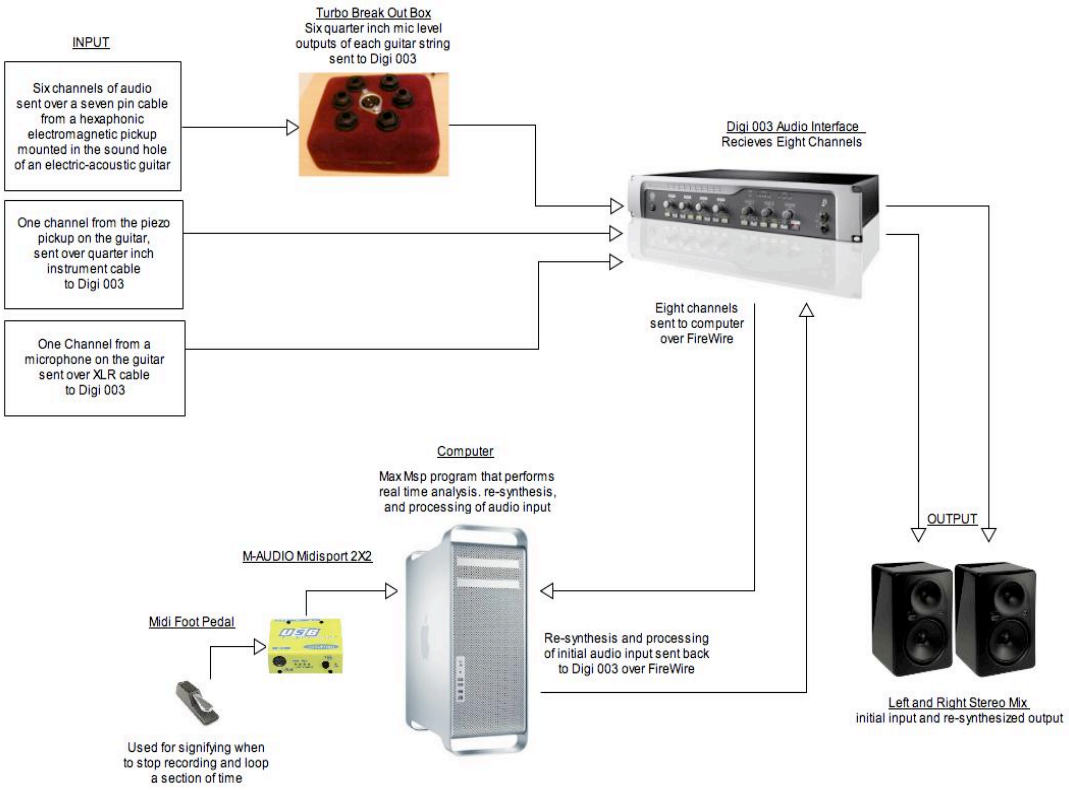
This thesis presents a real time compositional and performance interface designed and programmed by the author to intelligently vary granular processing parameters as a function of instrumental onset times. The interface also enables live instrumental performance passages to be recorded, sequenced, analyzed, and re-synthesized in real time. The various synthesis and sequencing modes motivate and reward musicianship by responding dynamically to the performer's careful and incisive interactions with the program. This interface was designed to be used for composition and live performance with multiple instruments and/or for single instruments with multiple outputs. For example, the author used this interface with a modified hexaphonic electric-acoustic guitar (i.e. one channel of audio per string) for composition and performance. The body of this paper will describe in detail how the hardware and software components of the interface work, and how they interact intelligently with the performer's real time instrumental input.

[1] Puckette, Miller. "Patch For Guitar". Presented at the second Pd convention. Montreal. 2007

Overview

A performer/composer initiates the use of this interface by producing instrumental sounds that are recorded by the computer. The computer reprocesses the input sound in real time in combination with the original sounds produced by the performer. The performer can then interact with the program by layering and mixing additional instrumental sounds, and/or by varying processing parameters via the program's graphical user interface. The nature of these interactions will be explained by an overview of the hardware setup followed by a summary of the user interface, software and performance techniques.

Figure 1: Hardware Overview



In figure 1 the hardware connections are shown in the form of a signal flow chart. Highly accurate multiple signal analysis is made possible through the use of a hexaphonic electro-magnetic pickup mounted in the sound hole of an electric-acoustic guitar. The pickup mount is adjustable, which allows the player to finely tune how close the coils are to the strings. Zebra strings were used to enhance the pickup's response and acoustical quality of the sound.[2] Each string on the guitar has it's own channel of audio sent over a seven pin cable to a breakout box. The breakout box then redistributes the incoming six channels to six 1/4-inch mic level outputs. These six outputs, in addition to one channel from the internal piezo pickup on the guitar, are sent to seven of the eight inputs on the Digi 003 audio interface for analog to digital conversion, which are then sent over firewire into the computer. A microphone is used on the remaining eighth input channel to achieve an even richer sound. The initial signal inputs from the guitar and the re-synthesized copies are mixed down in the computer to left and right stereo outputs. [3]

Figure 2: Guitar Overview

Hexaphonic pickup:

Outputs:

*Battery Supply is to the left of the outputs

Piezo1/4 inch output

7 Pin output

[2] Zebra strings wind phosphor-bronze plated steel wire side-by-side with 8% nickel-plated steel wire.

[3] A more portable and cheaper setup is possible by using a 13-pin cable from the hexaphonic guitar that goes to a Stringport analog to digital conversion box, which is then sent over USB to a laptop. However, one would have to modify the 7-pin jack on this guitar to be a 13-pin jack.

Figure 3: Software GUI Overview [Programmed in Max Msp]

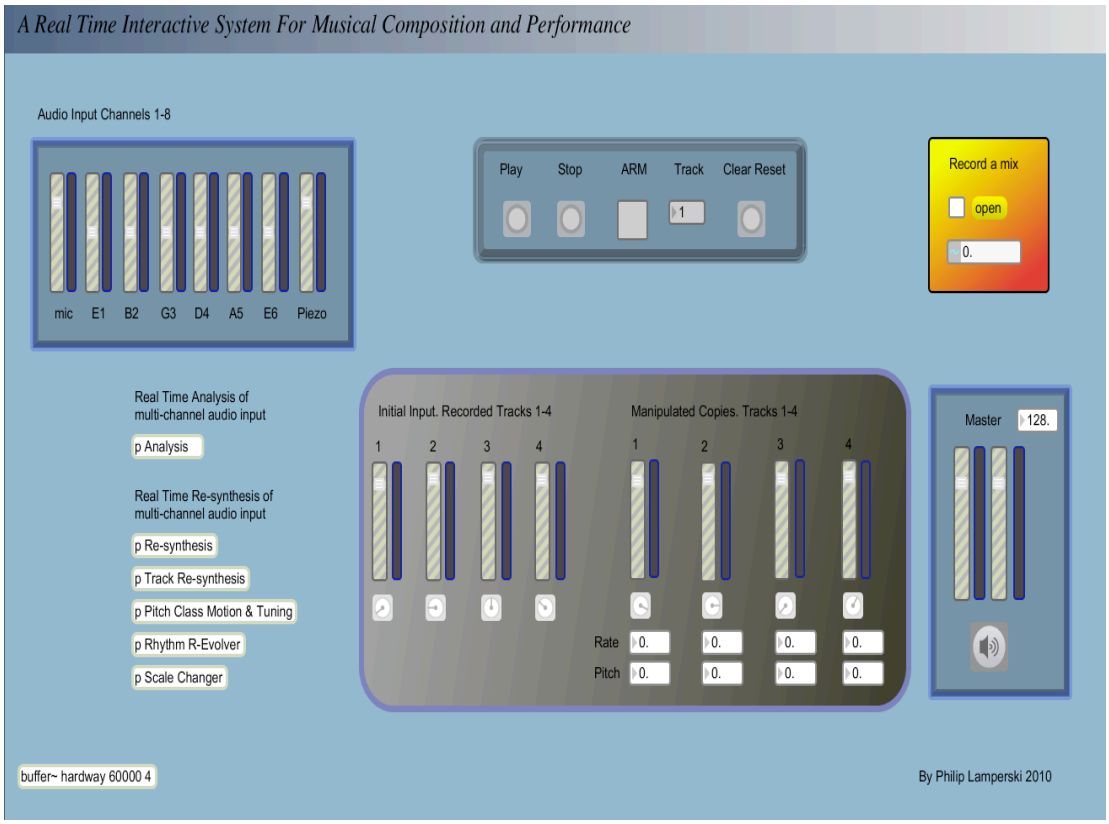
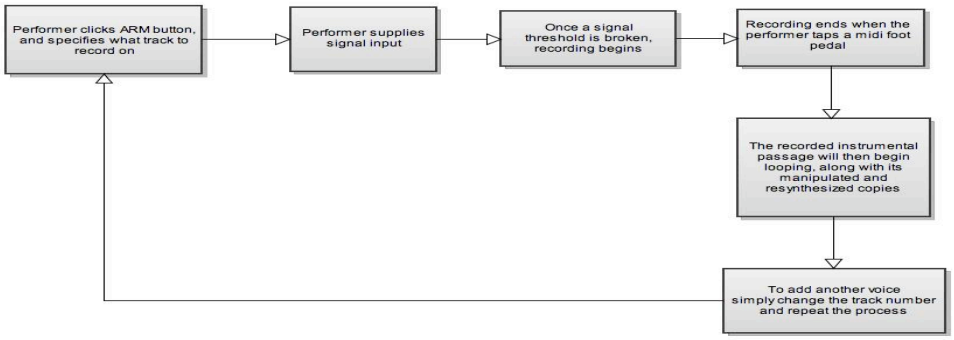


Figure 4: How To Record An Instrumental Passage

" If playing a musical instrument and/or singing were pure drudgery this would be all to the good, but it seems likely that much of musical knowledge is built up through the physical act of making music in time."[4]



[4] Puckette, Miller. "Patch For Guitar". Presented at the second Pd convention. Montreal. 2007

Figures 3 and 4 illustrate how the performer interacts with the graphical user interface to record an instrumental passage in real time. The performer can record and layer up to four tracks in real time. The initial four tracks of recorded instrumental passages loop continuously without any change, and can be silenced via the track's slider. All sounds originate with the instrument and the coordination of instrumental passages is determined by split-second performative timing.

The rate and pitch number boxes underneath the manipulated copies of the four tracks allow the user to time stretch without changing pitch, and pitch (frequency) shift without changing playback speed. The rate and pitch number boxes display decimal numbers that represent ratios. The signal can be reversed (played backwards) if the rate number box is a negative number. These fairly simple granular processing controls on pitch and tempo modulation can be controlled algorithmically as well as manually. These make it possible for the user to build complicated polyphonic textures that phase and evolve.

The gain sliders control amplitude levels, and the dials control panning. The clear/reset button clears all recorded sounds in the buffer. One can record a left and right stereo mix by clicking open in the "record a mix" window, providing a file name, and then clicking the toggle once to initialize record, and once again to end the recording. The user is also involved as composer who controls the layering, mixing, and sequence of the composition. The real-time nature of these interactions motivates and rewards musicianship.

And even though there is clearly much music to be made using studio techniques and/or sequencers, there is also much to be made instrumentally.[5]

In order for a computer to vary processing controls intelligently according to instrumental phrasing or onset time, highly accurate analysis is required. Pitch, amplitude, spectral frequency and amplitude, and onset time are analyzed in real time by this program. Six instances of Miller Puckette's object *fiddle* were used for analysis of each guitar string's onset time within a buffer. [6]

Figure 5: Analysis Of Each Guitar String

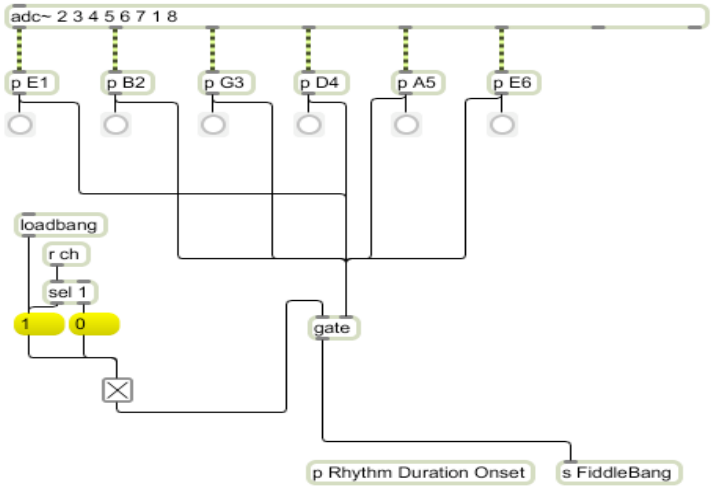


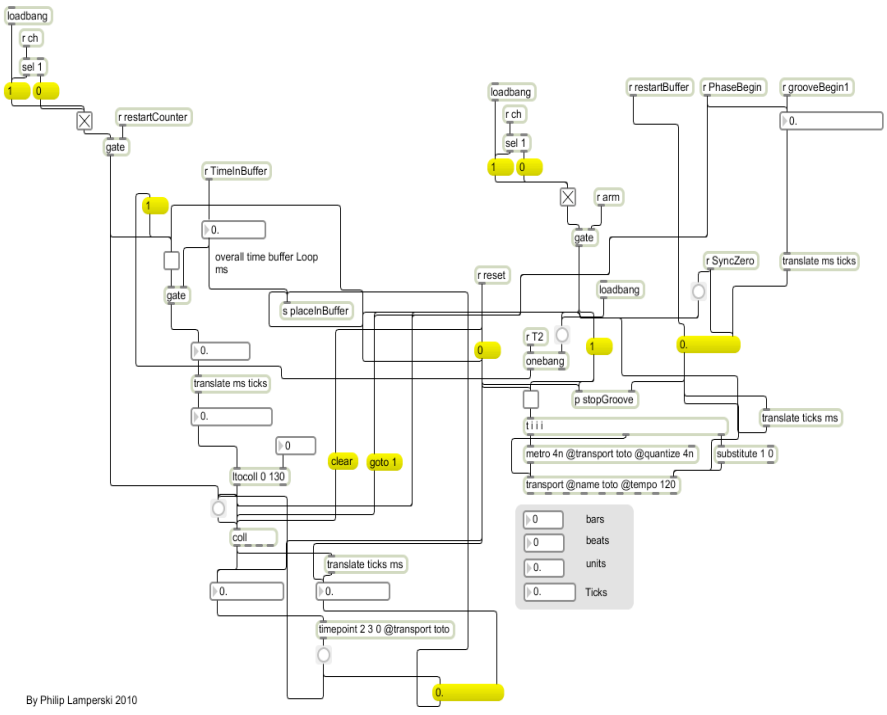
Figure 5 shows the signal path from each guitar string going into a separate patch where pitch, amplitude, spectral frequency and amplitude, and onset time are analyzed. The adjustable hexaphonic pickup, along with *fiddle's* amplitude sensitivity controls allow for clear analysis, without any noticeable cross talk. When the strings are excited while recording an instrumental passage, the onset times are stored and

[5] Puckette, Miller. "Patch For Guitar". Presented at the second Pd convention. Montreal. 2007

[6] Puckette, Miller. Lippe, Cort & Apel, Ted. "Fiddle~ Max/Msp Object For Pitch Following and Sinusoidal Decomposition". Version .05 for Mac Max/Msp 5

undergo various conversions and translations. Specifically, the onset times are originally captured as a sample number from a sample counter object within a specified buffer. The sample number is then converted to milliseconds. The milliseconds are then translated into ticks and stored in a collection object. The purpose of these time conversions and translations is to provide flexibility for interface with other objects. For example, the time-point object only accepts ticks as a time value. In fact, the algorithm in figure 6 uses nearly every available time value syntax in order accurately track the performer in real time.[7]

Figure 6: Tracking And Storing Onset Times

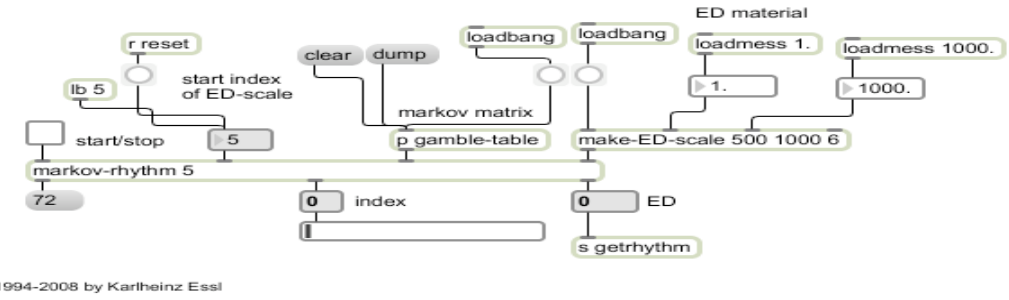


[7] Notice in figure 6 that the metro object's components (tempo, bars, beats, units, and ticks) are used in conjunction with the time-point object for accurate tracking and storage of the performer's instrumental attacks.

The algorithm in figure 6 tracks and stores instrumental onset times performed by the user. At each of the performer's onset times, a pool of durations tied to a markov matrix and a pool of pitch classes, both predetermined by the user, are used to process the performer's audio signal granularly in real time.[8] Specifically, at each onset, the algorithm immediately concatenates a sequence of processed segments of the initiating sound. Each of these segments is processed starting from the onset (t = 0) of the initiating sound, and each segment has a duration and pitch dependent on the selection pools provided by the user.

Two patches from Karlheinz Essl's real time composition library were modified and employed to generate the durations and pitch classes, supplied by the user, that drive the granular processing parameters.[9] Figure 7 shows the duration generator that creates the duration pool.

Figure 7: Duration Generator: The User Can Specify What Durations Are Used



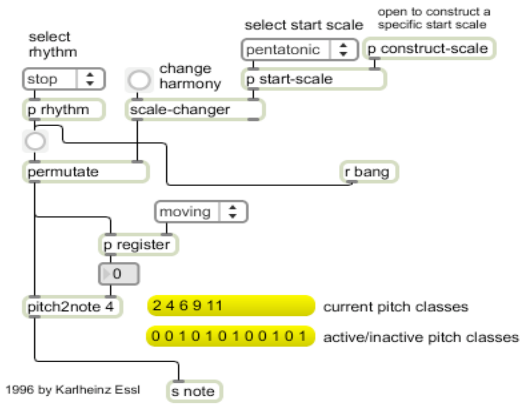
[8] The purpose of storing onset times is to continuously trigger the markov sequence of processing at each onset time in the recorded instrumental passage as it is looped through.

[9] Essl, Karlheinz. "Real Time Composition Library – RTC-lib". Software library for algorithmic composition – open source. 1992 – 2010.

To access this window click on “Rhythm R-evolver” object in the GUI. The two number boxes above the object labeled “ Make-ED-Scale 500 1000 6” can be changed by the user to generate a new pool of durations on a logarithmic scale. The number box on the left is for the shortest duration, and the number box on the right is for the longest duration. The durations are specified in milliseconds.

The concatenated segments are also pitch shifted according to pitch classes provided by the user. For example, in figure 8, five pitch classes: 2, 4, 6, 9, and 11 provided by the user, are used in various registers chosen according to Brownian motion. [10] The pitch generator also allows the user to choose from traditional scales (i.e. pentatonic, diatonic, etc). To access this window click on the “scale changer” object in the GUI.

Figure 8: Pitch Generator: The User Can Specify What Pitch Material Is Used



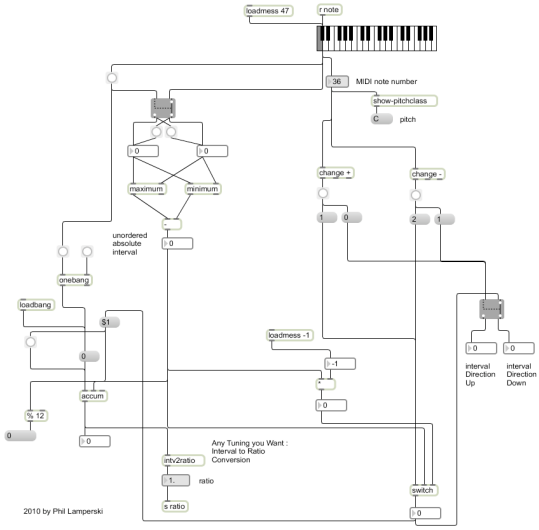
The user is able to specify what pitch classes and durations are used to improvise and harmonize with the performer in real time. As mentioned previously,

[10] The five pitch classes are taken from the barred seventh fret of a guitar in standard tuning

the user can also control time stretching, and signal reversal from the GUI without changing pitch. Adjusting all of these parameters manually and algorithmically allows for delicate and subtle differences in instrumental phrasing to produce drastically different output from an initiating sound.

The sequence of selected pitches is then sent to an algorithm that analyzes pitch class interval motion and assigns a tuning to those intervals. In figure 9 interval motion is represented by an integer, and then converted into a ratio represented by a decimal number. The user can supply any tuning formula they wish. The interval to ratio conversion provides the granular processing parameters with the necessary information for pitch shifting and time stretching.[11]

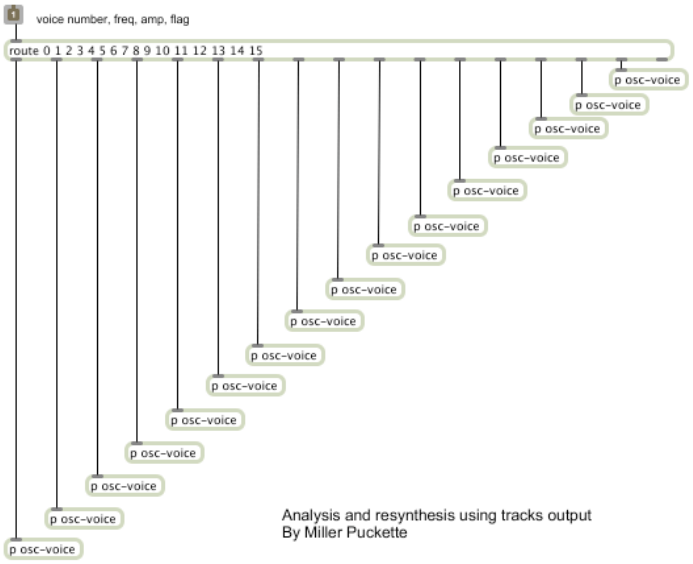
Figure 9: Pitch Class Motion and Tuning



[11] This algorithm is currently used to tune and analyze the interval motion of the pitch generator. However, it could be used to analyze the performer's interval motion, and develop an adaptive tuning system.

The final section of this paper will briefly address real time resynthesis of the performer's input. Given that analysis objects such as *fiddle*, or *sigmund* allow for the extraction of pitch, amplitude, onset, spectral frequency and amplitude, it is possible to resynthesize an audio signal using multiple oscillators. More specifically, Miller Puckette's *sigmund* object provides an example of routing individual spectral peaks (frequency and amplitude) in the incoming signal to separate sine wave oscillators.[12]

Figure 10: Routing Spectral Peaks To Oscillators (Real-Time Additive Synthesis)



The Sigmund resynthesis technique is used in the program on each string of the guitar as well as on each recorded track that loops. Users have the ability to separately adjust the gain level of resynthesis for both recorded tracks and guitar strings by double clicking on the "resynthesis" objects on the left hand side of the GUI.

[12] Puckette, Miller. Lippe, Cort & Apel, Ted. "Sigmund~ Sinusoidal Analysis and Pitch Tracking". Version .05 for Mac Max/Msp 5.

Figure 11: Resynthesis Of Each Guitar String

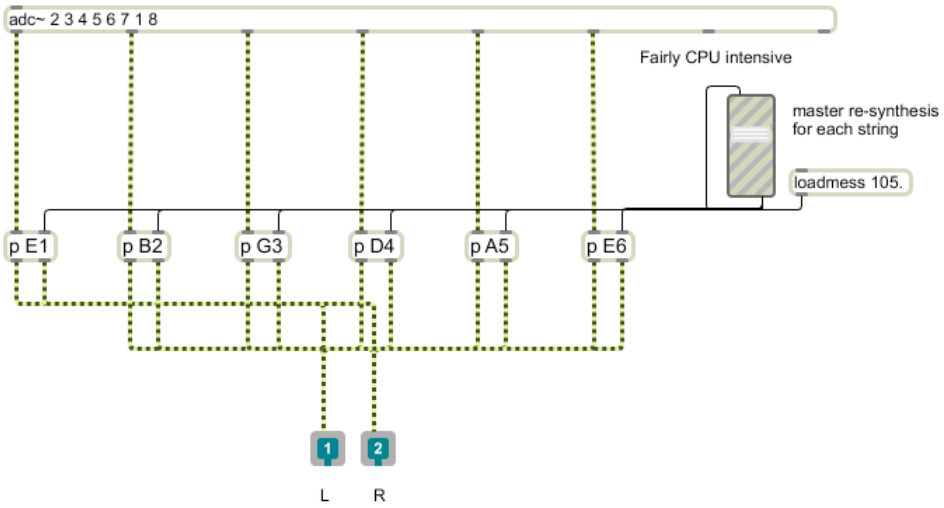
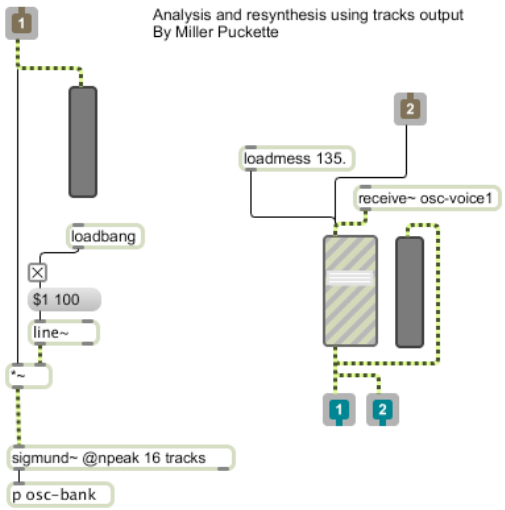


Figure 11 illustrates each string of the guitar going to a separate patch where sigmund analysis and real-time additive resynthesis takes place. The gain slider on the right allows the user to adjust the mix level. The mix of granular processing, additive resynthesis, and initial input creates thick and organic sounding textures with which the performer interacts.

Figure 12: Inside one of the Guitar String Resynthesis Patches



Conclusion And Future Work

This thesis has presented a real time compositional and performance interface that intelligently varies granular processing parameters as a function of instrumental onset times. The interface also enables live instrumental performance passages to be recorded, sequenced, analyzed, and resynthesized in real time. Though this interface does present some initial responses to Puckette's observations about the current state live instrumental performance with computers, it is primarily a foundation off which to build. Since the user has such accurate signal analysis on multiple dimensions (time onset, pitch, amplitude, and spectral components) and has the ability to store this information in memory, it is possible to create an extremely intelligent program that uses this information to create a compositional palette of relationships. The tuning ratios could become adaptive, or the resynthesis of spectral peaks could evolve over time as the performer plays. This performance interface could be integrated with the OMAX improvisation environment, and or combined with Max Score notation editor and Antescofo for composition and score following. The author will continue to develop the current program with the goal of creating a more powerful and flexible interface for musicians to use for performance, composition, and improvisation.

Bibliography

Assayag G., Bloch G., Chemillier M., Cont A., Dubnov S. “Omax Brothers: a Dynamic Topology of Agents for Improvization Learning”. Workshop on Audio and Music Computing for Multimedia, ACM Multimedia 2006, Santa Barbara, 2006

Cont, Arshia. “ANTESCOFO: Anticipatory Synchronization and Control of Interactive Parameters in Computer Music", Proceedings of International Computer Music Conference (ICMC), August 2008, Belfast, Ireland.

Essl, Karlheinz. “Real Time Composition Library – RTC-lib”. Software library for algorithmic composition – open source. 1992 – 2010. http://www.essl.at/software.html

Puckette, Miller. “Patch For Guitar”. Presented at the second Pd convention. Montreal. 2007. http://crca.ucsd.edu/~msp/Publications/pd07-reprint.pdf

Puckette, Miller. and Apel, Ted. 1998. "Real-time audio analysis tools for Pd and MSP". Proceedings, International Computer Music Conference. San Francisco: International Computer Music Association, pp. 109-112.

Puckette, Miller. Lippe, Cort & Apel, Ted. “Sigmund~ Sinusoidal Analysis and Pitch Tracking”. Version .05 for Mac Max/Msp 5. http://crca.ucsd.edu/~tapel/software.html

Puckette, Miller. Lippe, Cort & Apel, Ted. “Fiddle~ Max/Msp Object For Pitch Following and Sinusoidal Decomposition”. Version .05 for Mac Max/Msp 5. http://crca.ucsd.edu/~tapel/software.html