A Protocol for Audiovisual Cutting

Nick Collins, Fredrik Olofsson
sicklincoln.org, fredrikolofsson.com
e-mail: nick@sicklincoln.org, f@fredrikolofsson.com

Abstract

We explore the extension of an algorithmic composition system for live audio cutting to the realm of video, through a protocol for message passing between separate audio and video applications. The protocol enables fruitful musician to video artist collaboration with multiple new applications in live performance: The crowd at a gig can be cutup as video in synchrony with audio cutting, a musician can be filmed live and both the footage and output audio stream segmented locked together. More abstract mappings are perfectly possible, but we emphasise the ability to reveal the nature of underlying audio cutting algorithms that would otherwise remain concealed from an audience.

There are parallel MIDI and OSC realtime implementations and text file generation for non-realtime rendering. A propitious side effect of the protocol is that capabilities in audio cutting can be cheaply brought to bear for video processing.

Keywords: Realtime audio cutting, video processing, VJ techniques

1 An Introduction to VJing

A laptop musician at a contemporary club gig is most likely accompanied by a VJ, whose video manipulations underscore or contrast the aural interest. It's recent fashion and accessibility is shown by dedicated software like VJamm, Isadora, Aestesis and Arkaos VJ. Flexible coding environments are also available, the GEM extensions for PD, the Jitter and nato.0+55 extensions for MAX/MSP, CsoundAV and the DIPS library. There is an online community, evidenced by audiovisualizers.com, or the lev (live experimental video) mailing list to name but two.

The past decades have seen the rise of scratch video as an artform (Snider 2000), with visual artists emulating the liberating cut and dice games of hip hop producers. Witness Max Headroom's stuttering, Coldcut's 1997 Let Us Play album, Qbert's Wave Twisters (2000), and Addictive TV. A new product (www.cycling74.com/products/mspinky.html) makes commercially available a sync scratching tool. To quote Hart Snider (Snider 2000) 'people appear to dance to the beat of the music when a VJ is scratching the motion of samples'. It is this capability that intrigues us for automated breakbeat cutting.

Though VJs can be found working all evening in the background over many musician's sets, new acts are forming as audiovisual collectives from the outset. Naturally, one of the prime media for audiovisual experimentation and interaction is computer games. The first author recently witnessed a wonderful presentation by Frankie the Robot DJ (frankietherobot.com), an animated DJ character built using a 3D games engine. The team behind Frankie is committed to accessible work for a club audience, dancing with PlayStation controllers in hand rather than bending over laptop screens.

Film music is the dominant critical language, but an interesting alternative is given by Nicholas Cook's model of multimedia (Cook 1998). Indeed, for VJing we might consider Cook's music film rather than film music as an appropriate model. Snider notes 'the music always takes the lead in the relationship between visuals and music at a show'. Cook categorises three basic interactions, termed conformance, complementation and contest. He takes the view that 1-1 mappings (conformance) are not true multimedia, but other authors and many contemporary music videos highlight the potency of mickey mousing (Birtwistle 2002: particularly p25, Mick Grierson's audiovisual cutup machine Ariel).

A word of caution: VJing for electronica might be taken as an admission that the visual side of laptop musical performance is woefully boring. Yet the distraction of the audience by the strong visual medium is evident at some gigs. The VJ can find themselves in the reverse situation to the film music composer, trying not to overpower the audio.

2 Linking Audio and Video

In standard VJ practise, tempo tracking and spectral band energy following of the output audio allows the video artist to utilise gestures synchronised to the music. Visualisers like Gforce, Whitecap or even iTunes tend to work as fancy three dimensional graphic equaliser displays. Inner compositional details of the music are not accessible, since the tracking acts upon the finished product. Further detail must be explicitly passed to the video renderer. Since visual material provides a useful accompaniment to complex compositional systems in terms of marking the algorithmic composition routines underlying them, we wanted to explore the possibility for live revelation of automatic audio cutting routines possible through visual stimuli and correspondence. Further, the techniques of audio cutting can be cheaply implemented in parallel for the video processing domain through an appropriate map.

This paper primarily considers the club situation where the feed of information is from audio artist to visual artist, but the message passing could of course be two way, with auditory algorithms utilising information about the video processing. We focus on the live performance situation.

SuperCollider 2 (McCartney 1998) is a programming language for real-time audio, perfect for live audio synthesis and algorithmic composition. An extension set called the BBCut Library (Collins 2002) is dedicated to real-time rhythmic audio splicing and the investigation of cutting algorithms. It contains many procedures emulating styles of drum and bass and jungle, Aphex Twin style fast cut repetitions and stranger routines. In a new project, in collaboration with Fredrik Olofsson, the potential of this library for both visual and combined audiovisual cutting is being explored. This paper describes the messaging protocol from SuperCollider to external video applications developed for that task, and the creative possibilities that result. Since BBCut supports research into audio cutting, and the aim of scratch video is to pass on audio cutting tricks into the visual domain, this project is within the remit of scratch video art.

BBCut sends messages following a defined protocol via MIDI or OSC (Open Sound Control, Wright and Freed 1997) to a video application in the background on the same computer or on a dedicated machine. The authors have investigated the use of MAX/MSP with jitter and nato on the Mac, as well as custom C++ OpenGL software on a PC, but the receiver could be implemented in many other applications like director, flash or custom video software.

In the remaining sections we look at the protocol's technical specification and the sender realisation within SuperCollider and BBCut, the various receiver implementations across a variety of realtime video processing software, before going on to discuss some of the creative possibilities of audiovisual cutting and the results of live experiments with the system.

3 The Protocol

We are not aiming to cover general information passing between any SuperCollider patch and a video application, believing this to be too general a task and that further context is important in refining designs. Instead the protocol is set up with the restricted goal of communicating the internal states of BBCut objects to promote their use in video manipulation. We do not attempt to pass the hidden variables of individual cutting routines, but the rendering state of the BBCut hierarchy during performance. This is sufficient however to support a number of interesting compositional applications, detailed in a later section.

The protocol is illustrated here with a bias to the MIDI output version. The messages match BBCut rendering states, as explained in the next section.

Message	Values	MIDI cc numbers	Description
tempo	float, 0.25-10	2/34	beats per second
phrase length	float, 0.0–60.0	4/36	in seconds
source length	float, 0.0-60.0	6/38	in seconds
block length	float, 0.0-16.0	8/40	in seconds
phraseprop	float, 0.0-1.0	9/41	proportion of phrase completed at this time
offset	float, 0.0-1.0	10/42	offset (read) position into source. Vital for media correspondence in live cutting
isroll	boolean, 0/1	11	a flag set by some cut procedures
repeat length	float, 0.0-16.0	12/44	current atomic cut length in seconds
repeat number	integer	13	up to 128 repeats per block
block prop	float, 0.0-1.0	14	rough measure of position
amplitude	float, 0.0-1.0	16	amplitude of current repeat
synthparam	float, 0.0-1.0	any spare	arbitrary messages with values already mapped to 0.0 to 1.0

A number of wasteful, redundant or aesthetically unnecessary messages like phrase number, block number, source index and an on/off flag were removed from a first draft. The receiver can infer some parameters from others, keep it's own counters, and a cutter being on or off is indicated pragmatically by whether it sends messages or not!

Decisions were made about using only a single 7 bit controller message where practical in order to cut the transmission bandwidth. It was found useful to concentrate on the core messages above, rather than push too hard to pass many messages that would inevitably be ignored in favour of manageable mappings.

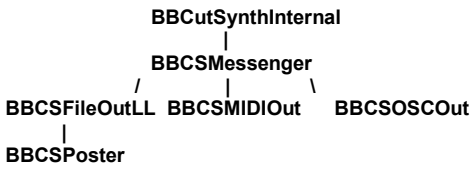
Different implementations can pass the information in different ways. The table here reflects the choices of the MIDI implementation, which is the most practical one for live work, since OSC in SC2 exhibits too much network jitter and delay under Mac OS 9. File writing and OSC can cope with writing any parameter value within the type specification. It was felt that at least a draft of the OSC version should be made since this will prove critical for porting to OS X and SC Server, and allow one to escape from the vagaries of MIDI.

All the MIDI messages are control change, being easy to implement and cheap to send. The protocol has been revised after testing in live performances, and to avoid MIDI overloads and demands on the receiver, kept to a relatively restricted form sufficient for the artistic ends discussed below. The volume of MIDI messages sent is around half the maximum MIDI bandwidth, for three simultaneous cutters running at 180 bpm with virtuosic rolls and stutters.

4 Passing the Rendering State of the BBCut Library

The current performance implementation takes advantage of the internal structure of the BBCut Library (Collins 2002); indeed, piggy-backing the internal message passing allowed us to set up an effective protocol remarkably cheaply. Further revelation rests on an understanding of the phrase/blocks/repeats paradigm of the BBCut Library. This is a simple hierarchy of cutting, with the phrase at the top level, subdivided into blocks that utilise a particular offset into the source. The blocks are in turn subdivided into atomic repeats (stutters, cuts) at that common offset. BBCut messaging for rendering follows this breakdown closely, synthesising by the repeat, and updating with each new block or phrase. This is a live protocol, so the pertinent message is sent at the point the information is available, that is, it depends on the time decision making is communicating internally in BBCut, at the onset of new phrases, blocks or repeats, plus an external transmission time.

It is not practical to go into full details of the structure of BBCut here but a portion of the class tree will be helpful to see how the existing rendering engine of BBCut, as represented below by the BBCutSynthInternal class, is adapted:



These BBCutSynth classes allow one to build in the external message passing of the protocol into any existing BBCut code, with simultaneous use of posting, file out, MIDI and OSC as desired. Each independent cutter can be tagged with an instance number so that independent streams of cutting information are passed to the external application. The instance 0 is reserved for global messaging, like global tempo and sectional markers.

As a brief sideline, the file out class stores data into a linked list while the Synth is running, so as to avoid file writing conflicts when recording the Synth output. The data file is written once synthesis stops.

The following SuperCollider client code should give an idea of how the protocol works in practise. Adapting existing BBCut code is just a question of nesting a BBCSMessenger derived object into the cutsynth chain.

```
//demonstration of client code for a single cutter that
//transmits protocol messages via MIDI and to a file

Synth.record({
var cutsynth, midiout,ll;
midiout= MIDIOut(7); ll= LinkedList.new;
cutsynth= BBCutSynthParam.easyain(AudioIn.ar(1));
cutsynth=BBCSMIDIOut(cutsynth, 1,  midiout );
cutsynth= BBCSFileOutLL(cutsynth, 1, ll);

BBCut.ar(cutsynth)
})
```

Simultaneous cutters share a midi out, linked list or OSCPort object rather than wastefully create their own instances, hence a MIDIOut, LinkedList or OSCPort object has to be passed into an argument of the *new class method.

Since the protocol was developed, it has also been used to pass messages from non BBCut SuperCollider code. This is easy to do simply by creating a dummy BBCSMIDIOut object or similar, and explicitly calling the instance methods of the class, rather than letting the BBCut structural code call it for you.

5 Video Implementations

Prototype systems for audiovisual cutting were developed independently by the two authors. Collins used MIDI messaging from a Mac to a PC running custom OpenGL + AVI video code. Olofsson implemented on a single Mac an internal MIDI stream from an early breakbeat cutting algorithm by the first author to MAX/MSP plus nato.

The current live implementation uses SuperCollider 2 sending MIDI to a MAX/MSP engine with both nato and jitter video processing. Receiving data in the video program is a decoder subpatch that converts back 14 bit midi into the 0.0-1.0 floating point range. The data is then outputted both via standard Max subpatch outlets and as send/receive pair variables for use in remote parts of the main video program. For controlling OpenGL objects parameters are usually scaled to 0-360 degrees and for other purposes, like the offset parameter mapped to number of frames in a captured video buffer, individual scaling is easily accomplished. With a special patch, text file output from SuperCollider can be imported and decoded in non-realtime. This allows for high rendering quality writing of movies to disk.

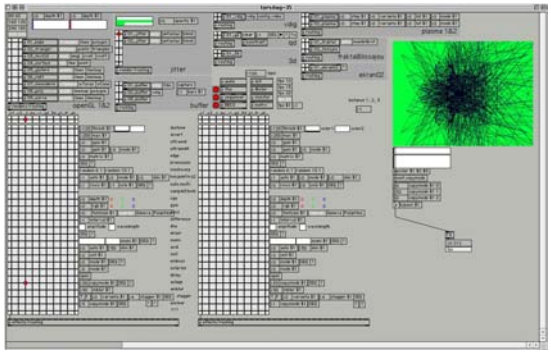


Figure 1 The main video processing patch

The main control page of the live video program is shown in Figure 1. Alongside other controllers like LFOs, audio analysers and a step sequencer, it allows for very interesting ways of syncing control of video source and effect parameters to music. The syncing features can be muted to avoid overuse tiring the eye.

The first author has demonstrated the control of very large flocks of OpenGL objects by writing C++ OpenGL applications as receivers on a home desktop PC, but has yet to take this live.

6 Live Rendering Possibilities

The motivations for a mapping supporting synchronised audio and visual cutting primarily included the cutting of live video streams in synchrony with audio stream cutting. A dancing audience on a live video feed can be cut and projected synced to the rhythmic events they hear (see Figure 2 for an example). A live drummer or other musician playing to a click track can have their music automatically cut whilst video of their performance is similarly split. Since BBCut supports multiple simultaneous tempi, this extends to acting upon an instrumental ensemble even in the case of complex independent parts.

Rather than cutting live streams, where one only has access to events which have already occurred, it is possible to use random access in the synchronised cutting of any prepared audiovisual material. The BBCut Library is already set up to deal with both the situation of live streams and finite buffers in memory.

We are not restricted to video processing but may utilise the protocol as a guide to any live image rendering. For instance, OpenGL 3D graphics can be triggered from the cutting protocol. Any visual display can react to timing and other rendering information reflecting audio cutting algorithms. In practise, more than three independent objects tracking cutters was found visually confusing, so most demonstrations tended to use a limited number of message passing instances.

In live tests, the most effective displays of synchronisation were found to be for slower paced cut sequences. The frame rate had to be sufficiently high to track fast audio rhythms without aliasing. For a typical 30fps rate, only rhythmic events up to 15Hz, corresponding to cut sizes of just larger than demisemiquavers (32nd notes) at 120bpm, were useful. In practical terms then, the VJ may be required to filter some of the faster rhythmic events into sensible video rate triggers.

In control of OpenGL objects, useful settings were found by equating brightness and object size to the repeat number parameter, so that swarms of objects exploded alongside the aurally exciting fast rolls.

7 Conclusions

Music software locked laptop performance is rife with the difficulties of conveying interaction and generative composition, though sometimes a laptop performer might project their screen display to give some clues to the audience of the profundity of their actions. We believe though the potential for demonstrating tangible realtime linkage between simultaneous audio and video allows far more audience comprehension of the new automatic cutting technology.

The authors have undertaken live performances using this technology with an inevitable learning curve but some encouraging success. A live performance sees laptop performers Sick Lincoln and f0 collaborate in a fusion of algorithmic breakbeat cutting, SuperColliding electronica and complex synced visuals. All of the situations under the live rendering possibilities section above have been demonstrated and are being further explored in practical contexts. Rather than refining the technical side, the authors now believe themselves to be in the position of refining their mappings aesthetically.

There is a selection of quicktime movie demos at http://www.fredrikolofsson.com/video.

References

Birtwistle, A. 2002. “Insects, Urine and Flatulence: the radical potential of mickey mousing.” *Proceedings of Cybersonica*, Institute of Contemporary Arts, London, June 5-7, 2002. pp 23-35.

Collins, N. 2002. “The BBCut Library.” *Proceedings of the International Computer Music Conference*, Goteborg, Sweden, September 2002. pp 313-316.

Cook, Nicholas. 1998. *Analysing Musical Multimedia*. Oxford University Press, New York.

McCartney, J. 1998. “Continued Evolution of the SuperCollider Real Time Synthesis Environment.” *Proceedings of the International Computer Music Conference*, Ann Arbor, Michigan, 1998.

Snider, H. www.artengine.ca/scratchvideo/indextoc.html. Online master’s thesis. Checked 28/2/03.

Wright,M. and Freed,A.1997. “Open Sound Control: A new Protocol for Communicating with Sound Synthesiers.” *Proceedings of the International Computer Music Conference*, Thessaloniki, Hellas, pp 101-104.

Figure 2 A sequence of 30 frames showing breakbeat style offsetting within a captured video sequence