Revista Vórtex | Vortex Music Journal | ISSN 2317–9937 | http://vortex.unespar.edu.br/
D.O.I.: ______________________ [Fonte 8 | Cambria | Espaçamento 1.0]

Elastic Rhythm in Signal-Synchronised Sequencing Objects for Pure Data

Dr Edward Kelly

Synchroma Audio Engineering

Abstract: this paper presents a family of objects for manipulating polyrhythmic sequences and isorhythmic relationships. These work together and are tightly synchronised to an audio signal, so that relative temporal relationships can be tempo-manipulated in a linear fashion. Many permutations of polyrhythmic sequences including incomplete tuplets, scrambled elements, interleaved tuplets and any complex franctional relation can be realised. Similarly, these many be driven with controllable isorhythmic generators derived from a single driver, so that sequences of different fractionally related lengths may be combined and synchronised. It is possible to use signals to drive audio playback that are directly generated, so that disparate sound files may be combined into sequences. A set of sequenced parameters are included to facilitate this process.

Palavras-chave: apenas para trabalhos em português ou espanhol. Até 5 itens separados por vírgula. Palavra-chave 1, Palavra-chave 2, Palavra-chave 3, Palavra-chave 4, Palavra-chave 5.

Abstract: em fringilla ut morbi tincidunt augue interdum velit euismod in pellentesque massa placerat duis ultricies lacus sed turpis tincidunt id aliquet risus feugiat in ante metus dictum at tempor commodo ullamcorper a lacus vestibulum sed arcu non odio euismod lacinia at quis risus sed vulputate odio ut enim blandit volutpat maecenas volutpat blandit aliquam etiam erat velit scelerisque in dictum non consectetur a erat nam at lectus urna duis convallis convallis tellus id interdum velit laoreet id donec ultrices tincidunt arcu non sodales neque sodales ut etiam sit amet nisl purus in mollis nunc sed id semper risus in hendrerit gravida rutrum quisque non tellus orci ac auctor augue mauris augue neque gravida in fermentum et sollicitudin ac orci phasellus egestas tellus rutrum tellus pellentesque eu tincidunt tortor aliquam nulla facilisi cras fermentum odio eu feugiat pretium nibh ipsum consequat nisl vel pretium lectus quam id leo in vitae turpis.

Keywords: polyrhythm, polymeter, sequence, isorhythm, synchronised

Paper's title translated to English; please adapt to a maximum of 90 characters with space. Recebido em 00/00/2021. Aprovado em 00/00/2021. Disponível online: 00/00/2021. Editor: _____. Creative Commons CC-BY [Fonte 8 | Cambria | Espaçamento 1.0]

Introduction

While the use of metro objects in Pure Data (Pd) is for many purposes completely adequate for creating complex rhythmic relationships (polyrhythms), another method can be derived from an audio signal (phasor~ object) by defining an event as the moment at which a signal crosses a threshold. There are benefits to this technique that become apparent when manipulating playback speed, in that the signal driver may be manipulated to generate playback drivers for tabread4~ objects, and hence when speeding up and slowing down the speed of the driver, it has the effect similar to speeding up and slowing down a vinyl record. Pitch relationships in a sequence manipulated this way are preserved, and signal drivers for sequenced events may be normalised or not. It is relatively simple to switch to another model where pitch is irrelevant of the rhythmic tming of the sequence using the rhythmic events generated. Although the method for achieving this is event-based (rather than a direct signal) it is still derived from threshold-crossing of the signal input (from a phasor~ object in Pd) and so the timing of these events is as accurate as it can be within the signal/event paradigm of a Pure Data object, within about 1 ½ milliseconds[1].

Any sequence of fractional polyrhythm may be used, including the use of incomplete tuplets such as 4/5ths followed by 2/3rds, although such sequences should include completion elements later on i.e. (in this case) 1/5 and 1/3.

Such sequences may be scrambled within the object, and novel serial polyrhythmic elements organised into sequences may be generated on-the-fly. Various methods are used to navigate between sequences either instantaneously or in sequential order.

It is important to note the distinction and difference between polyrhythm (different numbers of regularly spaced events within a given, fixed time frame) and polymeter (different integer groups of events at a specific tempo). Hence we will be using fractional notation to describe polyrhythmic relationships (e.g. 6/12 is equal to 6 triplets of 8th notes (quavers)) and ratio notation to describe polymetric relationships (e.g. 5:4 means that their are two independent loops, one 5 beats long and another 4 beats long, so that by the time the first plays the 5th beat of its time signature, the other is playing the first beat again, and the loops re-synchronise at 20 beats (5x4).

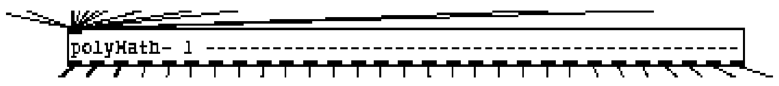
1. 1.45125ms is the maximum delay at 44.1KHz sampling rate, block size of 64. 1.33333 at 48 Khz. Lower values can be achieved with higher sampling rates, and hence greater temporal accuracy.

Paper's title translated to English; please adapt to a maximum of 90 characters with space. Recebido em 00/00/2021. Aprovado em 00/00/2021. Disponível online: 00/00/2021. Editor: _____. Creative Commons CC-BY [Fonte 8 | Cambria | Espaçamento 1.0]

At first, the notation appears to be counter to that which is normally used in music notation: a time signature of 5/4 against one of 4/4 produces the polymetric result and within that there may be triplets, quintuplets and so on – the mathematics of working out how to caluclate a linear phase value from a polyrhythm are much more coherent with the fractional approach, and polymeter is best expressed in terms of ratios. Fundamentally there are two objects presented here: polyMath~ which deals with polyrhythmic sequences and playback, and isoWrap~ which deals with polymetric ratios. The two may be combined to create novel systems of rhythmic organisation, with extra features provided such that a number of parameters may be sequenced simultaneously.

1. polyMath~

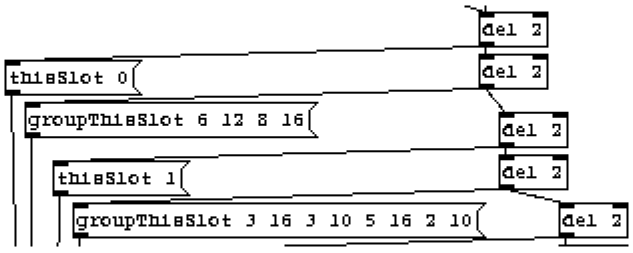
FIGURE 1 – The polyMath~ object



The polyMath~ object has 1 inlet and 29 outlets, 3 of which are signal outlets. Many of the outlets represent features of the grouping system, some are for sequencing extra parameters along with rhythmic events, and some are designed for early instigation of event parameters so that (for example) the correct sample array will be played when the signal outlets are used for this purpose.

Rhythmic events are organised into groups, where each group is a continuous sequence of events with the same time denominator (rhythmic note value). Using the fractional notation system for polyrhythmic sequences, we can use this format to instigate rhythmic sequences, using thisSlot and groupThisSlot messages as in figure 2. Note that in the first instance we are programming complete tuples (6/12, 8/16) and this is more like conventional music notation, whereas the second instance has broken tuplets (e.g. 3/10). Each sequence still adds up to 1.0 though, if each pair of values is taken as a fraction. 6/12 + 8/16 = 1.0. In the second sequence it is clear how complete sequences of polyrhythmic tuplets can be made from disparate groups of rhythmic elements i.e. we can have 3 quintuplets on their own, if we follow later on with two more quintuplets (making 5/10 which adds up to a single ¼ note or crotchet).

FIGURE 2 – Programming Slots (Rhythmic Sequences)



Internally, each sequence is a slot, and each slot has a limited number of variations, with variation 0 as the original programmed sequence. There are currently two versions of this object, polyMath~ and polyMathLite~, and each has a limited number each of slots, variations, groups-per-slot and elements per slot or variation. Since the buffers are a fixed size in this object, they use a predictable amount of memory, and the polyMath~ object can use a significant of memory if a number of the objects are used, the other configuration of the object included for use in lower memory applications. These objects were always intended for use in both desktop and mobile applications, so a knowledge of memory usage (and fixed memory usage) may be important in such scenarios.

Currently the configurations are as follows:

TABLE 1 – Object Configuration

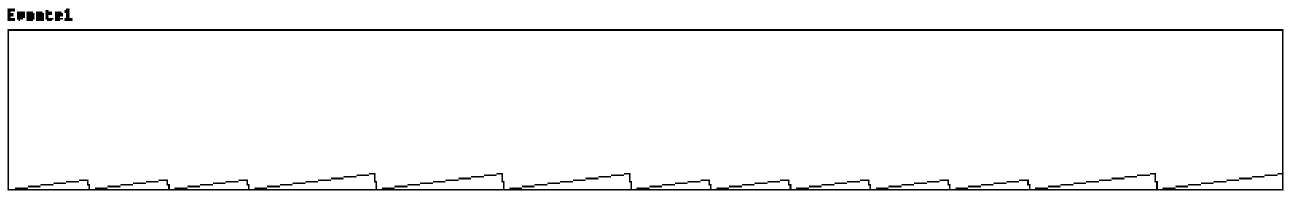
	polyMath~	**polyMathLite~**
Slots	128	64
Elements-per-slot/variation	2048	512
Groups-per-slot/variation	256	128
Variations per-slot	6	2
Extra sequence pairs	8 x 2	4 x 2

Within each slot or variation, a maximum number of groups is allowed. The groups themselves are collections of elements with identical fractional denominators, so 4 / 12ths would be a group but 3 / 16ths + 2 / 20ths would be two groups. Note that incomplete tuplets (4 /12ths meaning 4 x 8th-note triplets) are acceptable. Within a complete phase cycle, they can be completed by the same rhythmic type, or related (binary subdivision) later on in the sequence: 4/12 + 4/8 + 4/24 = 1. Hence the time signatures are used as fractions in order to calculate the phase of each event.

2. Event Ramps

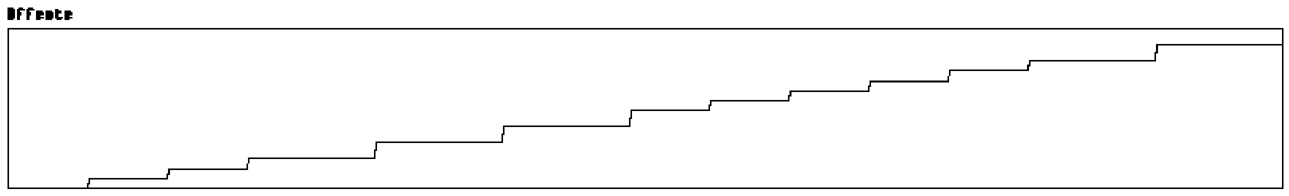
Each element, as it is played, generates a number of data points as Pd events, but also signals that may be used to drive the playback of one audio file, or multiple audio files where sequence pairs are used. The simplest form of this is that we use a set of ramps to play back one (or more) audio files in sequence[2] and the first signal outlet will generate in it's most basic settings a graph such as that shown in figure 4, which has a rhythmic sequence of 3 / 16 + 3 / 10 + 5 / 16 + 2 / 10.

FIGURE 3 – Raw Ramp Output from Signal Outlet 1, with s Sequence of 3/16, 3/10, 5/16, 2/10



If we were to play a single audio file as it was originally recorded using this, the third signal outlet would give the offset, and added to the amplitude of each ramp segment a smooth line from 0 to 1 would result[3].

FIGURE 4 – The offsets signal



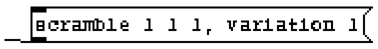
3. Scrambled Variations and Alternating Play Heads

In the case of playing back a single soundfile using the scheme developed above, there is no reason to do this. The result of adding the ramps to the offsets is a signal ramp that goes seamlessly from 0 to 1, so it is the same as the phasor~ output that feeds the object, multiplied by the sample array length to play an array at any speed. As is shown later, this information can be used to drive various forms of concatenative (or other) synthesis. Within the context of the audio loop however (or any other use of the sequencer) a scramble message can be used to create a variation of the sequence.

2. Sequence pairs may be used to set an audio file, although as will be shown a method for generating an early sequence pair and using alternate signal paths will be preferable.
3. Note that we are only considering a single "page" of sequence, i.e. a single phase signal from 0 to 1.

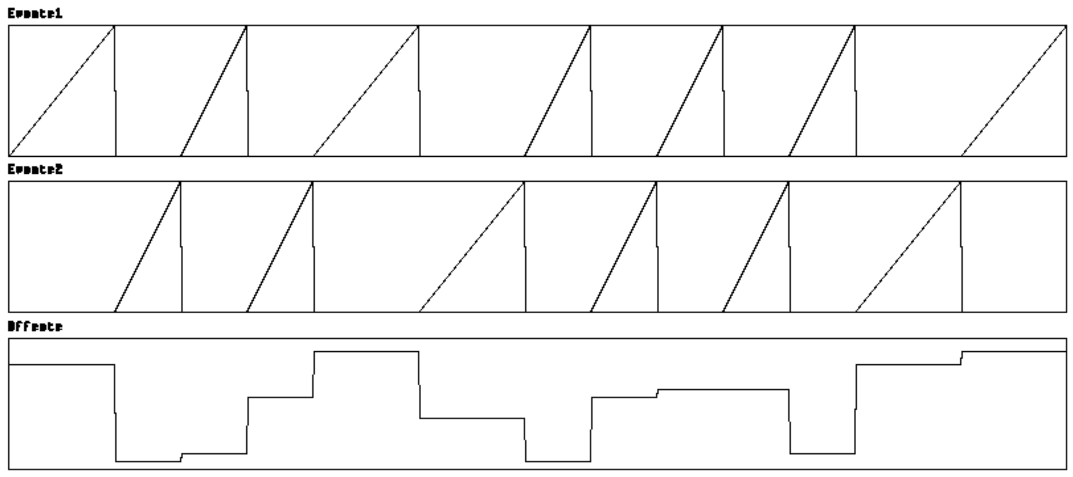
Paper's title translated to English; please adapt to a maximum of 90 characters with space. Recebido em 00/00/2021. Aprovado em 00/00/2021. Disponível online: 00/00/2021. Editor: _____. Creative Commons CC-BY [Fonte 8 | Cambria | Espaçamento 1.0]

FIGURE 5 – This message means scramble (slot) 1, (variation) 1, (amount) 1, then switch playback to variation 1.



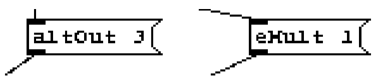
```
scramble 1 1 1, variation 1
```

FIGURE 6 – A sequence that has been scrambled to generate a variation, with alternating "play head" signal outlets and normalized event ramps.



The variation in figure 6 shows three new aspects. The first is that the same ramps and offsets are aligned, but have been re-ordered. The second is that the ramps are now normalized from 0 to 1, and the third is that there are now alternate signal ramp outputs from the first and second outlets of the object. If we didn't normalize the ramps like we did here, we could read a scrambled variation of a single audio file by adding all three signals together, and it would have the same rhythmic values of the original sequence but in a different order. But the audio ramps alternating and normalized could be used to read *any* set of audio files, or sections of audio in any set of files, and re-order this sequence (or even use the original sequence to play a new set of files). If so, then it is necessary to find a way of setting audio tables from within the object, perhaps even setting the length of a section of an audio file per-section, and even generating any arbitrary parameters for this sequence to occur smoothly.

FIGURE 7 – The altOut message sets whether to send event ramps out of one or two outlets. Any number greater than 0 will also set the number of sequence parameters to send early. The eMult message sets whether the event ramps are normalized or not.



```
altOut 3        eMult 1
```

4. Sequence Parameters and Early Messages

Along with the event sequence, there are a set of sequenced parameter pairs that are output as the sequence unfolds. They are organised into pairs so that they may be routed to diverse parameters downstream of the object in a Pd patch. These may be used to set the alternate event ramps to each play a different audio file, or sections of audio in different files, as well as controlling other parameters of signal processing and generation.

FIGURE 8 – Sequence parameter pairs, alt(ernate) and early parameters.

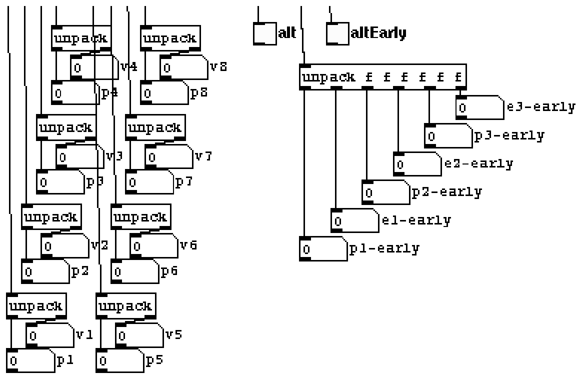


FIGURE 9 – The precent parameter sets the percentage of the current event phase to wait before triggering early parameters

Using alternate event-playback ramps would be challenging if the signal to play back an audio file was generated while the event to change the audio file waited until the end of the DSP block[4] (usually 64 samples) to do this. Clicks would result, as the difference between the time resolution of the signal path and the event path is anything up to a block - 64 samples by default, although this may be set within a patch using switch~ and block~ objects.

If a sequence parameter held information as to which sample table was *about* to be played by the next ramp to start, it could be initiated before the ramp starts. Since we are concerned with pre-programmed sequences, this is possible and once again is triggered by a signal crossing a threshold. This time, it is the current event ramp signal which generates the early parameter output – the signal of the current ramp crosses a threshold, and the early parameter output can be used to determine what the *next* playback instance (soundfile, wavetable etc) will be. The *alt-early* toggle happens before this, and so the early

4. Digital Signal Processing block size – by default in Pd this is 64 samples, and events generated within this time are transmitted at the end of the block, whereas a change in the audio signal may happen at any time within the block.

messages can by sent to different paths using spigots, to instruct one of two playback objects which file to play (for example), while the alternate ramp is still going (it is the signal outlet presently ramping that initiates the early signal to generate this event) and so the signal to change sample tables on one ramp is sent by passing a threshold (termed the "precent" value) on the other ramp. Once again, a threshold-based system based on a signal, can be used to sequence array-based sequences with an audio resolution equivalent to vinyl[5], but with the flexibility of sample-based concatenative synthesis. The number of sequence parameters that are output in this way is controlled by the number sent to the altOut message, and so there is potential for initialization of several parameters on each channel, ready before playback commences.

5. All of the Information at Once

If all of this complex rhythmic information is held in a monolithic object, and there is only one method to retrieve it (i.e. waveform playback using sample tables), it could be functional but it also points towards a didactic approach to its use. Considering the amount of information about each sequence stored within the object, it therefore makes sense for the object to make all of this information. To be specific, each sequence is stored in a variety of forms specific to the structural information required to process and interpret it.

5. This has not been tested where the event – the "precent" of the event is less than the blocksize. Other methods are available for dividing up audio files and sequences in millisecond scales.

Paper's title translated to English; please adapt to a maximum of 90 characters with space. Recebido em 00/00/2021. Aprovado em 00/00/2021. Disponível online: 00/00/2021. Editor: _____. Creative Commons CC-BY [Fonte 8 | Cambria | Espaçamento 1.0]

FIGURE 10 – polyMath~ outputs information about the current sequence in many forms

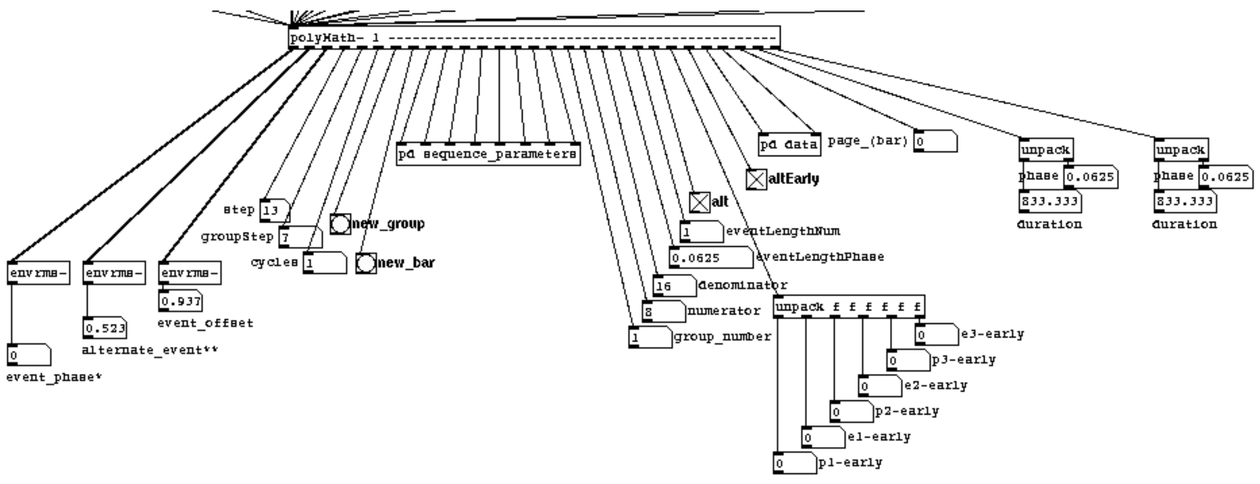


TABLE 2 – Non-signal Outlets

Outlet	Name	Description
4	step	Which sequential event is this?
5	groupStep	Which event is this in the current group (e.g. 2 out of 5, starting at 0)
6	cycles	How many cycles of the phasor~ does this sequence take?
7	new group	Bang when a new group begins
8	new bar	Bang when a new bar (or cycle, or page[6]) begins
9-16	sequence parameters	8 pairs of floating-point numbers, to route to any parameter
17	group number	Which group is this?
18	numerator	How many rhythmic intervals (e.g. 6 if the group is 6/8)
19	denominator	The rhythmic unit (e.g. 12 if the group is 5/12)
20	eventLengthPhase	Phase value (between 0 and 1) for the current event
21	eventLengthNum	How many multiples of the denominator
22	alt	Which signal ramp outlet is currently firing
23	Early sequence parameters	What happens next?
24	altEarly	Which event ramp signal is the next to fire
25	DataOut	When outputting data, it appears here
26	DataType	Which type of data are output
27	Page	Which cycle are we on (polyMath~ patterns can span multiple cycles of phasor~ output)
28	durFirst	Duration / phase pair of current event
29	durAlt	Duration / phase pair of next event (early)

Taking a look at the full output from the object, it is clear that there are many approaches to using this rhythmic coherence of phase values, numeric musical representations of rhythm, and all of the data that

6. Bar, cycle or page? – this terminology has yet to be finalised. They are all relevant, but cycle is more relevant since bar implies 4/4 time signatures due to the way phase signals are divided up. The idea of a "page" is useful since it does not imply completion.

Paper's title translated to English; please adapt to a maximum of 90 characters with space. Recebido em 00/00/2021. Aprovado em 00/00/2021. Disponível online: 00/00/2021. Editor: _____. Creative Commons CC-BY [Fonte 8 | Cambria | Espaçamento 1.0]

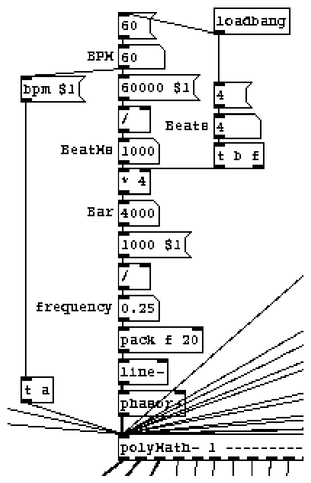
resides within the object as it unfolds as a sequence. It may of course be used in a simpler way, perhaps by triggering events from the change of the step outlet (outlet 4), or the raw phase values could be used to set fast (or slow) loop points within array playback, so the purpose of the 5 year-development process has not been to provide a monolithic way of controlling sequences of recorded audio samples. It has been always in the service of complex rhythm.

Without re-writing the object, it already has many approaches to rhythm in its 27 event outlets. Note that there are statistics as to how the rhythmic elements are organised within the sequence. The denominator (e.g. 8th notes, or perhaps 17th notes) is shown. Every event message is triggered before the "step" outlet so if all is needed is a polyrhythmic modulating rhythmic pulse then this is fine, but there are numerous ways in which the analytical data (and sequence data) from the other outlets could be used. Regardless of variations and scrambling permutations, this object allows for the storage of complex polyrhythmic patterns that will be tightly synchronised to audio-rate events.

6. Polymeter and Macro-Structure with isoWrap~

While it is entirely possible to create any sequence of polyrhythm and polymeter within a single sequence using polyMath~, to generate sequences of different lengths and synchronise them using more than one instance of polyMath~ requires more attention to the phasor~ signal that is used to drive the object.

FIGURE 11 – Basic conversion of musical time to frequency, for phasor~ input to polyMath~[7]



7. Note that the bpm message only applies to the outputs of outlets 28 and 29, where duration is stated along with phase.

Paper's title translated to English; please adapt to a maximum of 90 characters with space. Recebido em 00/00/2021. Aprovado em 00/00/2021. Disponível online: 00/00/2021. Editor: _____. Creative Commons CC-BY [Fonte 8 | Cambria | Espaçamento 1.0]

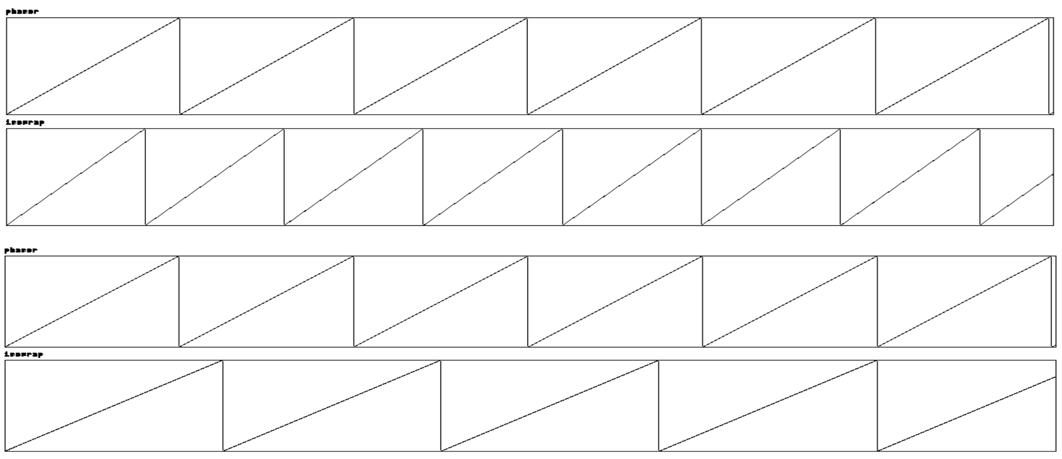
Consider the situation where two sequences, one of five beats lengh, and another of four, are made to play synchronously in one of the following scenarios:

1. The four-beat rhythm is slowed down só that it takes as much time as the five-beat rhythm
2. The shortest rhythm plays to the end and then stops
3. The shortest rhythm is played synchronously with the longest, but looping at the end of each bar causing an isorhythmic effect

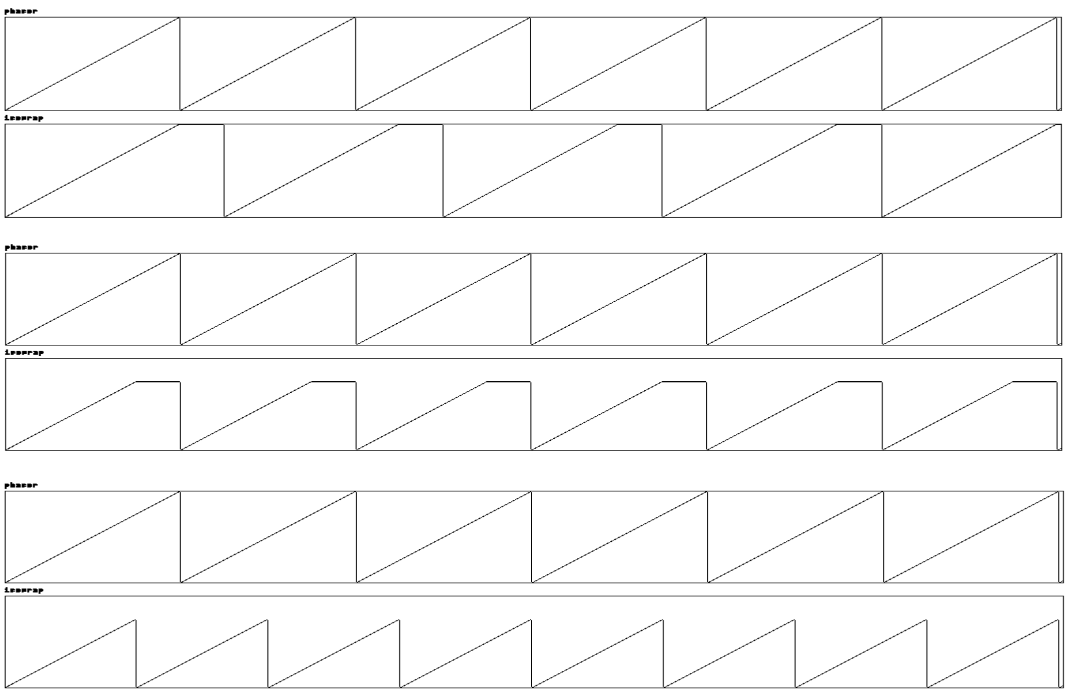
isoWrap~ takes the phasor~ signal and applies ratio-based temporal wrapping – it generates sympathetic phasor~ signals based on ratios of bar lengths and normalization values, só that two polyMath~ objects can be operating on different but related rhythmic bases without falling into or out of synchronisation. There are also several normalization functions só that the signals driving the polyMath~ objects can adopt different pulse-relationships on-the-fly, slip in-and-out of synchronous operation, and translate the rhythmic relationship to a different fractional relationship.

Some examples of phase relationships can be seen below. It is worth noting that polyMath~ sequences can go on for multiple "bars" - i.e. 0-to-1 phase ramps from phasor~, and the reset from a high value to a low one tells the polyMath~ object when to switch to a new "bar" or page[8]. With isoWrap~ the order in which the fractional relationships between the polyMath~ objects is permutable in real time, and their polymetric, indeed isorhythmic relationships may be developed.

FIGURE 12 – Some potential phase relationships between raw phasor~ input and isoWrap~ output at different settings.



8. Since the notion of a bar is arbitrary here, and not necessarily the entire passage of a phase signal from 0 to 1, perhaps we should just use a literary metaphor instead, as in some other branches of computer science.



These relationships are controlled within sections of the process prior to the injection into any polyMath~ object, and só macro- polymetric and polyrhythmic relationships are relatively simple to establish using this tool kit of objects.

It is important to realise that these signals are the drivers for polyMath~ objects, and within each of these ramps these signals can regulate polymetric or polyrhythmic structures at a microscopic scale, since if each phase signal controls a polyMath~ object, there is a flexible approach to rhythmic relationships on both the macro- and micro-scopic scale.

An example of the isoWrap~ objects combined with polyMath~ objects in a simple sample player (using metastudio objects[9]) is shown in figures 12 and 13. There are a number of assumptions – e.g. the sample arrays will be indicated by a number (and wavebank~ with its numbering system from 0001 to 9999 makes this easily possible using the "early" sequenced parameters.

9. https://shatrktracks.co.uk/html/software.html

Paper's title translated to English; please adapt to a maximum of 90 characters with space. Recebido em 00/00/2021. Aprovado em 00/00/2021. Disponível online: 00/00/2021. Editor: _____. Creative Commons CC-BY [Fonte 8 | Cambria | Espaçamento 1.0]

FIGURE 12 –Example sample player patch with isoWrap~ and polyMath objects.

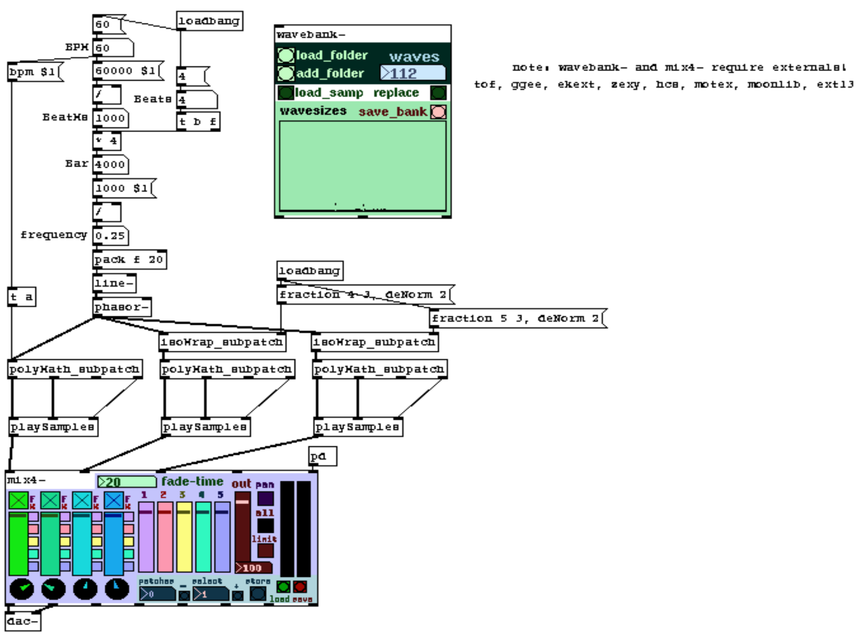
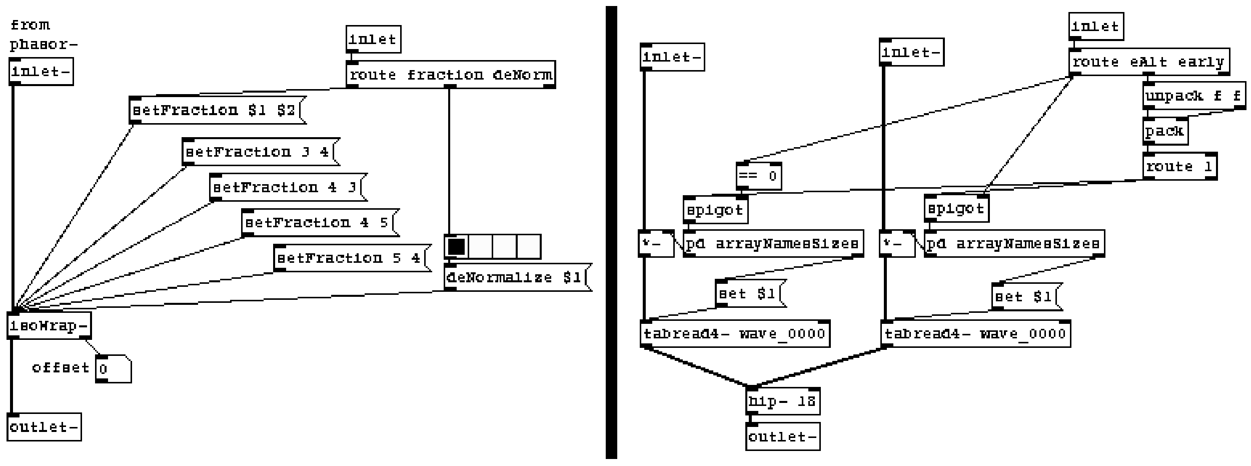


FIGURE 13 – isoWrap (L) and playSamples (R) subpatches. The playSamples subpatch shows the connections from the two alternate signal outlets of polyMath~, and some (prepended) alt-early and early parameters for controlling which array is played..



Conclusion

Polyrhythm and polymeter are concepts that have had some traction in contemporary culture, but any one example of these techniques in a piece of work is just a part of a network of rhythmic concepts – sets of conceptual ideas and techniques that encompass everything we do already know and use, but present a landscape where available tools become the means by which new musical territories can be explored.

The polyMath~ set of objects is a toolkit for exploring these realms. It is presently experimental and features are developing, but it offers a way to use rhythm as a continuum of events, rather than a single unassailable clock.

There are many ways in which this piece of software can be used, and there are also many ways in which it could be modified to form a family of objects that are more agile in aims toward the goals for which they are intended. This is a project to open rhythmic flexibility as a practical construct to be modulated in expressive, musical ways, as a tool not just for composition but also for on-the-fly improvisation with complex rhythmic phenomena.

ACKNOWLEDGMENTS

Much of the code for this was written during sessions of the Lambeth Music Service while my son had drum lessons. https://www.lambethmusic.co.uk/lambeth-sounds-partners.html

I am indebted to Simon Limbrick, who is always ready to engage with complex material. https://www.marimbo.com

Joao Pais has a wonderful metronomic approach to polyrhythmic metronomy, especially good for instrumentalists - https://jmmmp.github.io/clicktracker/

The full suite of polyMath objects can be downloaded from:

https://github.com/SynchromaAE/polyMath

REFERÊNCIAS / REFERENCES

to follow

ANEXO ou APÊNDICE / ANNEX or APPENDIX

to follow