Ian Grant

SS Music Technology

Final Year Project

Acceleration Canons

Mr. Richard Duckworth

Due Date: 27/04/2012

The aim of this project was to generate a set of software tools to assist the composition and performance of acceleration canons using open source software. The ability to create abstractions and their own easily accessible help files made Pure Data the ideal choice for creating the software tools. Csound is also utilised to offer audio synthesis options to those who prefer to work with Csound.

This report will give a brief introduction to acceleration canons with an overview of Conlon Nancarrow's contribution to the mode of composition. Clifton Callender's equations will be introduced. These serve as the basis for the application of acceleration in the software patches. Then the implementation of the software in Pure Data will be outlined in detail.

Acknowledgements:

This project would not have been possible without the input and support of too many people to list here, but here is an attempt: First and foremost I owe my thanks to my supervisor Jenn Kirby for all of her approachability, motivation and advice. I appreciate that I was never turned away from her door whether my visit was announced or not. Thanks to Richard Duckworth and Keith Hennigan for their invaluable tutelage. Thanks to Trinity College Dublin Music Department for providing the equipment that allowed Richard to introduce me to music programming and allowed me to explore it. Thanks to Dan Trueman who recommended that I read Clifton Callender's journal on accelerando and Justin London's *Hearing in Time.*

I would also like to thank all of the people who have put so much effort into helping people like me understand things like acceleration canons and music programming and making it entirely free for me to access. This includes, but is not limited to, Clifton Callender, Johannes Kreidler, Miller Puckette and Mike Moser-Booth

Finally I would like to thank my family, friends and my flatmates Fionnán Howard and Killian McCarthy who have shown infinite understanding and patience and for Fionnán's generous help with learning maths again.

Table of Contents

Acceleration Canons……..5

Conlon Nancarrow…….7

Clifton Callender……..11

Implementation…….13

- Acceleration Equations……13
- Predetermined Canon Tools…….14
- Real-time Canon Tools…….16
- Signal Generators…….17
- Effects/Dynamics…….18
- Mixing……..19
- Other…….20

Demo Canon and Conclusion……...20

Acceleration Canons

Most musicians are familiar with the concept of canons of displacement, where a melody is performed and one or more imitations are played alongside the first melody after a given duration. This imitation can be performed at any rhythmic distance but the difference between the two voices remains constant. If the first note of the *dux* (leading voice) occurs one crotchet before the first note of the *comes* (following voice) then the last note of the *dux* will occur one crotchet before the first note of the *comes*. Figure 1 depicts the first thirteen bars of Johann Pachelbel's infamous *Canon in D.*

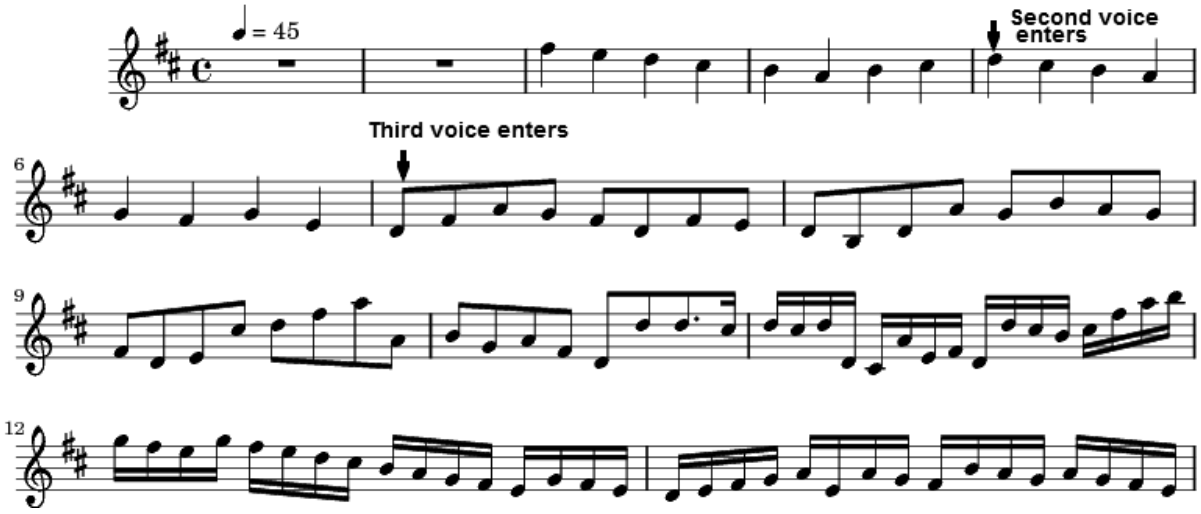

Figure 1

Not as many musicians are acquainted with tempo canons which consist of a melody imitated at a different tempo, a sort of polymetric imitation called tempo cannon. In this case the due to the different tempi the voice that is leading (presenting the melodic material first) can change over time. The terms *dux* and *comes* no longer have relevance. In some cases these tempi have small integer ratio relationships in other cases the tempi can have almost no relation to each other. The former are referred to as mensuration or prolation canons and can be more easily notated and performed. Figure 2 shows the first three bars of Kyle Gann's *Tempo Canon* which shows the ratio of inter-onset intervals between the harp, piano and

harpsichord to be 3:4:5. This means that the tempo relationship equals 20:15:12. At the tempo indicated on the score this conveniently places the harp at 20bpm, the piano at 15bpm and the harpsichord at 12bpm.

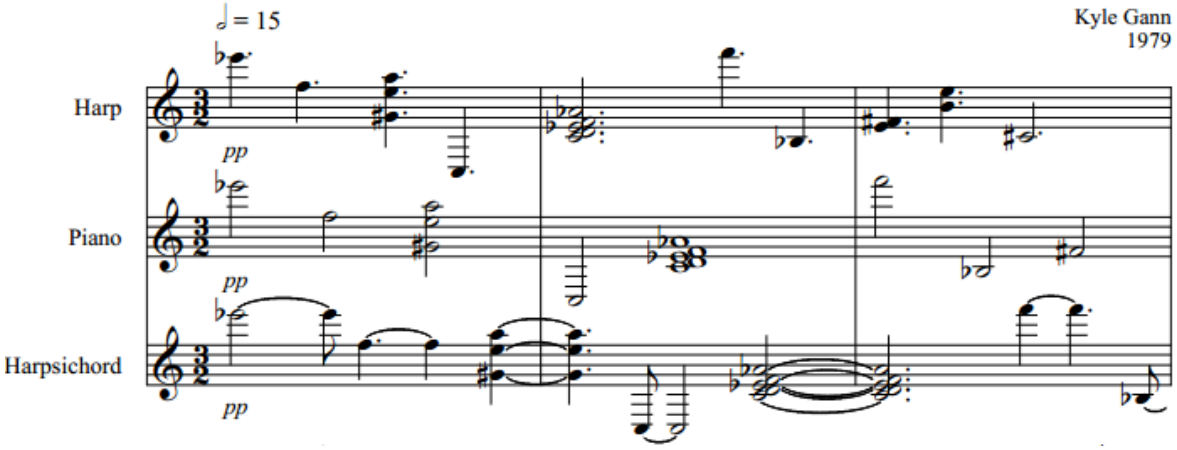

Figure 2[1]

Acceleration canons are more complex even than tempo canons. These are formed by performing an imitation or imitations of a melody at different rates of accelerando and ritardando. These, along with the more complex tempo canons, are much more difficult to represent in traditional music notation though this is generally unnecessary as even if musicians were able to read the music it is very unlikely that they could perform it with an adequate degree of accuracy.

Due to this, the acceleration canon's development relies on music technology. Instead of making acceleration canons accessible to musicians composers aim for their canons to be understood by machines. In the 1940s, before computer music was a viable option for controlling acceleration, Conlon Nancarrow, after being inspired by Henry Cowell's *New Musical Resources*, began composing tempo and acceleration canons for player pianos.[2]

[1] http://www.kylegann.com/TempoCanon.pdf
[2] Roger Reynolds, 'Conlon Nancarrow: Interviews in Mexico City and San Francisco' in *American Music*, Vol. 2, No. 2 (University of Illinois Press, 1984), 3.

Conlon Nancarrow

Nancarrow was born in Texarkana, Arkansas in 1912. He studied at the Cincinnati College Conservatory of Music. In 1937 he fought with the Abraham Lincoln Brigade in Spain. He also spent some time in New York in 1939 where he associated with Elliot Carter and Aaron Copland, two composers who along with Nancarrow have created some of the most rhythmically complex music ever composed.[3]

Nancarrow was refused a passport in the United States of America due to his association with communism from fighting in Spain. Due to this Nancarrow moved to Mexico City in 1940[4] and only returned to the USA once afterwards.

Figure 3 [5]

[3] James R.Greeson and Gretchen B. Gearhart, 'Conlon Nancarrow: An Arkansas Original' in *The Arkansas Historical Quarterly*, Vol. 54, No. 4 (The Arkansas Historical Association, 1995), 457.
[4] *Ibid.*, 458.
[5] http://www.nancarrow.de/arbeitsweise.htm

In 1947 Nancarrow spent a few months in New York in search of a piano roll punching machine. He could not find one for sale but met Jean Lawrence Cook who owned a machine made by Leabarjan shown in figure 3. Cook allowed Nancarrow to employ a mechanic to copy the design. Nancarrow's version is depicted in figure 4, he used this to punch his studies 1-20 which, in spite of their complexity, were all written in standard notation.[6]

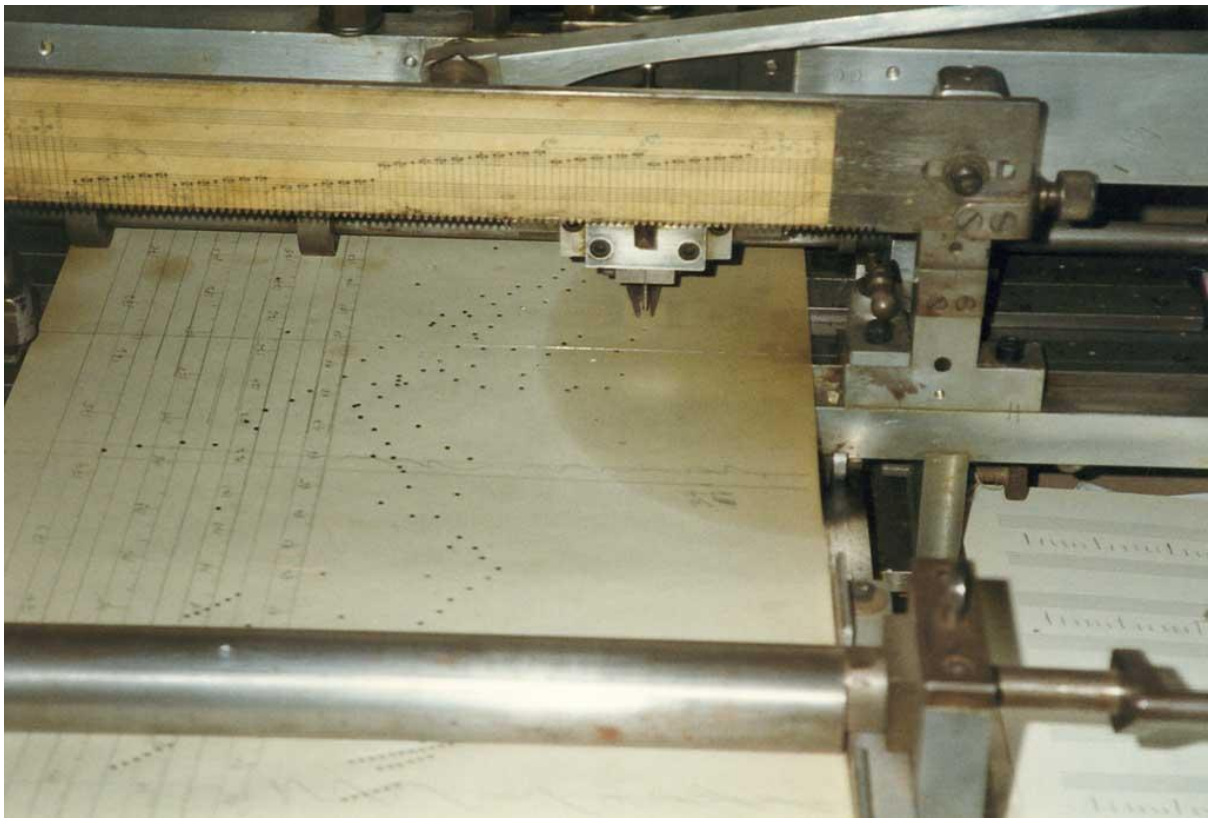

Figure 4[7]

From *Player Piano Study No. 21* on the rhythmic complexity of Nancarrow's canons exceeded the limitations of musical notation and required a scale to be used along the piano roll to measure accurate distances between notes. *Study No. 21* is a very significant work for

[6] *Ibid.*
[7] *Ibid.*

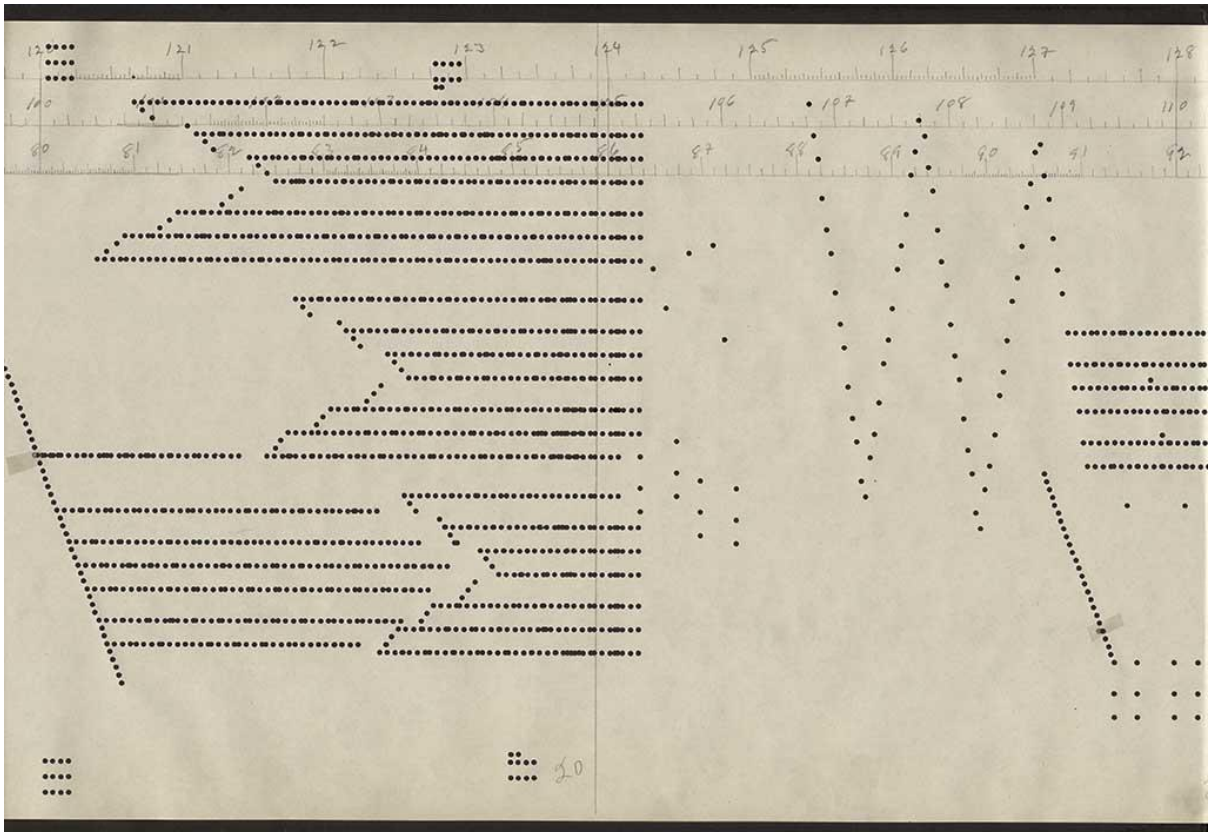

Figure 5

Figure 6

Figure 7

Figure 8[8]

[8] Figures 5–8, http://www.nancarrow.de/arbeitsweise.htm

Nancarrow as it is his first large-scale attempt at a complex acceleration canon, though *Study No. 8* also implemented some acceleration

Figure 5 is an extract from the piano roll for *Study No. 49c* where there are graduated scales printed in pencil along the top. These scales were made by Nancarrow once and reused when necessary. Figures 6 and 7 show the cabinet full of different scales used by Nancarrow labelled as different percentage rates of accelerando and ritardando. Nancarrow's workstation is shown in figure 8, the goal of this project is to reduce that workstation down to a set of convenient and portable software tools. This was made much easier achieve with the resource of Clifton Callender's research.

Clifton Callender

Clifton Callender is an Associate Professor of Composition in the College of Music at Florida State University as well as an Associate Editor for *Perspectives of New Music*.[9] Callender wrote the journal 'Formalized Accelerando: An Extension of Rhythmic Techniques in Nancarrow's Acceleration Canons' in *Perspectives of New Music 39.1* in 2001. In this journal Callender specifies the two different types of acceleration used by Nancarrow in his acceleration studies. The types are defined by Kyle Gann as being arithmetic and geometric.[10]

Arithmetic acceleration is applied when a constant value of time is subtracted or added to the duration of the beat which preceded it. Callender represents this with the equation: $d_n = d_0 - cn$, where d_n is the duration of the *n*th beat where d_0 is the duration of the first beat and *c* is the constant. With this type of acceleration Callender points out that the

[9] http://myweb.fsu.edu/ccallender/
[10] Clifton Callender, 'Formalized Accelerando: An Extension of Rhythmic Techniques in Nancarrow's Acceleration Canons' in *Perspectives of New Music* Vol. 39, No. 1 (Perspectives of New Music, 2001), 189.

rate of acceleration increases as the tempo increases.[11] For example if the arithmetic acceleration formula is applied to a canon with an initial duration of 150 and a constant of acceleration of 1 then the ratio of duration of a beat to the duration of the beat that follows it ($d_n : d_{n+1}$) will be 150:149, 149:148, 148:147, . . . , 4:3, 3:2, 2:1.[12]

Geometric acceleration was employed by Nancarrow in an effort to curtail the issues caused by the changing rate of acceleration incurred by arithmetic acceleration. In this equation the *c* is removed and replaced by *r*. The $r = 1 + {}^{x}/_{100}$ where *x*% is the percentage change in duration from d_n to d_{n+1} (the *x* is negative for deceleration). The geometric equation is: $d_n = d_0 r^n$. This form of acceleration is much smoother though it does have its own upper limits as shown in figure 9 taken from Callender's journal. This illustrates the number of beats per second in a geometric acceleration with an initial duration of one second and an acceleration ratio of 5%.

time span	number of beats
15"-16"	~3.7
16"-17"	~4.6
17"-18"	~5.9
18"-19"	~8.3
19"-20"	~14.2
20"-21"	~503.9
21"-22"	∞

Figure 9[13]

Callender then takes steps towards reaching a generalised view of acceleration; the first step provides a contextualisation of the geometric formula which allows for subdivision of the beat. The formula reads as follows:

[11] *Ibid.*, 190.
[12] *Ibid.*, 190.
[13] *Ibid.*, 191.

$$t_i = \sum_{n=1}^{i} dr^{n-1} = \frac{d(1-r^i)}{1-r}$$

when *i* is the number of the beat and t_i is the time in milliseconds after the onset of the first beat that *i* occurs. This can be used to subdivide the beat when *i* is not an integer.[14] For example the beat halfway between the third and fourth beats can be found by inputting 3.5 for *i*, The second beat of a triplet between the fourth and fifth beats could be found by inputting 4.3333 for *i*, etc.

The journal goes on to outline more equations which create a heavier focus on tempo as opposed to durations. This is something which may be built into the project at a later date but the three equations already outlined here, specifically the geometric ones, serve as the basis for the implementation of acceleration in this project.

Implementation

The patches built in the project are all listed in the Pure Data patch 'reference.pd'. This provides a full list of all the software tools at the user's disposal for creating acceleration canons. In 'reference.pd' the patches are separated into groups determined by their functions. These groups are: Acceleration Equations, Predetermined Canon Tools, Realtime Canon Tools, Signal Generators, Effects/Dynamics, Mixing and Other. For clarity the implementation of these patches will be addressed in this order.

- Acceleration Equations

This section consists of the patches 'arithmetic.pd', 'geometric.pd' and 'generalised.pd' which implement the equations from outlined above from Callender's journal. The patches 'arithmetic.pd' and 'geometric.pd' are controlled similarly to a metro object. Their

[14] *Ibid.,* 192.

parameters can be set as creation arguments and so are made persistent when saved in a patch but can also be updated by control signals in their second and third inlets. The first inlet receives bangs to begin outputting bangs which accelerate or decelerate as programmed. The first inlet can also receive stop messages or be controlled with a toggle just like a metro object.

The left inlet on the ‘generalised.pd’ abstraction behaves differently as it receives floats (floating point number) to be used as i and outputs a float representing t_i. The second and third inlets and creation arguments behave in the same fashion as those on ‘geometric.pd’.

- Predetermined Canon Tools

The ‘grapher.pd’ and ‘graphtoarray.pd’ patches work in conjunction with each other to draw lines using arrays onto a blank manuscript style interface. This acts like a much faster, more compact and more flexible version of Nancarrow’s cabinet of scales. The creation arguments of ‘grapher.pd’ define the initial conditions of two geometric accelerations. The right inlet determines the number of beats to be graphed though the default of 500 is usually ample. The left inlet is the resolution of the graph in tenths of a second and to make this more clear there are timestamps at the beginning and end of each array to indicate the period of time over which these sections of score take place.

The abstraction ‘coincidencecheck.pd’ can be useful for trialling potential creation arguments for ‘grapher.pd’. This abstraction uses the same arguments as ‘grapher.pd’ but the creation arguments can be updated via the inlets. This compares a variable number of beats of two accelerations by compiling lists of variable lengths of the t_is of beats and comparing them using the ‘zl sect’ object in Pd-extended. The common elements are then compiled to a list which can be cycled through using the radio button on the gui which uses the ‘zl nth’ object to cycle through the output. A ‘donecanvasdialog’ message is used to resize the gui to

ensure it is not needlessly large but always large enough to accommodate the radio button which resizes to indicate the number of output values there are to choose from.

The furthest right inlet of ‘coincidencecheck.pd’ allows the user to change the resolution of the comparison. This rounds the output to the nearest multiple of the input and so searches for beats which occur between *x* milliseconds of each other when *x* is the input of the rightmost inlet. Generally a higher value for *x* yields more, less accurate values in output. In the help file the user is reminded of the two millisecond threshold of human hearing for discerning separate onsets of notes and the twenty millisecond threshold of being able to discern the note onset order between two notes.[15] These patches can be seen working together in figure 10. Note the coincident beats at the beginning and just before the end of the section.

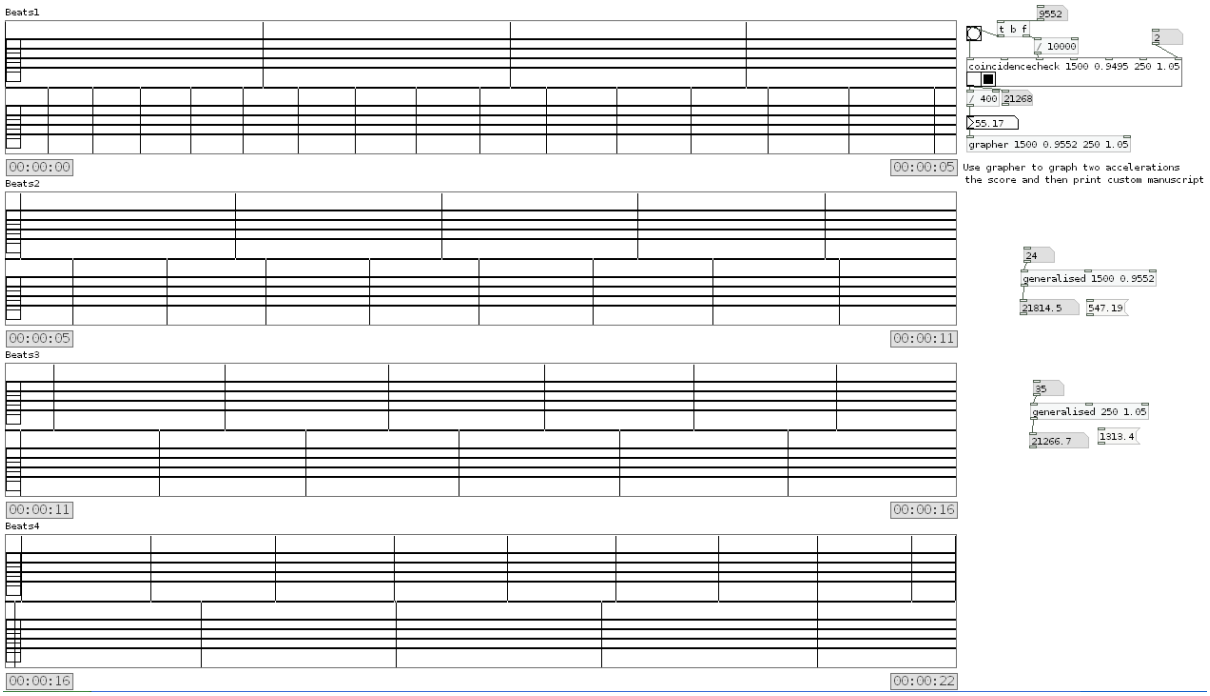

Figure 10

Once the composition is done, a MIDI device can then be used to write a MIDI score to be read by inputting a pulse which could be generated with the ‘geometric.pd’ patch and

[15] Justin London, *Hearing in Time: Psychological Aspects of Musical Meter*, (Oxford: Oxford University Press, 2004), 29.

also using a ‘generalised.pd’ object with a delay object to apply planned subdivisions of the beats.

The MIDI score is written to simply by entering the notes in the order they are to be performed. The score can be saved to the patch it is being used in with the ‘savestate.pd’ object. This can also be used to save the input then edit the message it is saved in to correct any possible errors made in inputting the score with MIDI. This is an admittedly unfriendly interface for correcting errors and will hopefully be improved upon in future versions.

- Real-time Canon Tools

Using the real-time canon patches a user could potentially read and perform a written score of a complex canon as accurately as Nancarrow would have liked. Using the ‘quantizer.pd’ patch the real-time MIDI input is quantised to a pulse in its rightmost inlet. The performer must anticipate the pulse though as only notes inputted prior to the reception of a bang will be quantised. Also since the patch is monophonic there must be one used for each desired voice but this allows for more flexibility in distributing input over multiple changing tempi.

The ‘clocksend.pd’ object can be used to send pulses across a network using TCP packets over netsend. The pulses are received in the left inlet and the destination IP address is accepted in the right inlet in message form. By default this address is ‘localhost’ which sends to the machine that the ‘clocksend.pd’ is on. Creation arguments prevent crosstalk between different ‘clocksend.pd’ and ‘quantizer.pd’ objects. When a ‘quantizer.pd’ receives bangs over a network it outputs them through its rightmost outlet. This gives the user the option to use the network clock for applications other than quantisation.

- Signal Generators

The patches in this section all receive MIDI-style control messages and output audio signal. The ‘wavevshapes~.pd’ patch employs the wave shaping techniques outlined in chapter 3.5 of the Johannes Kreidler’s book, *Loadbang: Programming Electronic Music in Pd*.[16] This offers the choice of classic synthesis wave shapes and two random options. The first option generates random points on the ‘$0-scope’ array and goes between them with sinusoidal interpolation. The second loads the array with a snapshot of white noise which when repeated at above 20Hz provides a tone. Otherwise this array functions as an oscilloscope displaying the time domain view of the waveform. Below this is a view of the frequency domain of the signal which uses Mike Moser-Booth’s patch ‘spectrum.mmb~.pd’.[17] This patch suitably scales the output of a real fourier analysis to provide an informative view of the output.

Various other features are included in ‘waveshapes~.pd’, among them are MIDI learnable controls for pitch bend and modulation, glissando and an attack, decay, sustain and release envelope generator.

The ‘additive~.pd’ patch is similar to ‘waveshapes~.pd’ in the features it offers but instead of a selection of wave shapes and a frequency domain view of the ouput signal ‘additive~.pd’ offers a bank of vertical sliders. These work in a similar fashion to drawbars on an organ and represent the relative strengths of the partials of the output signal. This enables quick and easy additive synthesis.

The ‘csound~.pd’ object appears to combine aspects of both other signal generators. In some ways it does but the wave shaping is really just driven by additive synthesis so it is not

[16] Johannes Kreidler, *Loadbang: Programming Electronic Music in Pd*, Mark Barden (trans.), (Fuldaer Verlagsanstalt, 2009), 146–162.
[17] https://github.com/dotmmb/mmb

as drastically different a synthesis technique as those found in ‘waveshapes~.pd’. The real purpose of ‘csound~.pd’ is to act as a simple front end for the combined score document ‘pdwaveshapes.csd’. This offers an alternative to Pure Data’s signal generators for those who prefer the sound of Csound. Capable Csound users are able to edit ‘pdwaveshapes.csd’ to suit their own ends.

All of the signal generators (and the patch ‘parametric~.pd’) use a pre-set system driven by Mike Moser-Booth’s patches ‘save.me.mmb’ and ‘my.hero.mmb’. This system searches the directory that the patch is saved in for pre-sets to populate the dropdown menu in the top right hand corner. When a pre-set is saved in a different location then the dropdown menu is populated by the contents of the save folder instead. Pre-sets are saved as text files and to ensure that one abstraction does not load a pre-set intended to be loaded by another abstraction the pre-sets are saved in the format ‘presetname.abstractname.txt’. So for example if you save your ‘waveshapes~.pd’ pre-set as ‘Pad’, the file created will be called ‘Pad.waveshapes~.txt’.

- Effects/Dynamics

The ‘parametric~.pd’ abstraction is a five band parametric equaliser which uses the fast fourier transform based filter from chapter 3.8.2.1 in the *Loadbang* book.[18] Though the array controls the filtering, the drawing of the curve onto the array occurs entirely on the control level and is only linked to the filtering process through the array.

The curves are drawn onto the array using a reapplication of the same sinusoidal interpolation used to create waveforms for the ‘Random 1’ setting on ‘waveshapes~.pd’. This

[18] Johannes Kreidler, *Loadbang: Programming Electronic Music in Pd*, Mark Barden (trans.) (Fuldaer Verlagsanstalt, 2009), 184.

all happens inside the subpatch ‘pd interpolator’. As with all of the patches in the ‘Effects/Dynamics’ section, ‘multiplex~’ is used to offer a true bypass.

The ‘compress~.pd’ patch is a tidy, efficient amalgamation of the compressors found in the *Loadbang* book[19] and on the Pure Data Forum.[20] There is the standard addition of a true bypass and the ability to save state with ‘savestate.pd’. Also added is a variable control for look-ahead, which can be used for more advanced styles of compression but causes latency.

The abstraction ‘reverb~.pd’ is a simple gui for the ‘freeverb’ object which offers true bypass and the ability to save state. This is the only stereo effect in the effects suite.

The ‘delayer~.pd’ implementation is based on the efficient ‘simpleDelay.pd’ patch by Ken Cleary.[21] This also offers the ability to save state and a true bypass.

- Mixing

The mixing utilities are intended to work together as a modular mixer. The ‘mono~.pd’ object controls and displays the levels for monophonic signals, the ‘stereo~.pd’ object does the same for stereophonic signals and ‘panner~.pd’ converts a mono signal into a stereo signal with panning functionality.

All of the mixers can have their state saved by ‘statesave.pd’. The solo and mute toggles use global sends to speak to other modules and the potential number of mixer channels with fully functional solo and mute toggles should only be limited by the number of values which can be assigned for the value $-0. Hopefully future versions will offer surround panning and mixing capabilities.

[19] *Ibid.*, 186.
[20] http://puredata.hurleur.com/sujet-6189-simple-compressor-interface
[21] http://puredata.hurleur.com/sujet-2338-simple-delay

- Other

Patches which don't fit into the categories of the other patches include: 'savestate.pd', 'qwertymidi.pd' and 'ioi.pd'. As mentioned before 'savestate.pd' is used to save the state of patches for effects, dynamics and mixing. Simply connect its outlet to the inlet of a message box and connect the outlet of the message box to the inlet of the 'savestate.pd', then click the gui bang and save the patch. This way effects can be made persistent.

For those who don't have a MIDI controller or desire the use of a second one (one to control each voice of an acceleration canon with 'quantize.pd' perhaps) 'qwertymidi.pd' turns a qwerty keyboard into a MIDI controller with variable velocity and a range of a little over an octave beginning at middle C on the letter 'A'. This is depicted in figure 11.

Figure 11

The 'ioi.pd' patch measures the inter-onset interval between the last bang it received and the bang that preceded it. This is a simple patch but can be very helpful when working with acceleration canons, for testing and also when experimenting with the human thresholds of audio perception.

Demo Canon and Conclusion

The patch 'Demo_Canon.pd' briefly outlines some of the possibilities of this software. It exemplifies an acceleration canon in which there is both accelerando and

ritardando, the voices converge on a set point and use it a structural point on which to diverge and the beats are subdivided.

Though this shows one way in which these tools can be used it is not intended as a blueprint for how they must be used. This project is aimed at being accessible to users but any time a choice had to be made between accessibility and versatility the decision was made to make the tools more versatile.

The goal of this project was never to offer a ‘canon by numbers’ solution to acceleration canons and to do so would render it pointless. The attraction for many people to these canons is their variety and novelty. Every conclusion will be different.

Bibliography

Callender, Clifton, 'Formalized Accelerando: An Extension of Rhythmic Techniques in Nancarrow's Acceleration Canons' in *Perspectives of New Music* Vol. 39, No. 1 (Perspectives of New Music, 2001), 188–210.

Drott, Eric, 'Conlon Nancarrow and the Technological Sublime' in *American Music*, Vol. 22, No. 4 (University of Illinois Press, 2004), 533–563.

Greeson, James R. and Gretchen B. Gearhart, 'Conlon Nancarrow: An Arkansas Original' in *The Arkansas Historical Quarterly*, Vol. 54, No. 4 (The Arkansas Historical Association, 1995), 457–469.

Kreidler, Johannes, *Loadbang: Programming Electronic Music in Pd*, Mark Barden (trans.) (Fuldaer Verlagsanstalt, 2009).

London, Justin, *Hearing in Time: Psychological Aspects of Musical Meter*, (Oxford: Oxford University Press, 2004).

Reynolds, Roger, 'Conlon Nancarrow: Interviews in Mexico City and San Francisco' in *American Music*, Vol. 2, No. 2 (University of Illinois Press, 1984), 1–24.

Web Resources

https://github.com/dotmmb/mmb

http://myweb.fsu.edu/ccallender/

http://puredata.hurleur.com/sujet-2338-simple-delay

http://puredata.hurleur.com/sujet-4261-midi-learnable-mixer-channel

http://puredata.hurleur.com/sujet-6189-simple-compressor-interface

http://www.kylegann.com/TempoCanon.pdf

http://www.nancarrow.de/arbeitsweise.htm

Word Count: 3,964