Music in expression - A DSP based compositional methodology.

Chun Lee
75 Bromfelde Road
London, UK, SW4 6PP
chun@goto10.org

Abstract

Composing music in Pure Data, conventionally involves the use of a discrete messaging system for ryhthmic sequencing and audio rate signals for sound synthesis. While such a combination serves its purpose well and has its merits, the aim of this paper will attempt to outline an alternative approach - using audio signals only for creating both time related events and sound synthesis - and discuss various interesting factors which arise.

Keywords

DSP, Sequencing.

1 Introduction

Pure Data, as a programming environment for music composition, can be described to contain the following two components for its functionality: a time-based discrete messaging system and a DSP engine. This design is commonly shared by many other programming languages which deal with real-time audio signal processing. In Pure Data, operations associated with `metro` and `list` objects, as well as messages, signifies the characteristic of the former domain. The latter, on the other hand, can be realised in simple process involving signal manipulation with `*~` or `+~` objects.

How did these two distinct types of systems come into coexistence? Is it due to historical computing hardware and software developments? Is it to maximize and combine the benefits from each individual domain? Is it simply a traditional artistic tendency that sees music divided into its abstract time-based structure and its real-time interpretation?

Although the investigation into questions above are not within the scope of this paper, it did provide the starting point and initial challange: to compose only using the DSP engine, without the discrete messaging system.

1.1 Motivation

By removing the messaging system entirely, several interesting properties can be observed. First, it encourages a more minimalistic approach in composition. In other words, instead of having separate functional domains, only one remains. Moreover, DSP procedures that can be used to generate musical structure, as well as to synthesise sound are often based on the same set of objects, thus further simplifying the necessary building blocks in composition. Additionally, all operations, be they structural control or sound synthesis, are directly interchangeable, allowing easy integration between one and another.

Although certain capabilities [1] in the “discrete” domain may or may not be fully implemented in a pure DSP manner, the simplification gained would likewise be uncertain to recreate with the conventional approach. Therefore, it is necessary to point out that replacing the messaging system would never be the aim of this investigation; this paper aims only to propose an alternative compositional method with a possible potential.

Second, since the remaining operational domain is the DSP engine, this implies a degree of inherent generality. In other words, different DSP engine may vary in the technical implementation, the behavior they produce, should be reliable and fulfil its generic tasks. A chosen wave oscillator would produce the correct and corresponding audio signal, regardless of the environment. For example, the methodology presented in this paper have also been experimented in other languages such as SuperCollider, and has behaved the same way as in Pure Data. The effects of generality would therefore suggest any derivative procedures to be just as portable, as well as robust.

Finally, the continuous quality in signal domain creates many intriguing conceptual analogies extending beyond its original context. The method in which patterns and structures are derived from audio signal, as well as being able to directly feed these signals back to sound synthesis

[1] Such as list processing and manipulation.

closely resembles the principle of control voltage and modulation found in analog modular synthesizers. Moreover, the utilisation of pulse wave is comparable to the basic functioning of digital integrated circuits. Whether or not these similarities are scientifically sound or purely metaphorical, they certainly do provide an additional dimension for experimentation at the artistic level.

1.3 Musical context

Before any practical example is presented, it is important to clarify the term “musical composition”. Depending on the genres of music being made, their corresponding method of composition can differ. In other words, certain types of compositional procedures suits particular kinds of music better.

In this paper, “musical composition” refers to music that is predominantly sequenced or rule based, with a deterministic tonal outcome. Specifically, the genre of “video game music” can be best used to give it a concrete context. Apart from the limited scope of investigation, as well as personal preference, such a choice of genre also carries some practical advantages.

The chosen type of music typically has very distinct, well defined characteristics, which helps to clearly outline, as well as constrain, the parameters for compositional experiments. Furthermore, since sequencing is conventionally taken care of by time-based messaging, and is an essential element in many types of music making, being able to perform such a task in the DSP-only approach would demonstrate the versatility of the proposed method.

2 Operation examples

2.1 Design principles

In the following section, the DSP-only compositional method will be introduced with practical examples. Details surrounding them will also be highlighted and discussed.

Since sequencing is a vital component in the composition of the chosen genre, the objectives of experimentation are therefore primarily aimed to address this issue. The design principles, in particular, can be defined below:

- To create discrete segments from a continuous signal.
- To isolate or select certain segments within the sequence they belong to.
- To place the chosen segments on desired points in time.

Two simple mathematical functions are employed extensively to achieve the necessary outcome; these are the floating-point remainder and greatest integer. Both are utilised from the DSP version of the expression object. They represent `expr~ fmod($v1, 1)` and `expr~ floor($v1,0)` repectively.

2.2 Step counter

It can be argued that step counting is one of the most fundamental aspect in sequencing of any type. Its task is to separate a given unit of time into identifiable subdivisions. In a discrete domain, this is typically implemented with a recurring event, triggering a accumulative variable, which adds a constant value to it at each iteration. While such an operation may seem intuitive, producing the same effect with continuous audio rate signal requires a slightly different approach.

In order to do so, a sawtooth wave is used as a control signal, with its period representation the duration in which consequent steps will be derived from. This signal is first multiplied with the number of steps to be counted, then the value of the greatest integer is taken from it. This results to the counting of equal steps within one wavelength of sawtooth wave.

The technique of using greatest integer represents a quick and simple way (see Figure 1) of obtaining an desired stepping signal. Typically, it can be realised within one expression object in the form of `expr~ floor($v1*$v2, 0)`, where $v1 is the control signal and $v2 being the number of steps in counting.

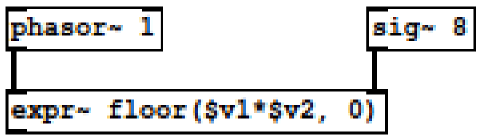

Figure 1: Step counting using floor function

Moreover, with additional arithmetic operations, its output can be transformed into other variants of counting. However, due to its simplistic nature, obvious limitations can be quickly identified.

In discrete counting, the triggering events can easily be obtained and used for other process to influence the output. For instance, these triggers can first undergo certain probabilistic evaluations, in order to determine the exact algorithmic behavior in the following iteration. Since the precise moment at the beginning of each step cannot be reliably isolated, the method in Figure 1 therefore can not achieve this type of result.

To remedy this in the DSP sequencing, a different implementation needs to be devised. Still relying on the `phasor~` object as control signal, one possible alternative design would be to use the sample-and-hold (see Figure 2) function to extract the falling edge of sawtooth wave, and utilise it for any additional operation[2].

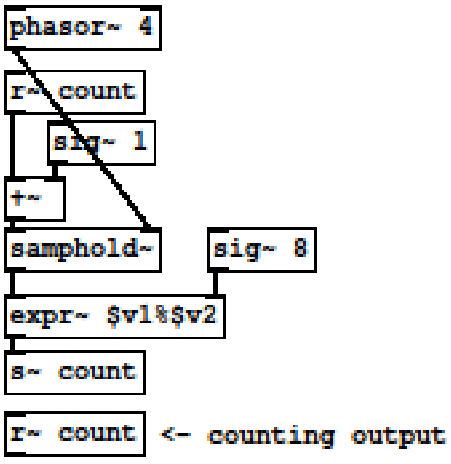

Figure 2: Step counting using sample-and-hold function

If such a method is to be adopted, its surrounding issues should also be noted. First, the wavelength of the control signal now signifies the duration of each individual step, instead of the entire counting cycle. Second, depending on the implementation detail, a pair of DSP `send~` and `receive~` objects may be used to act as the conduit to pass the "accumulative" counting signal back into the `samphold~` object. Although this method has proven to work, it does so by exploiting the feedback loop otherwise not permitted with DSP objects. As a result, the delay introduced by the send and receive objects may cause undesired effect and needs to be taken into account. Lastly, as it takes several objects to accomplish simple sequencing, it often requires noticeably more, as well as an elaborated setup to increase capability and flexibility. In other words, the relationship between the effectiveness of the desired outcome and the maintainable level of complexity in the patch may work against each other.

When considering these implementations side by side, their differences quickly become apparent. While the "greatest integer" is fast to program, with limited functionality, the "sample-and-hold" alternative can be more cumbersome to realise, but with possible flexibility to gain from. This contrast between them, therefore, can be argued to demonstrate the range of possibilities in DSP sequencing. Depending on the goal of composition, one might be more preferable than the other. In the context of this paper, the former seemed more suitable, as it allows a more rapid development and experiment to emerge, which does not obstruct the focus in composition.

There is one other method to produce sequential DSP signal involving the use of pulse wave. A example of this will be covered in the later part of this paper.

2.3 Structure derivation

The rhythmic system of any genre under the influence of western classical music theory is fundamentally based on subdivision. For example, a bar of 4/4 measure consists of four quarter notes, each of which can be further divided into two eighth notes or four sixteenth notes. This segmentation can continue to any arbitrary level according to the compositional need. As the divisions are based on whole number integer, the rhythmic system is highly hierarchical and largely symmetrical. Furthermore, the point of reference in subdivision can arguably also be arbitrary. That is to say, it can start at the level of one bar measure, or start at the composition as a whole. In the latter case, the top-down hierarchy of division can be seen as: composition, sections, phrases, bars and beats. In such a case, it is interesting to notice the possible crossover between subjective abstract "musical structure" and objective concrete "rhythmic duration".

To an extent, step counters can already pro-

[2]An example of this can be seen in Figure 6, where signal of random value is obtained

vide the means to materialise the multiple metric levels required in composing rhythmic patterns. In essence, sixteenth notes are counts of four against one quarter note, or counts of sixteen in one 4/4 bar so on and so forth. Depending on the desired effect, the "resolution" of counting may be scaled accordingly. However, in order to produce the necessary control signals that correspond to all the possible levels of quantisation, being able to first generate sawtooth wave at an arbitrary scale becomes imperative.

The floating-point remainder operation previously mentioned is employed to achieve this. The reference signal representing the top-level unit to be further divided is first multiplied with the sum of subdivisions required, then the value of remainder module of 1 is taken from it. This results in a sawtooth wave that is higher in frequency than the original signal. While the expression object can be adopted for this in the form of `expr~ fmod($v1*$v2, 1)`, where $v1 is top-level signal and $v2 represents the division scale(see Figure 3), conventional DSP `*~` and `wrap~` object can also be used.

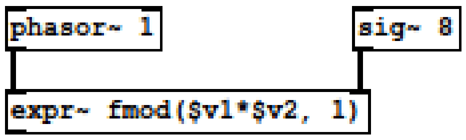

Figure 3: Control signal scaling using fmod function

By now, the parallel between step counting and control signal derivation in the DSP only method should become apparent. First, both operations are based on the manipulation of exactly the same signal. Second, They all require the same scalar value, which defines the "resolution" of the end result. The only difference between the two, therefore, is the use of floating-point remainder and greatest integer function.

This comparison may seem trivial at first, but they may, however, have more subtle and intriguing implications. Because the same signal is used to produce step counting and any derivative control signal, no matter how complex the sequencing operations become, it can always be traced back to one single `phasor~` object. As a result, any modulation introduced to this top-level control signal can bring rapid, sometimes difficult to predict changes to all its subsequent process. By exploring, or exploiting, this property, it may result in a compositional structure which is highly dynamic, flexible in global changes, as well as generating unusual effects. Moreover, besides being used to perform tasks such as nested step counting, the sub-divisional sawtooth signal can easily be adapted to control other aspects of composition or synthesis. For example, envelope can be derived from sawtooth signal with its period duration corresponding to the sequencing steps of the same level. To this end, musical structure and sound synthesis becomes highly integrated.

While sub-dividing a sawtooth signal is simple in the given example, to reverse its result can be problematic. In other words, to construct the control signal at higher level from its divisible constituents is much less an easy task. If the compositional process begins with a single motif or phrase pattern, the ability to switch between current and a higher quantisation level is crucial. To solve this, once a particular sequencing pattern is composed and duplication is necessary for further compositional development, the rate of its original control signal can simply be decreased to reflect the new working duration or metric level. In the instance of replicating a pattern of eight notes twice, the frequency of the original control signal needs to be halved first, then depending on the context, the scalier value for each notes in the pattern can be doubled, or constructing a intermediate two steps counting and signal division.

2.4 Phase Shift

In so far, sequencing has been restricted to only allow patterns that consists of single durational units. If the step counter is set to represent sixteenth of a bar measure, any resulting pattern must contain only notes or rests of this duration. As a result, sequencing patterns that have a different quantisation value than the step counter it relies on would not be possible. Musically speaking, attempting to place an eighth note on the rhythmic grid of sixteenth note is prohibited.

To combat this, the technique of phase shifting could be employed. By offsetting the phase of sawtooth wave, it becomes feasible to move subsequent sequencing steps to any arbitrary position within the wave cycle of the control signal. Once again, the floating-point remainder is involved to modify the phase of a given signal. It takes the form of `expr~ fmod($v1+$v2, 1)`, where $v1 is the control signal and $v2 signifies the amount of phase to change(see Figure 4).

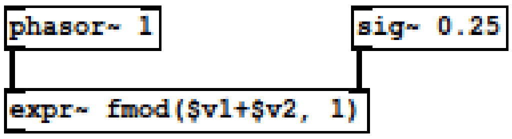

Figure 4: Phase shifting with fmod function

It is worth noting that since the control signal governs the sequencing of patterns, the musical context of shifting its phase can produce a syncopated, or upbeat effect. Therefore, the term "phase" used here may be better considered musically, rather than its common synthesis context. Similarly, the wave length of the controlling sawtooth signal can be thought to represent a musical phrase or loop.

To demonstrate this, assuming a counter is set up with a range of eight steps, and the modulo of 2 is taken from its result to switch the amplitude of a given oscillator on and off. While keeping this steady rhythmic pulse to continue, replicating the same process but with an added shift in the counter's control signal by one sixteenth of its wave length will create the simple syncopation mentioned above(see Figure 5).

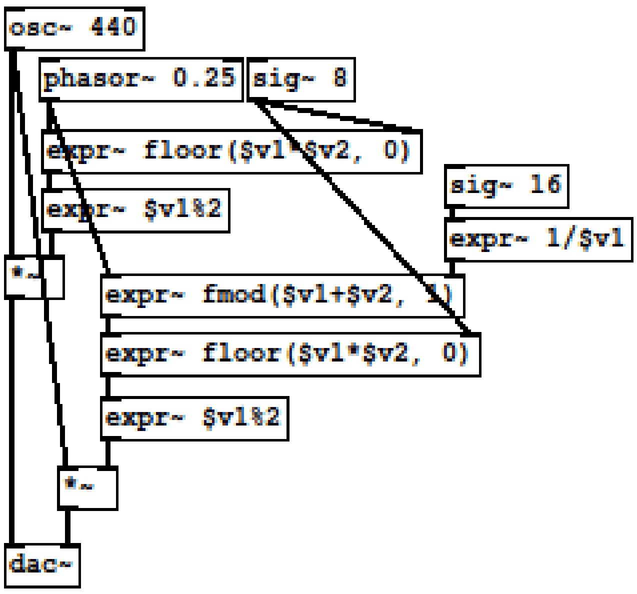

Figure 5: Phase shifting syncopation

As much as utilising the phase position is useful to vary the quantisation level in sequencing, some care must be taken to avoid any undesired effect. One of the main concerns is that, if both the original and shifted version of the sawtooth wave are used by further compositional or synthesis procedure, these signals may have issues in synchronization. As the wave period is offset, if new operation is triggered by the falling edge of an unshifted sawtooth cycle, it may abruptly end the process derived from shifted control signal. Consequently, attention needs to be invested to ensure the change in operation occurs safely for all control signals, regardless of their phase position.

2.5 Conditional test

Having the ability to produce sequential counts of varying quantisation value, as well as their corresponding sawtooth signal, the remaining task is to shape these elements into musical patterns. Two types of simple functions are predominantly involved to accomplish this: modulo operator and conditional comparison.

The main goal of using conditional comparison is to select or target specific elements within the counting series. Moreover, although more labour intensive, it may also be employed to transform one set of values to another, which can not be easily derived arithmetically. In the former case, a typical scenario would consist of targeting the last repeating phrase for different pattern treatment to achieve the "break". The latter, can be used to map a count of eight into relative MIDI values for "white notes" of one octave, thus being able to compose diatonically.

This operation can take the form of `expr~ if($v1==$v2, $v3, $v4)` in expression object, where $v1 is the signal to be tested, $v2 represents the value to target, $v3 and $v4 are the actions to be taken when the comparison returns true or false. If $v1 is a loop counter in steps of four, by having $v2 with a value of 3, it would be possible to mix two separate patterns received on $v3 and $v4, thus having a loop fed into $v3 acting as the break(see Figure 6).

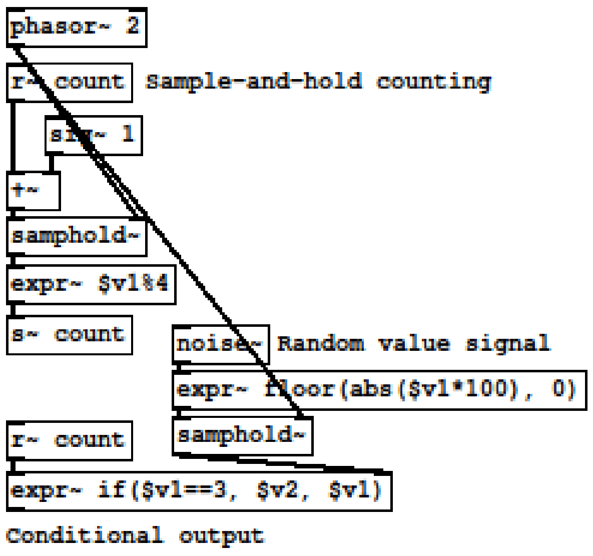

Figure 6: Using If statement to mix two counting signals

Value mapping, on the other hand, depends on nested if statement in form of `expr~ if($v1==0, $v2, if($v1==1, $v3, if($v1==2, $v4, $v5)))`. In the above expression, a count of four is mapped to values held by $v2, $v3, $v4 and $v5 respectively. The painstaking nature in devising such type of statement is obvious. Because of this, for a larger set of mapping, it would be more efficient to use a table, rather than using `tabread~` to retrieve the necessary value. However, the effect nested statement should not be overlooked, especially since each of the predicates, consequent and alternative, can be taken from modulated signals which are typically more dynamic than a simple repetitive pattern. With experimentation, nested statement can often result to surprising compositional effects, whilst maintaining a reasonable degree of clarity in implementation.

The modulo operator can also be used to derive musical patterns from a counting signal. With the previous example demonstrating syncopation, a stepping signal manipulated by modulo of 2, controls the amplitude gate of an oscillator thus producing simple rhythmic pulses. While a single modulo operator can only achieve very limited effects, the result of putting it in series one after is anything but the same. Thus, if a given counting has the range of 8 steps, by initially putting it through a modulo of 5, then again with modulo of 3 would generate a rhythmic pattern which has the accents displaced in the mixed groups of 3 and 2. Furthermore, if the value of divisor is modulated, the effect may become highly dynamic.

In expression object, modulo operation takes the form of `expr~ $v1%$v2`, where $v1 is the control counting, and $v2 is the divisor value. Additionally, as described above, if the intention is to "gate" a particular audio signal, a simple if statement which turns step 0 into 1 and all the remaining steps into 0 would be necessary.

2.6 Pulse wave

Ultimately, sequencing can be described as the ability to produce series of musical events, which may be perceived independently, as well as having common aspects responding to changes in time as a whole. In a passage of short melody, both discrete notes and their changes in pitch may be heard. In a rhythmic pattern, each single beat can be individually identified, as well as the changes in their duration in comparison to one another. In short, "notes" and "beats" are the discrete events, whereas "pitch" and "duration" are the common time varying relative properties they possess.

To a large extent, the combination of the previously mentioned techniques aims to enable such ability. By having a step counter, a measurement of time is first broken down to distinguishable parts. Through the process of conditional statement, modulo operator or even modulation, musical context can be assigned to these sub components. These assignments can be made in the frequency, amplitude or again, in the time domain.

There is, however, another approach employed to produce the effect of sequencing. Borrowed from pulse-width modulation(PWM), discrete events can also be obtained, with time varying characteristics. In doing so, it relieves the need to be predominantly dependent on the step counting , as well as bringing forth other challenging and interesting aspects in the DSP only compositional method.

Keeping the same convention of using a sawtooth oscillator as control signal, a pulse wave must therefore first be generated. To this end, a conditional statement can be employed once more to accomplish this(see Figure 7). By passing a sawtooth signal through the expression of `expr~ if($v1>$v2, 1, 0)`, a pulse wave of variable duty cycle can be delivered. In the above statement, $v1 represents the control signal, and $v2 represents the percentage of duty cycle. Since $v1 has a range between 0 and 1, 0.5 in $v2 would yield equal proportion in both the on and off state of pulse wave. Moreover, depending on the context of the resulting wave, the alternative in the if statement may hold the value of -1 or 0 for audio and control signal respectively.

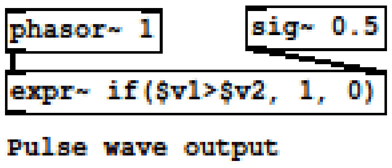

Figure 7: Pulse wave of 50 percent duty cycle

To demonstrate a simple case of “sequencing” using pulse-width modulation, the following configuration can be devised(see Figure 8). A control signal is connected to both a if statement, exactly same the one mentioned above, as well as fed through a floating-point remainder operation to scale up its initial frequency. The sub-divisional sawtooth signal is then used in place of the $v2 variable of the if expression. This results to the percentage of the duty cycle being modulated over time. To hear the effect of this, one can simply connect the PWM output to a signal multiplication object to switch the amplitude of an audio oscillator on and off. In addition, PWM signal can also be used as the trigger of sample-and-hold function, in turn to produce sequence of values that changes according to the inherent order residing in the modulation.

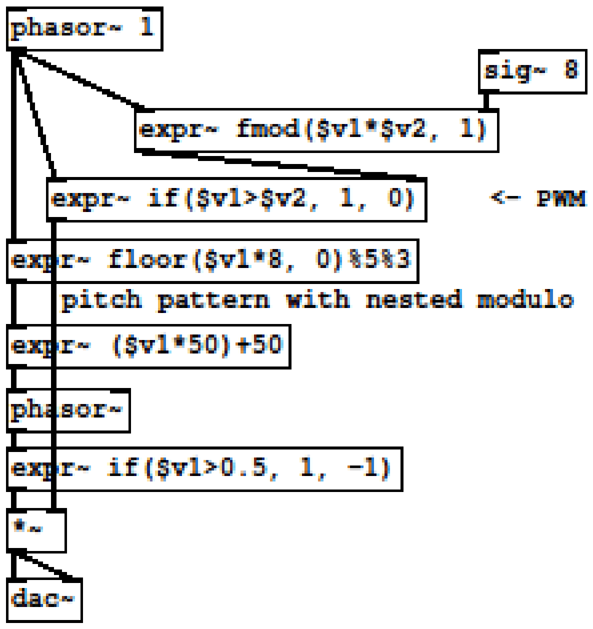

Figure 8: Amplitude gate using PWM

Compare to sequencing which relies solely on step counting, in pulse-width modulation it is generally harder to gain precise control over the exact musical pattern desired. With this in mind, PWM based method can be more spontaneous and responsive to variations in the properties of control signal. Such playful characteristics are particularly well suited for a certain type of compositional approach. Furthermore, pulse-width modulation also has great application in sound synthesis. Thus, the potential of using the same technique to both govern the musical structure, as well as sound generation, further generalises the type of compositional approach that can be achieved in the DSP domain only.

2.7 Compositional examples

Until now, the fundamental building blocks of the proposed methodology have been individually introduced and discussed. To provide a more comprehensive understanding, a series of compositions have been made exclusively with the above mentioned techniques. They not only serve as demonstrations, but also as opportunities to further explore its possible capability and limitations.

These compositions have been released in April 2011 by Gosub10[3], an Internet based music label derived from the GOTO10[4] collective. In the form of an extended play, under the title of "expr~", it is hosted by the Internet Archive[5], under Free Art license. Moreover, it is also the first musical release by the "0xA" collaboration, originally a group of artists making audio/visual performances using Pure Data. The source patches with which these compositions are generated from are also included in the release[6].

3 Conclusion

The feasibility of composing within the digital signal domain should hopefully now be evident. To arrive at the diversity in compositional tasks from a selected few common operations, can potentially be beneficial in terms of simplifying, as well as consolidating the necessary programming vocabulary and idioms. Most importantly, bringing the arrangements of musical structure and sound synthesis procedures together, with interchangeable operations, contributes to the unique attribute in this alternative framework. Even though certain aspects of its implementation may initially seem counter intuitive, once familiarized, they can lead to a rapid and flexible way of development.

However, several issues remain unaddressed due to the scope of this investigation. First, the accuracy of synchronizing multiple signals with the same control source at the sample level has not been objectively tested. While the precision in the musical outcome has been empirically sufficient, it should not be taken as proof to guarantee the sample level accuracy. Second, the influence and resistance in changes of DSP engine's sampling rate, as well as other related parameters, are not entirely clear. Lastly, the performance comparison between the DSP-only and conventional method should also be examined.

Leaving these issues aside, other possible compositional applications utilising the proposed methodology still deserve further experimentation. Although the musical examples found in the "expr~" release only make use of simple synthesis techniques as means of sound production, there is no reason to assume that the same process can not be applied to the structuring and manipulation of sampled sounds. Moreover, depending on the aesthetic tolerance, it can even be employed in the context of live coding. To this end, the author had produced several performances to specifically explore this possibility, and a variety of feedback was received.

In addition, the outlined methods were developed independently and to a large degree in an isolated circumstance. In other words, the knowledge demonstrated here was gained from practical experience, rather than through rigorous research. Hence, subsequent work should seek for an existing formal body of studies, to better contextualise given experiments, but also deepen the understanding of the subject.

This paper has hopefully brought forth an interesting alternative approach to composing music in Pure Data. Future works should be expected to address the drawbacks and problematic areas, but also further explore and extend its potential possibility.

[3]http://gosub10.org
[4]http://goto10.org
[5]http://www.archive.org/details/GOSUB10-004
[6]The source patches can also be obtained from https://gitorious.org/0xa/expr_ep