Metastudio: An Integrate Music and Video Performance System

Synchroma
Edward Kelly
Camberwell College of Art
254 Camberwell New Road
London SE5 0RP
+44(0)784 641 2891
morph_2016@yahoo.co.uk

ABSTRACT
The Metastudio for Pd-extended 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.

Keywords
Sequencing, Interface, Modular, Music, Video

1.INTRODUCTION
Frequently, live performance systems are created on an ad-hoc basis to fulfil a particular requirement or explore a specific idea. This approach often yields interesting results, but the PD programmer 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.

The Metastudio is a project designed to address these issues. Performance systems created within the Metastudio environment have integrated state-saving mechanisms, and sequences and settings may be stored and retrieved in performance, and collections of patch-settings or sequences may be loaded and save from files.

The MetaVJ implements the concept of Metastudio for pdp video processing and playback modules. Video performance systems in MetaVJ may be synchronized to and connected with the Metastudio to create complete audio-visual performance systems.

PdCon09, July 19-26, , 2000, São Paulo, SP, Brazil.

2.History and Philosophy
This project began in 2005 as an attempt to build a drum machine capable of polyrhythmic and polymetric sequences, and with the ability to store and retrieve patterns and settings. Another goal was to create self-contained objects with functional graphical user interfaces (GUIs). Improvements to the Graph-On-Parent (GOP) functions introduced in PD 0.38, the implementation of GUIs on patchable objects made it possible for the Metastudio modules to evolve. With the addition of new modules for synthesis, processing, mixing etc. it has evolved into a complete performance system. The present incarnation incorporates a number of new features that have been devised in order to integrate individual machines into a complete system. One of the goals of Metastudio is to create objects that are quick to incorporate into a PD patch while expanding the sonic possibilities of regular synthesis, processing, sequencing and mixing objects. Another key principle of development is to integrate the storage and retrieval of settings and sequences within each object, so that ideas may be re-used and revisited to at a later date.

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 every aspect 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.

3.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 unlimited number of different sequences, and new sequences are generated as copies of the current sequenc so that variations may be created.

3.1 Multiplicity in Musical Rhythm

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.

Both of these methods are incorporated into Metastudio sequencing modules, the *num* and *div* parameters allowing the user to specify additive and divisive playback speeds relative to the tempo. This concept is extended using the seq_rhythm object, which when connected with a sequence object facilitates sequences with uneven beat durations, different multiples of basic values and individual subdivision settings per-sequence step.

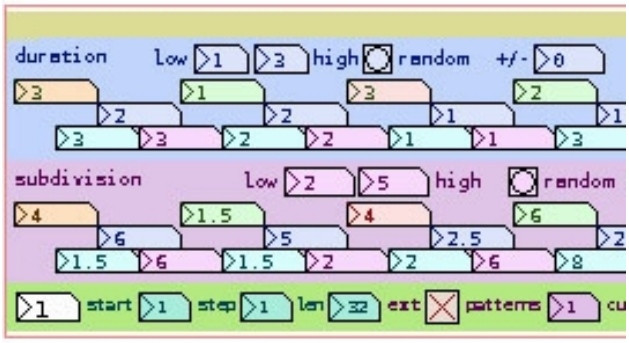

Figure 1. seq_events.pd allows different settings per-step for rhythmic timings.

3.2 Swing

Another way in which musical rhythm is modulated is with swing, where even and odd beats are longer or shorter than each other. PD allows the combination of swing with polymetric or polyrhythmic pulses, and this feature is implemented in the trigseq abstraction. Swing may be either positive or negative, where positive swing results in a long-short pattern and negative results in a short-long pattern.

4.Analogue Analogies with Synthesized Percussion

Metastudio clicker~ chlicker~ and clinoker~ 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 analogue machines present a new version of the same sound to the listener on each occurrence (unlike a sample), the drum synthesizer modules in Pure Data are created with a controlled amount of randomization as to their temporal parameters.

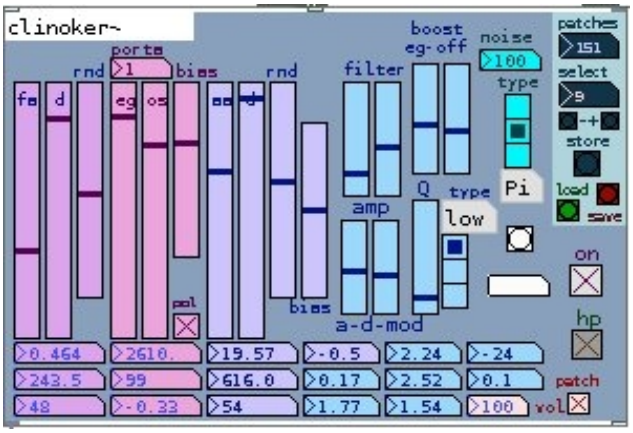

Figure 2. A drum machine synthesizer.

Attack and decay envelopes are randomly variable between t=0.01x and t=2x the preset value according to the bias and rand variables. The rand variable controls the amount 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, whereas a bias setting of 0 means that they will all be longer. There are three units with this feature. Clicker~ has a simple sine wave generator, with attack and decay envelopes for the amplitude and frequency. There is a frequency offset, and the frequency envelope may be inverted. Chlicker~ has also a harmonic or multiple of the basic frequency, and the amplitude envelope for this is set as a multiplier for the basic amplitude envelope. This means that all envelopes are synchronised in terms of their relative timings and randomizations.

The clinoker~ unit has a noise generator and a state-variable filter built into it. Envelope parameters are derived from the envelope frequency of the oscillator and the amplitude envelope as with the chlicker~ module, so that they accurately track the modifications to the basic envelopes. The noise generator can be defined as a white, pink, or 3-operator FM synthesis generator, and FM spectra are randomly generated but are stored with the patch settings. 3-operator frequency modulation synthesis (op1 modulates op2, op2 modulates op3, op3 → out) is a way of introducing a spectrum of frequencies that is deterministic, yet unpredictable. This is achieved by sending the object an rspect command, or by banging the corresponding button. The frequencies of the fm operators are stored with the rest of the patch settings if so desired, and the "warp" factor (fm index) is also stored. Thus, novel percussion sounds can be generated at will, in performance time. These can be stored for later retrieval, so that a body of work can evolve from performance to performance.

5. Message-Tagged Control System

For all Metastudio objects, any parameter may be accessed remotely. This is standardised to some extent – e.g. pitch is always accessed by sending the correct inlet a [pitch $1(message, where $1 is substituted (by PD) with any float that is entered into the message object. However, each object possesses special parameters of its own, and so there are only so many parameters that may be standardised.

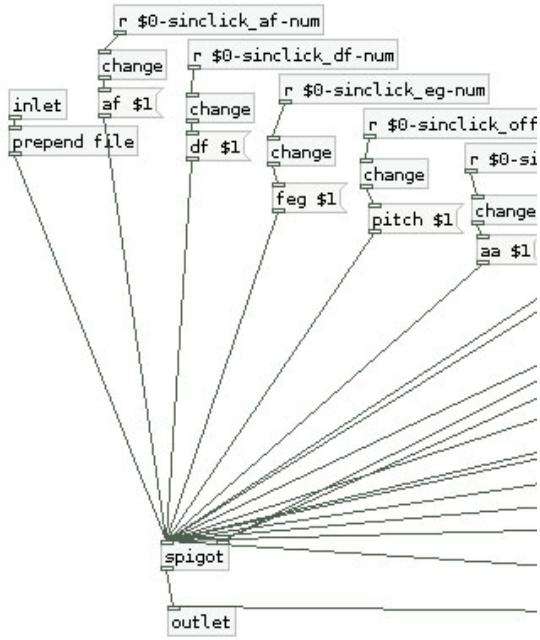

Figure 3. Parameters are sent out of Metastudio objects in the same format as those used to control them.

Message tagged control parameters come out of Metastudio objects, and they are 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 (using prepend and route objects). The inlet for control messages is always the same for each type of object, since the standardisation of inlets is always trigger/pitch (or signal), control, signal-based-modulation. Change the object, and the inlets remain the same.

5.1 Internal and External Control

Every parameter of a Metastudio object is externally controllable. A parameter name combined with a value in a list is enough to change the corresponding parameter of the object. The names of parameters to be influenced is given in a help file for each object. Whereas most of the objects within the Metastudio suite are self-contained in terms of patching – the settings for each object are stored within itself – there is a new category of Metastudio objects that requires 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. 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 this synthesis unit according to how many objects are used in its creation. It is the first of the self-organizing discussed here.

6. Modular Synthesis with Metastudio

There are a number of modules built into Metastudio that do not have internal patching mechanisms. These are devised so that user-defined synthesis systems may be implemented, although there are help-files for some of these objects that give ideas as to how this may be achieved. Simple frequency modulation synthesis (without the 0 frequency phase modulation of Chowning FM a la Yamaha DX7) is presented in a help file for the wrapped sine oscillator. These objects allow fast prototyping of complex synthesis algorithms, and a self-configuring module (modpatch) may be used to store the settings of such synthesis patches.

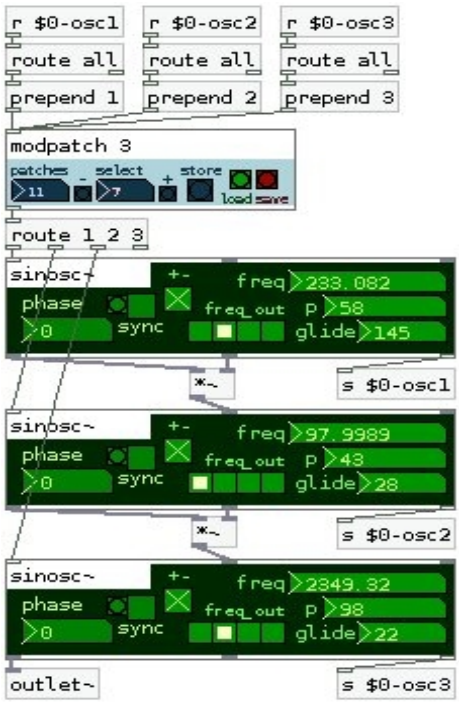

Figure 4. The modpatch object configures itself to store as many lists of settings are there are objects. The number of objects is set by its creation argument.

Through the use of various PD-extended modules there are three Metastudio oscillators – sinosc~, pwosc~ and trisaw~. While the first two are quite straightforward (a sinewave and a pulsewidth oscillator) the trisaw~ object is an oscillator that morphs between a triangle waveform and a sawtooth waveform. The waveshape modulation objects (pwosc~ and trisaw~) can produce complex phase-distortion like spectra, and audio-rate modulation of the waveshape is possible, creating rhythmic variations in the sound when modulation frequencies are close to harmonics of the fundamental. These objects combine with the traveler~ filter object to form user defined synthesis objects, and the modpatch object allows the parameters of mid-level objects such as the oscillator, filter and envelope objects to be stored in an object with a user-defined number of storage entities. This allows the user to create complex synthesis creations without having to specify the number of modules used until the modular synthesizer is created and working.

7.Self-Organizing Polyphonic Sampler

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. The polyphonic

sampler is constructed from samplevoice~ objects 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.

Figure 5. The sampler_matrix~ is a self-configuring polyphonic sampler.

The sampler_matrix~ uses global variables to get lists of note data into the object, which is unfortunate since there can only be one of these in a patch. Future releases will use iemguts to create inlets in the master patch, and link them to the newly created machines. However, Metastudio is designed for use with the latest stable pd-extended release, so this is a future development.

8.Mixing and Processing

There are a number of processing and mixing modules employing the Metastudio philosophy. Each object has a patch-storage core, so the settings of each object may be stored and retrieved by the user, and message tagged control parameters allow the user to manipulate parameters of an object from outside – by a sequenced parameter or by real time manipulation (e.g. with a MIDI controller).

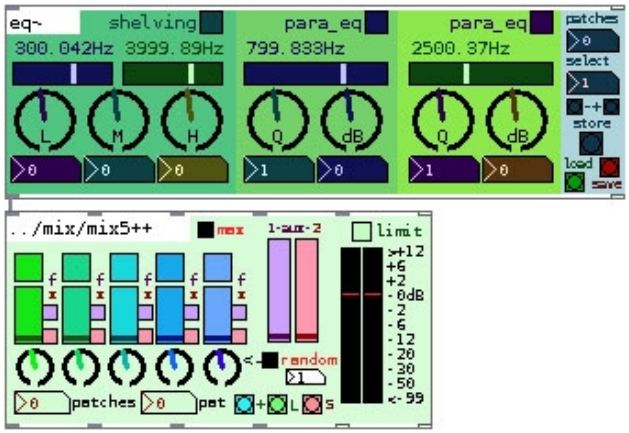

Figure 6. Objects for equalization and mixing. The mix5++ object has a built-in limiting function.

There are a number of equalization and mixing devices available to the user in Metastudio, and several of these have limiting functions built in. This avoids the 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, causes a resonance to quickly grow out of control. 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. Limiting is built into several of the mixer units.

9.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 GOP 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.

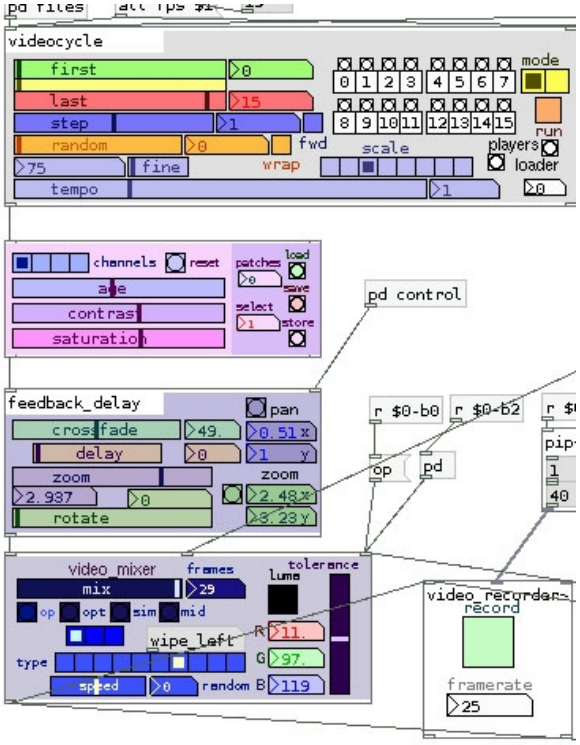

Figure 7. Some of the MetaVJ objects for video processing.

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.

10.Conclusion

The Metastudio is a versatile suite of software that allows the user to create performance situations quickly, and so ideas may be explored in a creative manner without the need to redefine common objects or techniques continually. It does not attempt to

address every creative idea, but provides a set of tools that the user can deploy quickly with internal state-saving mechanisms. Ideas realized with the Metastudio may be stored as patch and pattern files, so that ideas may be revisited and reworked at the leisure of the user.

One of the strengths of the Metastudio approach is that it allows the user to quickly create new configurations of modules to experiment with ideas about sound and video processing and generation.

Finally, the self-building objects point to a new development in object design. The release of the iemguts objects by Johannes Zmoelnig in 2008 opens up the possibility of creating objects that build their own GUI and internal structure according to creation arguments, since it is now possible to send object creation and manipulation messages to the master patch. Sequencers with arbitrary numbers of steps, randomly configured synthesizers and many other possibilities are available. However, the Metastudio is designed to work with Pd-extended, so that it may be used by non-experts either in educational environments or artistic projects. One principle of releasing such a set of tools is that it should work with the latest freely available Pd-extended release (0.40-3 at the time of writing) and so such developments will be held back from public consumption until a new release of Pd-extended incorporates these objects.

11.REFERENCES

[1] Kelly, E. *Time in Music: Strategies for Engagement.* PhD Thesis, University of East Anglia, Norwich, UK, 2005.

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

[3] Zmoelnig, J. *Pure Agents: Augmenting Live Coding.* Piksel 08, Bergen, Norway, 2008.