METASTUDIO: AN INTEGRATED MUSIC AND VIDEO PERFORMANCE SYSTEM FOR PURE DATA

Dr Edward Kelly

Faculty of Design
London College of
Communication
London SE1 6SB

ABSTRACT

The Metastudio is a suite of sequencing, synthesis, sampling and DSP tools for the creation of live performance systems in Pure Data. With the 0.3 release, a consistent framework has been implemented for the storage and retrieval of entire performances through a system of symbol-tagged control messages, and implementation of a set of core functions for the storage and retrieval of parameters and sequences. Metastudio and MetaVJ high-level objects with or without integrated GUIs may be linked together to form open-ended performance architectures with user-configurable state-saving systems.

INTRODUCTION

Frequently, live performance systems are created on an ad-hoc basis to fulfil a particular requirement or explore a specific idea.

While this approach is often appropriate for a particular task, the process of developing a functional system in graphical dataflow languages is frequently characterised by long development times and problematic debugging procedures. This is not a problem if the end is quite specific, such as the realisation of a composer's score, but artistic practice is often experimental. The large expenditure on time and energy in developing systems for musical creation can often act as a barrier to musical activity. Furthermore, an artist may often find himself or herself re-inventing something basic like a feedback delay, or may spend considerable time adapting a chunk of code to fit within a new framework.

Another issue with live performance systems is that material may prove to be unrepeatable, and a sonic discovery may be consigned to an audio file at best, or lost forever.

These issues are more or less relevant depending on the programming environment, and the continuing success of traditional music software is a result of the priority of an artist to make work quickly, to get ideas “down” into some form of structure or score. Within the context of live improvisation however, some environments provide a great deal of flexibility but very little recall of settings or functional elements that require a certain amount of plumbing before they become useful. This paper describes a set of objects created in one such environment (Pd-extended-0.41-4) that allow for experimental design of musical and visual performance systems quickly, and the ability to mix and match parameters and event sequences allows for improvisation with structured musical or video events as well as more common timbral twaeking and mixing.

HISTORY

The project began in 2005 as an attempt to build event sequencers capable of polyrhythmic and polymetric synchronisation in PD, and with the capability to store and retrieve sets of sequences. Another goal was to create functional audio generators and effects with a similar “patch” retrieval system and inbuilt functional interfaces, so that different configurations of sequence, generator and effect could be quickly implemented and tested.

External access to the parameters of an audio unit is important for real-time control. Up until the second release of the Metastudio, development of modules was a piecemeal process. The most recent phase of development has standardised the way Metastudio objects communicate with one another, and a system of message-tagged control routing that allows the user to record and retrieve aspects of a performance has been integrated into each object. Once recorded, a performance may be played back with “no gui” versions of the same objects, so that complex sequences of material may be incorporated into a variety of performance situations.

Finally, the standardisation of object structures and no gui versions of objects allow the creation of self-organizing polyphonic objects. Several of these are already incorporated into the latest release including a polyphonic sampler and a modular synthesis state-saving object.

SEQUENCING AND POLYRHYTHM

Metastudio sequencing objects fall into two categories – sequence generators and sequence modifiers. Sequence generators have an internal clock, and store patterns of

numbers and triggers which may be played back in sequence. The internal clock is governed by tempo, division and numeral. This allows the formulation of additive (polymetric) rhythmic sequences or subdivided (polyrhythmic) beats. Sequence generators can store an many different sequences, and new sequences are generated as copies of the current sequence so that variations may be created, but may be initialized or randomized afterwards.

Additive and Modulated Rhythms

Musically, there are two rhythmic methods for the disruption of regular, pulse-based rhythm. Additive rhythm uses a fixed number of durations as ticks (steps) based on the duration of a previous, concurrent or future ticks. When this is instigated based on a previous (simple) pulse, this fools the listener into thinking that they are listening to a rhythmic pattern based on that pulse, so the realisation of a new basis for the pulse forms a cognitive shift in their understanding of the musical pattern, and hence a shift in their perception of musical time. The polyrhythmic shift based on the subdivision of a beat (4 semiquavers or 1/16th notes in Metastudio) is another way of altering the perception of time, as a pulse based on 5 in the time of 4 retains a synchronisation with the original pulse over a larger timescale (where 4 or 5 steps happen within the same duration). The memory and sense of the original (e.g. 4/4) pulse remains, so that there is a distortion of temporal experience based on a previous experience of subdivided time. This is known as a metric modulation.

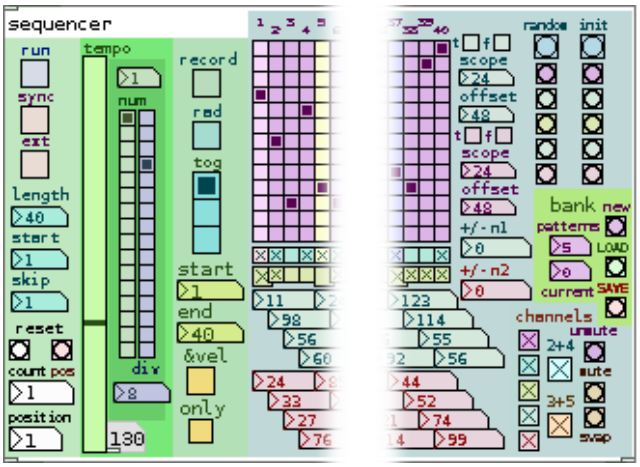

Figure 1. The two ends of A Metastudio sequencer, showing polyrhythm and sequence length controls, step input and randomization functions.

The *seq_rhythm* Sequence Modifier

Both of these methods are incorporated into the Metastudio sequencing modules, the *num* and *div* parameters allowing the user to specify additive and divisive pulse values relative to the tempo. This concept is extended using the *seq_rhythm.pd* object, which when connected with a sequencer object allows the user to specify the numeral and divisor of a sequence per-step.

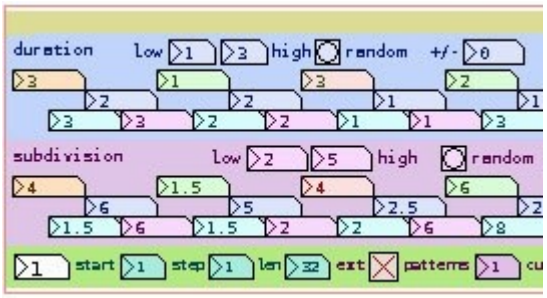

Figure 2. The *seq_rhythm.pd* sequence modifier allows different rhythmic timings per-sequence step.

Swing

Another way in which musical rhythm may be modulated is with alternate long and short beats, known as swing. The Metastudio allows the combination of swing (in the *trigseq.pd* drum sequencer) with polyrhythmic or polymetric pulses. Swing may be set either positive or negative, resulting in a long-short pattern or a short-long pattern respectively.

ANALOGUE ANALOGIES WITH SYNTHESIZED PERCUSSION

Metastudio percussion modules contain built-in variances in order that no two instances of a sound event may be the same. As a response to the notion that acoustic drums and analogue machines present a new version of the same sound to the listener on each occurrence (unlike a sample), the drum synthesizer modules in Metastudio are created with a controlled amount of randomization as to their temporal parameters.

Attack and decay envelopes are randomly variable between t=0.01x and t=2x the preset value according to the *bias* and *rand* parameters. The *rand* variable controls the degree of randomization, and the *bias* variable controls the direction of that randomization. A *bias* setting of -1 means that all timings of an envelope will be shorter than the preset value for instance.

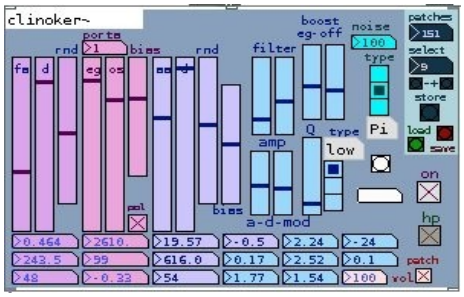

Figure 3. A Metastudio percussion module, showing a unique GUI as well as the standard patch storage

interface..

MESSAGE-TAGGED CONTROL SYSTEM

Input

For all Metastudio objects, any parameter may be accessed remotely. This is standardised for common parameters such as pitch. So, sending a *pitch $1* message with a number plugged into it[1] into the correct inlet will alter the pitch for any module with a pitch parameter. Each object possesses unique parameters also, but a set of examples is included with the help file for each module.

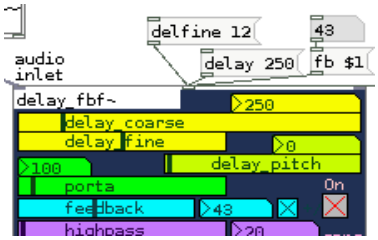

Figure 4. A Metastudio module with external control.

Output

Message tagged control parameters come out of Metastudio objects in the same format as those used to control the object. A qlist object can be used to store all parameters and patch changes of a single performance, and to play back such an instance, provided there is distinction between individual objects in the message stream (e.g. using *prepend* and *route* objects).

MODULAR SYNTHESIS WITH METASTUDIO

Whereas most of the objects within the Metastudio suite are self-contained in terms of patching – the settings for each object are stored within the object – there is another category of Metastudio objects that require a different approach. The modular synthesis objects within Metastudio 0.3 are designed not to work as discrete sound generators or processors, but as units within a functional synthesis object.

[1] PD substitutes the *$1* element for the first element of a list, or for a single float plugged into the message in which it appears. For more information, see the PD documentation.

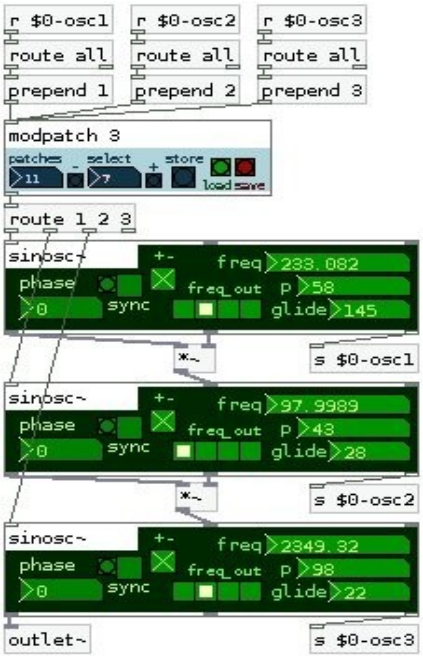

Figure 5. The modpatch object configures itself to store as many lists of settings are there are objects. The number of objects is set by its creation argument.

The modpatch object, combined with the internal structure of a Metastudio modular synthesizer, allows the user to create complex synthesis units with open-ended architecture, and to save all of the settings of the modules used to create thissynthesis unit according to how many objects are used in its creation. There are a number of modules built into Metastudio that do not have internal patching mechanisms. Examples are presented in help files for the *sinosc~* and *trisaw~* (sine, and phase distortion) objects. These objects allow fast prototyping of complex synthesis configurations, and a self-configuring module (*modpatch*) may be used to store the settings of such synthesis patches.

DYNAMIC OBJECT CREATION

Another module developed for the Metastudio is the sampler_matrix~ object. Using messages to create and connect objects within a subpatch is a useful method for the creation of patches with variable internal architectures.

Figure 6. The sampler_matrix~ object is a dynamically configurable polyphonic sampler.

The polyphonic sampler is constructed from *samplevoice~* abstractions in the same folder. These may be chained together to form polyphonic voices, and this process is used to create up to 12 samplers with up to 12 voices each. The user specifies the number of channels and voices, and bangs “make”, and they are created and linked automatically.

MIXING AND EQ

There are a number of equalization and mixing devices available in Metastudio with limiting functions built in. This avoids a common problem of feedback loops, where the loop generates a resonance so powerful it clips and distorts the signal. This is particularly an issue with the delay_fbf~ and pitchshift~ modules, where a high Q setting for the filters in the delay feedback path, or a very small amount of pitch shift, both with high feedback settings, cause resonance to quickly grow extremely loud. Such resonances can be played using a MIDI controller if the signal is limited, although they are still so powerful that they silence all other material when plugged stright into the mixer. Limiting is built into several of the mixer units, but may be used with a *mix~* object (a simple gain control with a limiter) in order to minimize this effect.

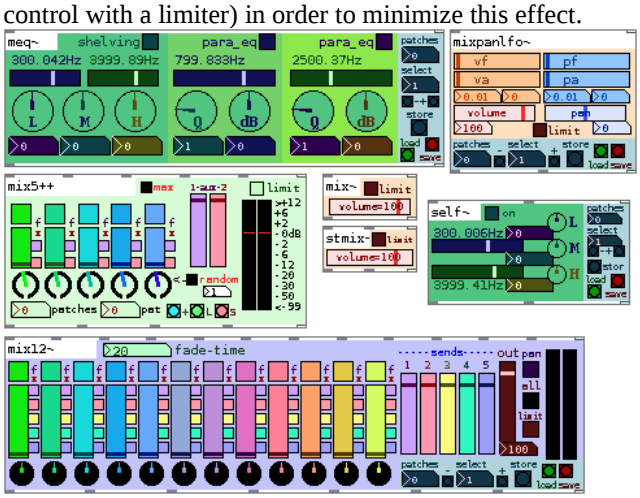

Figure 7. Some of the mixers and equalizers available. Many have limiters, and some sophisticated methods for constrained randomness are implemented in the larger mixers.

METAVJ

The philosophy of Metastudio has been expanded to incorporate pdp video objects in the MetaVJ. Gui objects for the playing, processing and mixing of video sources exist as self-contained objects. These can be configured to work with the modpatch object, so as with the modular synthesis objects, a modular video processing system can be created with state-saving features. Although PD has the capability to run both video and sound in a single application, it is generally more reliable and stable to run one or the other. Although PD is not multi-threaded (and hence cannot use multiple processors) modern dual-core computers will assign a second instance of PD to the other processor core of the computer. Thus it is possible to run both video and sound software on a single computer without running the risk of one application taking too much CPU for the other to work. This allows for the creation of a full audio-visual system of performance. Netsend and netreceive objects may be used to communicate between the two systems, and so synchronisation and control parameters may be passed from one to the other.

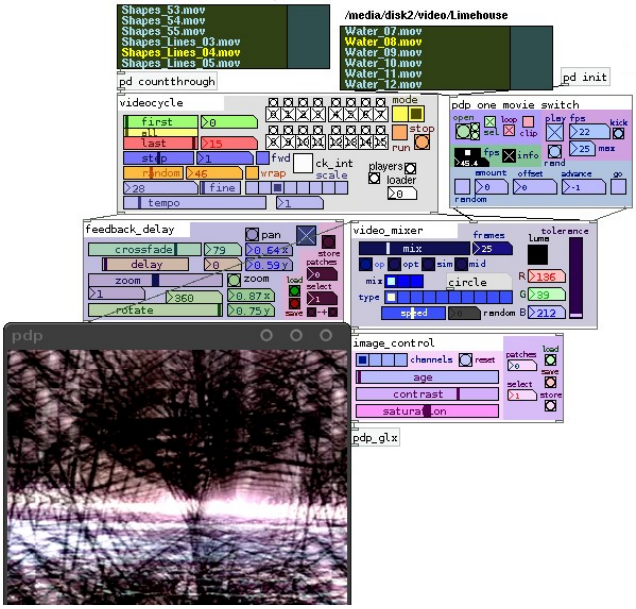

Figure 8. A MetaVJ Patch

REFERENCES

[1] Kelly, E. “Time in Music: Strateiges for Engagement,” PhD Thesis, *University of East Anglia*, Norwich, Uk, 2005.

[2] Kramer, J. “The Time of Music: New Meanings, New Temporalities, New Listening Strategies,” *Schirmer*, New York, USA, 1988.

[3] Puckette, M. 1996. "Pure Data: another integrated computer music environment." Proceedings, Second Intercollege Computer Music Concerts, Tachikawa, Japan, pp. 37-41.