Wavetable Modular Synthesizer

Task 1: Direct Sample Level DSP Routine for Sound Synthesis/Audio Effect

Written Report

Candidate Number: 19329

Advanced Computer Music: G6003 (UG)

University of Sussex

November 2013

Wavetable Modular Synthesizer

1.0 Overview

To achieve this task I implemented wavetable synthesis through look-up table waveform calculations and sample based wavetable creations, within a modular environment. Inspired by such programs as Max/MSP[1] and Hadron[2] I aimed to carry out these types of wavetable synthesis with modulation synthesis, achieved by creating a graphical environment to patch outputs of modules to inputs of others all within SuperCollider[3].

Three main aims were set; first to take a new approach to wavetable creation that differs from a drawing input or straight forward pre-calculated table values. Second to implement sample-based wavetable synthesis and third to create a modular system that allows these different synthesis techniques to interact with each other i.e. an environment that allows for ring, amplitude and frequency modulation implementation.

2.0 Wavetable synthesis

Critical to this project was to carry out a Direct Sample Level (DSP) routine. Meaning manipulation of amplitude and phase[4] values without the aid of higher level synthesis manipulation tools such as SuperCollider Ugens[5]. To realise this, wavetable synthesis was chosen, specifically wavetable look-up.

Wavetable look-up involves reading back stored amplitude and phase values from a table/ array. These values determine the level and shape of the waveform and can be attained multiple ways, which include abstracting samples from a sound file, or using iterative mathematical equations to retrieve and store calculated values in arrays. I chose the second method within this program because once calculations have been made, we can alter the waveform in real-time.

"One method is to let the machine calculate only one cycle of the wave and store samples in its memory" (Miranda 2002) [6]. For my wavetable design I followed Miranda's proposed method, and imposed a law on every equation that the calculated frequency will be 1Hz or one cycle (See MathsEquations class source code). This means when their playback rate is increased, the value of playback rate is equal to cycles per second. E.g. 440 times the rate = 440Hz/440 cycles per second of the waveform period.

---

[1] Max/MSP is a modular visual programming software http://cycling74.com/products/max/ (accessed 04/11/2013)

[2] Hadron is a SuperCollider based graphical patching environment http://www.earslap.com/projectslab/hadron. (accessed 04/11/2013)

[3] SuperCollider is a real-time audio synthesis programing environment, written by James McCartney http://www.audiosynth.com/icmc96paper.html, http://supercollider.sourceforge.net/. (accessed 04/11/2013)

[4] Phase/Phase degree = position of a point on a waveform cycle in time. E.g. the starting phase degree of a wave cycle is 0° the end 360° and the middle 180°.

[5] SuperCollider Ugens (unit generators) are the basic building blocks of synths on the server, used to generate or process audio. http://danielnouri.org/docs/SuperColliderHelp/UGens/UGen.html (accessed 04/11/2013)

[6] Miranda, E. "Computer Sound Design: Synthesis techniques and programming", Elsevier Ltd. p44

The mathematical equations I chose to use, range from simple sine wave calculations (with altering phase degree onsets) to more complex waveforms like Square and Sawtooth waves. To create sawtooth waveforms we use Fourier Synthesis/additive synthesis[7]. "A sawtooth waveform is the sum of all harmonics, each with an amplitude of 1/n times the amplitude of the fundamental (n = number of harmonic) and with all even-numbered harmonics shifted 180° out of phase"[8]. The harmonics effect the timbre of the sound.[9]

Example: Sawtooth Wave Formula[10]

$$y(t) = \sum_{k=1}^{N} \frac{(-1)^{k+1}}{K} \sin 2\pi kt$$

Noise was also generated, (using random values) however since the wave created is just 1Hz, we end up repeating the same sample values over and over instead of constantly cycling through continuous randomness. Although a weakness of the one cycle design, it does lead to more rhythmic results in the sound, which is a welcome side effect.

"We iteratively read out the precomputed sample values from the table in real time to generate the sound"[11]. Reading of samples is handled by the PlayBuf Ugen, which uses a pointer to read through the array index of float values at a specified rate. The float values correspond to amplitude values and the index numbers correspond to phase values. "The index corresponds to the instantaneous phase of the oscillator". [12]

2.1 A different approach to wavetable synthesis

A problem I found when generating wavetables, was that I could only make a waveform of one particular equation at a time. Therefore I proposed an algorithm which allows the user to define the mathematics for sections of the waveform. These 'divisions' allow us to split the waveform up into proportionate equal sections, e.g. if divisions = 2: split waveform into halves, = 3: split waveform into thirds. Allowing for a different type of wave per section. It must be noted that amplitude and phase values for every waveform are maintained regardless of number of divisions.

---

[7] "Additive synthesis involves the construction of complex spectra by adding various sine tones together, tones that may or may not be harmonically related, depending on the spectra desired": Naumann & Wagoner, Analog Electronic Music Technologies In Tape, Electronic, and Voltage-Controlled Synthesizer Studios, Collier Macmillan Publishers, p116

[8] Ibid, p112

[9] "Timbre is determined by the harmonic content of the signal": Steven W. Smith, The Scientist and Engineer's Guide to Digital Signal Processing http://www.dspguide.com/ch22/2.htm, (accessed 04/11/2013)

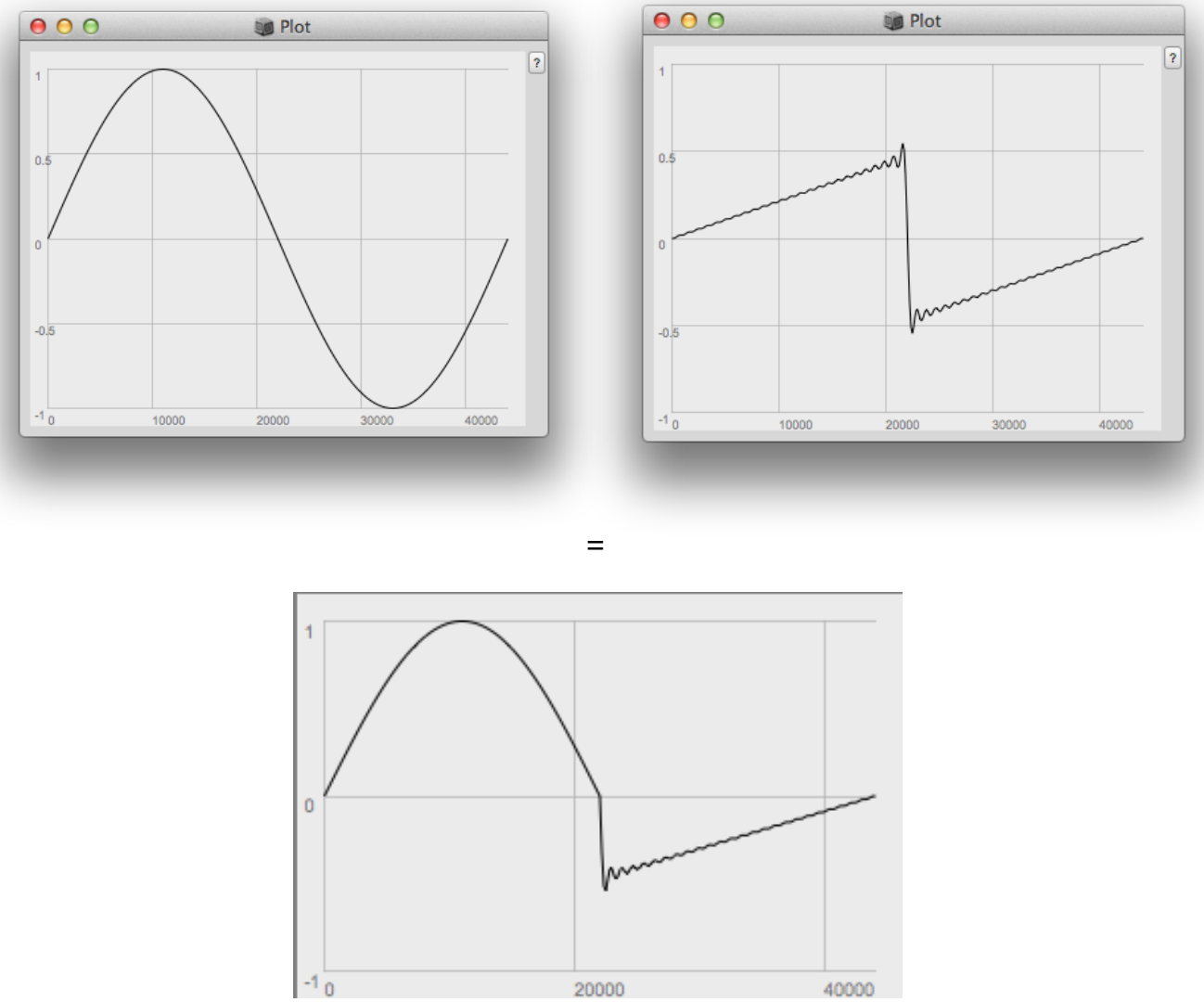
[10] Loy, G. "Musimathics: The mathematical foundations of music volume: 2", MIT Press, p376

[11] Ibid p378

[12] Ibid p379

To illustrate let us suppose we have two divisions:

Division1: Sine Wave equation (left)
Division2: Sawtooth equation (right)
The outcome (bottom)

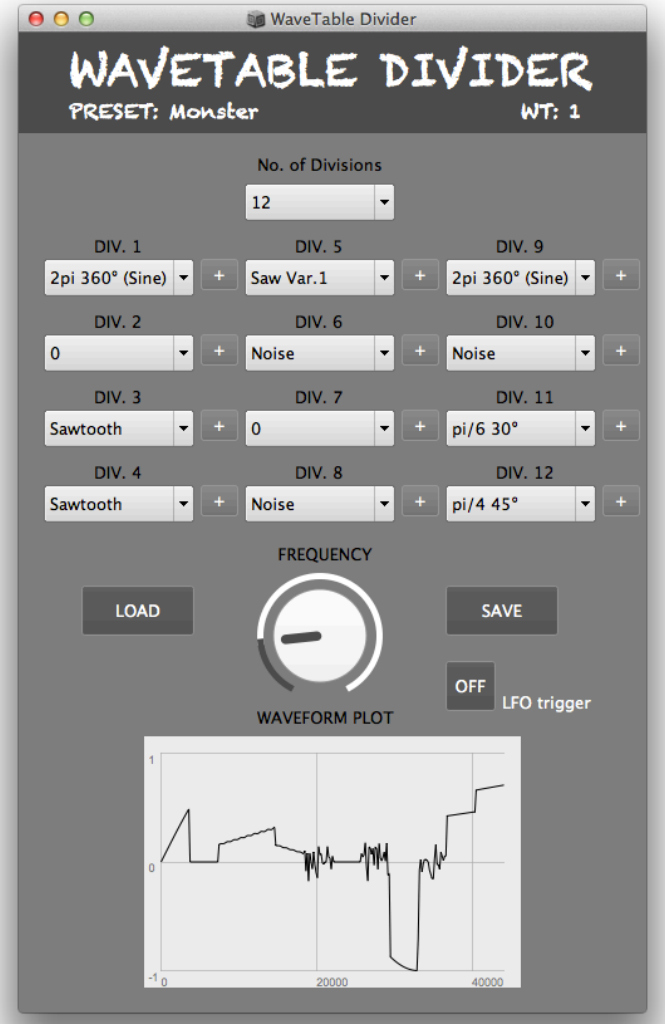


(Fig. 1.1)

Figure 1.1 shows that when divisions = two, the wavetable created is split into two equal halves (half = half the number of total samples). The first half maintains the sine wave amplitude and phase values of division1 (from 0 - half no. samples) and the second half maintains the sawtooth values of division2 (from half no. samples - total no. samples). Thus creating a new waveform as the outcome above shows. The number of divisions (n) is used to divide the number of total samples (ts).

The above illustration can be proved by the user, clicking the plus buttons next to division math menus to show one full cycle of each individual mathematics and witnessing the result of multiple divisions by viewing the plot at the bottom of the Wavetable Divider window.

Example of 12 divisions with Wavetable Divider:



(Fig. 1.2 [13])

2.2 Encountering Problems

As stated previously, we use a pointer to read through the array at a specified rate. At a rate of one we are cycling through every sample value, "if we set the rate = 2, we skip every other sample, running through twice as fast, and the frequency doubles"[14]. A problem arises when we change the playback rate to become a float e.g. 2.2, since we do not have an index value at 2.2, it is just empty.
"Since the next value we need to output will not fall on an original sample position but between the samples, various schemes have been introduced to cope"(Collins 2010)[15].

[13] Wavetable Divider demonstrating 12 divisions. Notice that you can even count the number of divisions by analyzing the different shape of the waveforms for each split section.
[14] Loy, G. "Musimathics: The mathematical foundations of music volume: 2", MIT Press, p379
[15] Collins, N. "Introduction to Computer Music", Wiley & Sons p69

These ‘schemes’ Collins discusses are as follows:
1. "Truncation, choose the nearest sampled value toward 0”.
2. "Rounding, we can improve on truncation by performing rounding on the index before selecting the sample".
3. "Linear Interpolation another approach is to assume a straight line joins adjacent sample points and to select a point on this line that is proportional to the fractional part of the index".[16]

The PlayBuf Ugen uses linear interpolation, and I let it handle this task for me. The formula for linear interpolation is as follows:

$$x_2 = ((y_2 - y_1)(x_3 - x_1) / (y_3 - y_1)) + x_1$$
$$y_2 = ((x_2 - x_1)(y_3 - y_1) / (x_3 - x_1)) + y_1$$ [17]

A note on interpolation:

The approach of dividing a single waveform cycle into different mathematical functions could be further applied to interpolation. Currently interpolation mathematic curves include linear, exponential, cubic and many more. Taking this division approach proposed for wavetable creation above, we could potentially divide interpolation curve lines into an arbitrary number of sections. Therefore the overall curve can be part linear part cubic part exponential and on and on. I think the possibilities for these types of custom curves to exist will lead to very interesting sonic results, especially when applied to interpolation curves of envelopes for high level control and for granular level control.

Theory Example of divided interpolation:

Let us suppose we want to have a interpolation curve that is a combination of linear and polynomial interpolation. We take the length of the interpolation curve and divide it into halves.

New Interpolation curve =

0 - halfwaypoint (Linear) = $x_2 = ((y_2 - y_1)(x_3 - x_1) / (y_3 - y_1)) + x_1$
$y_2 = ((x_2 - x_1)(y_3 - y_1) / (x_3 - x_1)) + y_1$

halfwaypoint - end (Polynomial) =
$f(x) = -0.0001521x^6 - 0.003130x^5 + 0.07321x^4 - 0.3577x^3 + 0.2255x^2 + 0.9038x.$ [18]

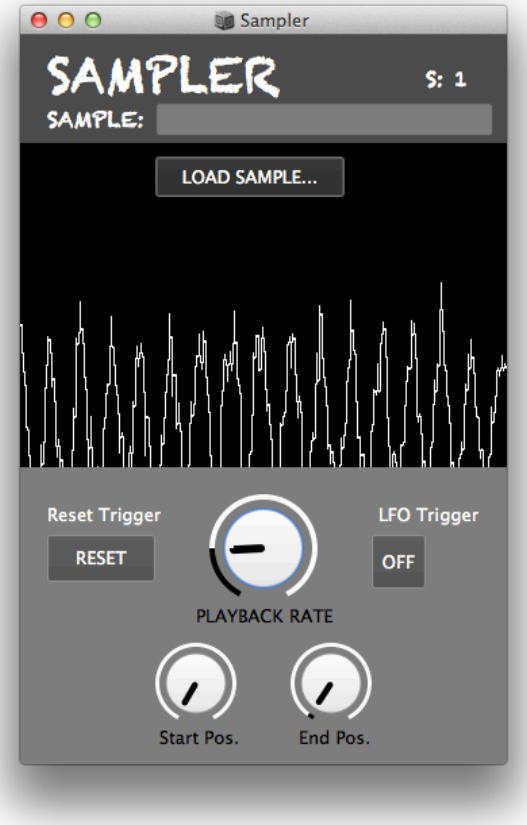
This proposal of an interpolation function that can be a mixture of interpolation mathematics is definitely a point for further exploration.

---

[16] Loy, G. "Musimathics: The mathematical foundations of music volume: 2", MIT Press, p380
[17] Linear Interpolation Equation http://ncalculators.com/geometry/linear-interpolation-calculator.htm (accessed 04/11/2013)
[18] Polynomial Interpolation Equation http://en.wikipedia.org/wiki/Interpolation (accessed 04/11/2013)

3.0 Sample-Based Synthesis

The Sampler synth module gives users the ability to create periodic waveforms/wavetables in real-time by adjusting the start and end position of a sound file and looping the section continuously. Thus playback rate changes the frequency of the newly shaped waveform.



(Fig. 1.3 [19])

"One advantage of sampling is that longer wavetables allow for the use of pointers within a sample, in order to define internal loopings"(Miranda 2002)[20].

So far all wavetables created (with wavetable divider) have been enforced to be 1Hz, as Miranda, 2002 states with sampling we can create longer wavetables and this is exactly what the Sampler synth succeeds in doing. The start/end positions control the length in time of the waveform and the playback rate controls the pitch.

A key difference between the frequency rates of the Sampler Synth and the Wavetable divider is the frequency scale mapping ranges from -200 to +200 for the sampler as opposed to +10 - +8000 for the wavetable divider. When the rate is in minus figures the playback pointer instead of skipping forward in the index array, skips backwards. Therefore resulting in reversing the waveform.

---

[19] Sampler Synth, showing a periodic waveform, structured from a piano sample with the end position altered to give a very small window to loop the sound. Therefore creating a waveform from a sample, increasing the rate will increase the pitch just as with a calculated waveform from a look-up table.

[20] Miranda, E. "Computer Sound Design: Synthesis techniques and programming", Elsevier Ltd. p46

4.0 Modulation Synthesis

"Modulation in signal processing refers to the control of an aspect of one signal by another. The controlled (modulated) signal is the carrier, and the controlling signal the modulator".[21] Three types of modulation have been implemented within this design. Ring Modulation (RM), Amplitude Modulation (AM) and Frequency Modulation (FM). All three are variants of using a modulator signal to control different parameters of a carrier signal. To carry out these modulations, I assigned Bus outputs to each synth module, which allow Buses to act as modulator or carrier signals (see ModularSynthesizer class).

"A straight multiplication of two signals is called ring modulation (RM)"[22]
Ring Modulation Formula:

$$R(t) = C(t) * M(t)$$

Input1 of each module is used for ring modulation. Meaning any Bus output that is plugged into a modules input1, will execute the above formula. E.g. when Bus16 output is plugged into Bus18 input1, the output of Bus18 = Bus16 * Bus18. (Input2 = AM, Input3 = FM).

An issue that arises from ring modulation is a drop in amplitude level. This is a result of amplitude values that are below 1 multiplying together. For example if modulator signal has amplitude values 0.1, 0.5, 0.7 and carrier signal has 0.5, 0.4, 0.6 the result of multiplying the signals gives us a new signal with amplitude values 0.05, 0.2, 0.42. Quieter than the original signals. To combat this decrease, I included the ability to increase amplitudes of modules accessible from the 'modular playground' view. The potential for such a decrease in amplitude is available because ring modulation allows for bipolarity of signals[23]. In ring modulation, "with sinusoids the spectrum ends up with two frequencies (two side-bands)"[24]

Amplitude modulation is a very similar modulation process to ring, with the difference of the modulator being unipolar[25]. With amplitude modulation, the "spectrum ends up with the side-band frequencies in ring modulation, as well as the original carrier frequency"[26]

$$A(t) = C(t) * M(t) + C(t)$$

Frequency modulation was the third modulation type used.

"In FM, the instantaneous frequency of a carrier wave is varied according to a modulating wave, such that the rate at which the carrier varies is the frequency of the modulating wave, or modulating frequency. The amount the carrier varies around its average is proportional to the amplitude of the modulating wave"[27].

---

[21] Collins, N. "Introduction to Computer Music", Wiley & Sons p123
[22] Ibid, p123
[23] Bipolar: "Signals whose amplitudes can take both positive and negative values" Ibid p123
[24] "Side bands refer to (C-M and C+M) where C is carrier and M the modulator" Ibid p123
[25] Unipolar: The modulator is always positive (amplitude levels are scaled between 0-1).
[26] Collins, N. "Introduction to Computer Music", Wiley & Sons p124
[27] Chowning J. The Synthesis of Complex Audio Spectra by Means of Frequency Modulation. J. Audio Eng. Soc. 21, 7, 1973, https://ccrma.stanford.edu/sites/default/files/user/jc/fmsynthesispaperfinal_1.pdf (accessed 04/11/2013)

Carrier frequency C, modulation frequency M, and the modulation depth/frequency deviation D:

$$y(n) = \sin(2\pi(n / R)C + (D * \sin(2\pi(n / R)M))$$ [28]

Within my modular system, I took this standard FM equation structure and altered it, so that instead of addition we use multiplication. The non-standard formula used therefore becomes:

$$y(n) = \sin(2\pi(n / R)C * (D * \sin(2\pi(n / R)M)))$$

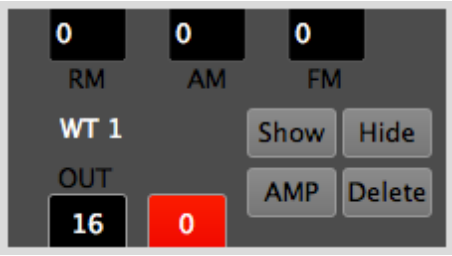
The result, is a siren like sound rather than vibrato that is associated with standard FM implementation. This variation plugs the whole signal of the modulator and multiplies the frequency of the carrier signal, the amplitude of the modulator's amplitude level still causes deviation of carrier signal frequency but a different sonic output is achieved.

LFO Note:

Every module created, has the ability to become a low frequency oscillator (LFO) (with LFO trigger button). An LFO is "an oscillator that processes a low frequency that works anything below 20Hz, which is the lowest threshold of hearing for humans"[29]. Triggering this within the synths, changes the playback rate of the sample array to lie between 0-20Hz for the wavetable divider and -20-20Hz for the sampler synth. The ability of making synth modules into LFO's enables different control over modulation synthesis, than when a module is cycling within audio rate >20Hz and is a feature commonly found within past analog modular synthesizers such as the Moog Modular Synthesizers[30].

5.0 Evaluation/Outcomes

The final result of the implemented synthesis methods above has resulted in a well formatted GUI design to easily interact with modules, enabling the user to connect inputs and outputs with simple text input (figure 1.4), which proofed a suitable alternative than the line connection drawings found in existing modular software such as Max/MSP.



(Fig. 1.4)

[28] Collins, N. "Introduction to Computer Music", Wiley & Sons p166

[29] d'Escrivan J. "Cambridge Introduction to Music technology", Cambridge University Press, p82

[30] Moog Modular synthesizers http://www.vintagesynth.com/moog/modular.php (accessed 04/11/2013)

A strength of the architectural foundation code, is that it allows for additional modules to be added very simply for further development. Additions such as envelopes, filters and effects units can be written outside the core ModularSynthesizer class and easily implemented inside it just as the Wavetable Divider and Sampler synth have been, using the defined bus incrementation system to take care of new modules.

An interesting path for future development, would be to see the Wavetable Divider implement loading of sound samples (e.g. taking the amplitude and phase values of one period of a sample waveform) and using it as a new maths function. Thus allowing for a combination of mathematical equations and sampled sound wavetables to define the divisions of the 1Hz waveform as seen in the current Wavetable Divider. Following the subject of divisions, it should be made possible for the division locations to be moved into arbitrary positions by the user, instead of having divisions set by proportionate ratios of halves, thirds etc.

It would be easily bettered, with the potential of multiple inputs. For example, each module can take an unlimited number of Bus inputs into each modulation input provided. Increasing the sonic possibilities, furthering the systems ability to reach deep sonic output. The ability to save and load the state of the modular environment would be a good aspect to add too. The Wavetable Divider makes suitable use of loading and saving of sound creations, which users benefit from and without this function in the overall environment, it can be overwhelming to start from scratch every time it is launched, especially if a favored output has been created. MIDI support was initially implemented but I found it a distraction in creating soundscapes, since only having output when the keyboard is triggered became a hindrance compared to continuous output from the modules. Most of the time you want to keep a continuous output of sound that is manipulated over time, without the annoyance of having to press a midi key to test the sonic palette.

Band limiting the generated signals would be another key aspect to improve the system. The waveforms created using additive synthesis (sawtooth waves and square waves) were made with a limit of fifty harmonics. Since the maximum frequency of the wavetable divider is 8000Hz and each harmonic is a multiple of the fundamental frequency we pass half the sampling rate on the third harmonic (of max freq) reaching 24000Hz. “Any signal with energy above the Nyquist rate (half sampling rate), will always create aliasing”[31](Loy 2007). As Loy states, aliasing occurs, which when you experiment with the synth can be heard. Therefore an anti-aliasing filter could be implemented to stop the harmonics from passing the Nyquist rate.

The overall sonic output possible from this system is vast and diverse. Having the ability of synthesizing sound, sampling sound and the combination of the two with the functionality of modular routing allows for deep sonic possibilities. Harmonic and rhythmic content is achievable as demonstrated in the provided sound examples (see below) and with the further improvements discussed above this wavetable modular synthesizer has potential to become a very powerful sample level synthesis tool.

Sound Examples:
https://soundcloud.com/jamesnapierstuart
(Number of sound examples, to demonstrate the range of sonic output possible)

[31] Loy, G. "Musimathics: The mathematical foundations of music volume: 2", MIT Press, p19

References

Max/MSP visual programming software available at http://cycling74.com/products/max/ (accessed 04/11/2013)

Batuhan Bozkurt, Hadron: graphical patching environment available http://www.earslap.com/projectslab/hadron. (accessed 04/11/2013)

SuperCollider written by James McCartney available at http://www.audiosynth.com/icmc96paper.html, http://supercollider.sourceforge.net/. (accessed 04/11/2013)

SuperCollider Ugens (unit generators) available at http://danielnouri.org/docs/SuperColliderHelp/UGens/UGen.html (accessed 04/11/2013)

Eduardo Reck Miranda (2002). *"Computer Sound Design: Synthesis techniques and programming (second edition)"*. Elsevier Ltd..

Joel Naumann and James D. Wagoner (1985). *"Analog Electronic Music Techniques, In Tape, Electronic and Voltage-Controlled Synthesizer Studios"*. Collier Macmillan Publishers.

Steven W. Smith (2002). *"The Scientist and Engineer's Guide to Digital Signal Processing"*. available at http://www.dspguide.com/ch22/2.htm (accessed 04/11/2013)

Gareth Loy (2007). *"Musimathics: The mathematical foundations of music volume: 2"*. MIT Press.

Nick Collins (2010). *"Introduction to Computer Music"*. Wiley & Sons.

Linear Interpolation Equation available at http://ncalculators.com/geometry/linear-interpolation-calculator.htm (accessed 04/11/2013)

Polynomial Interpolation Equation available at http://en.wikipedia.org/wiki/Interpolation (accessed 04/11/2013)

Chowning J. The Synthesis of Complex Audio Spectra by Means of Frequency Modulation. J. Audio Eng. Soc. 21, 7, 1973, available at https://ccrma.stanford.edu/sites/default/files/user/jc/fmsynthesispaperfinal_1.pdf (accessed 04/11/2013)

Julio d'Escrivan (2012). *"Cambridge Introduction to Music technology"*. Cambridge University Press.

Moog Modular synthesizers available at http://www.vintagesynth.com/moog/modular.php (accessed 04/11/2013)