Chroma

Newsletter of the Australasian Computer Music Association, Inc.

PO Box 284, Fitzroy, Victoria, Australia 3065
Issue Number 37
March 2006

ISSN 1035-8271

Editorial

Adelaide is a place name that is increasing in use in regards to computer and electronic music. A number of ACMA members are currently involved with Project 3, and there is an entire Australasian Computer Music Conference scheduled for a few months time. It is good to see activity in Adelaide. It will be even better to go there and experience it live. My next project after this edition of Chroma is to write up an article for the Adelaide conference of ACMA. April the 18th is the date to keep in mind for submissions, so after you read this edition of Chroma, go and submit something for ACMC 2006, and I will see you in Adelaide!

Chroma 37 exhibits the latest workings of Impromptu, a fast growing musical environment created by Andrew Sorensen. John Rimmer shares his wonderful experiences of computer music, and Gordon Monro reminds us of all the highlights from ACMC 2005, just to invigorate us to prepare for the next conference. The submission details for the conference are also included, to help you plan for the upcoming conference.

Enjoy!

Timothy Opie

Presidents Report

As occurs in every year, the major event that ACMA holds is the national conference. These events are very significant in that the casual networks that exist throughout Australia and New Zealand between ACMA members (and the wider community) are made apparent. All at once you can feel connected to ideas, activities and technologies that for most of the year seam a little distant. Even though the internet has brought information, and to some extent communication, from the wider world closer to us, we can still feel the isolation that comes with living in our part of it. I suspect that the reason that the internet does not wholly cure this is that information without the context of actual human interaction is not as meaningful or useful. I think this is why many ACMA members find the conference so useful and motivating.

There is a particular reason why I am interested in the conference this year, and that is because it is the first time that it will be held in Adelaide. When I first joined ACMA I was studying in Adelaide and a lone South Australian member. The studio that Tristram Cray set up at Adelaide University was at that time in moth balls and things where not looking good for computer and electro-acoustic music in that part of the world. Given this I am very pleased to see that gradually the activities in computer music have developed and that once more there is an ongoing culture based around the Elder Conservatorium.

So I'm looking forward to ACMC 06 and I would encourage all ACMA members to make the trip to Adelaide.

Peter Mcilwain

The ACMA Committee

President: Peter Mcilwain
Peter.Mcilwain@arts.monash.edu.au
Vice President: Lissa Meridan
lissa.meridan@vuw.ac.nz
Secretary: Greg Schiemer
schiemer@uow.edu.au
Treasurer: Ian Kaminskyj
ian.kaminskyj@eng.monash.edu.au
Publications Officer: Timothy Opie
timopie@fastmail.fm
Promotions Officer: Robert Sazdov
r.sazdov@ecu.edu.au
Web Officer: Peter Mcilwain
Peter.Mcilwain@arts.monash.edu.au
E-list moderator: Lissa Meridan
lissa.meridan@vuw.ac.nz
Membership Officer: Robert Sazdov
r.sazdov@ecu.edu.au
Public Officer: Paul Doornbusch
mailto:pauld@koncon.nl
QLD Representative: Andrew Brown
a.brown@qut.edu.au
ACT Representative: Alistair Riddell
alistair.riddell@anu.edu.au
SA Representative: Christian Haines
Christian.haines@adelaide.edu.au

The views expressed within are not necessarily those of the Editor or of ACMA, Inc.

Contents

Going to Adelaide?

Going to Adelaide?

Going to Adelaide?

Australasian Computer Music Conference
Medi(t)ations: computers, music and intermedia

11 - 13 July 2006
Elder Conservatorium of Music,
University of Adelaide
Adelaide, Australia

Composer Profile

John Rimmer

"Your other compositions will never be the same again now that you have begun electronic music composition."

When Gustav Ciamaga, director of the Electronic Music Studio, University of Toronto uttered these words to the graduate electronic music class in 1967, I never realised their full implication. To me, electronic music was a mysterious, new creative art kept in a separate box marked 'Handle with Care'.

Looking back many years later, I can now see what Ciamaga meant. Composers have found that the experience of composing with electronic media has undoubtedly influenced their approach to composing with the more conventional forces; voices, and instruments. One becomes highly sensitised to all sounds and in my case particularly those of the natural environment.

For me the electroacoustic medium provides a powerful way of expressing 'New Zealandness' in musical sound. This is achieved when one uses the sounds of the local environment as sound sources. Many of these sounds are not unique to this country but some are. In using them we subconsciously relate our work to the land and the sea that surrounds us.

I established an electronic music studio at the University of Auckland when I began teaching there in the mid-1970's. At that time the only major studio in New Zealand was at the Victoria University of Wellington where Douglas Lilburn had established one about ten years before. I worked in Lilburn's studio over a ten year period mainly in term vacations. This was the decade before small computers were used to record and shape sounds and therefore reel to reel tape recorders with tape splicing, microphone, mixer, ring modulator, filter and synthesizer were the norm.

On University study leave in 1979, I attended the computer music summer course at the Massachusetts Institute of Technology along with fellow Australasian Graham Hair later to become professor at Glasgow University. The director of this course was expatriate New Zealander Barry Vercoe whose 'Music 11' computer music language was later developed by him into 'Csound'. This language became an international standard running on all small computers. I composed a short 5' piece entitled 'Poi' with its acoustic images of the Maori dance of the same name. I marvelled at the possibilities of shaping sounds and disseminating these through the concert space via computer control.

After the MIT course I travelled onto the UK and worked in Denis Smalley's excellent studio at the University of East Anglia. Here I realised the electronic part for 'Seaswell' for trumpet and electronic sounds for Gordon Webb who was teaching the trumpet and brass studies at the Victorian College of the Arts in Melbourne. I delighted in using the famous Synthi 100 in 'Seaswell' and it was good to return to classical techniques. I particularly responded to hearing the sounds immediately from loudspeakers again rather than having to wait for them from slow computer calculations as had been the case at MIT. In 1979 Denis Smalley was developing sophisticated methods of sound dissemination via multi loudspeaker configurations. His performance practice was to play stereo pieces through 16 loudspeakers. This required much practice and understanding of the space. Denis's concert that year in the 500 year old St Giles Church in Norwich was a real ear opener.

Much later in 1986 the School of Music at the University of Auckland moved to a long awaited new building and we were able to purchase several small computers as part of the establishment equipment grant. By now 'Csound' was running on Apple Macs and soon after I began teaching this language in the graduate electronic music class. Results were immediate with several well crafted and exciting pieces by students. Digital synths with sophisticated midi control via DrT and Protools were also popular at this time.

Gradually the philosophy of the Composition Studios became that of exploiting existing equipment in fresh and imaginative pieces. Some of this is traceable back to Lilburn's sensitive approach to sound, an approach combining an interest in rugged environmental aspects with the most fastidious criticism of the acoustic and musical aspects of the end product, regardless of production method.

With New Zealand's strong tradition of interest in environmental sounds, we forged links with like minded composers in other parts of the world for instance in Vancouver at Simon Fraser University where interesting possibilies of extending sounds by granular synthesis were pioneered by Barry Truax.

My friend and colleague John Elmsly visited Simon Fraser University on our separate short study leave periods. I went in 1985, 1990 and 1995 to work with Barry Truax's PODX system which from 1990 included a granular synthesis engine. In 1985 I worked on 'Fleeting Images' a piece in which I used frequency modulation for the first time. Along with POD's score possibilities I realised a 12' piece in which environmental sound images develop during the piece. The active, rhythmic textures are suggestive of the dynamism of nature - birds, insects and the bubbling energy of geothermal activity while the long sustained sounds evoke images of waves crashing on rocky coastlines mixed with the humming of telephone wires across open plains. The contrast of noise-like with pure timbres is evident as is the "joie de vivre" aspect of the piece heard in the energetic rhythmic sections the pitches and durations of which were chosen by controlled random procedures afforded by the versatile POD system.

When I returned to SFU in 1990 Truax had composed his celebrated 'Riverrun' using his early granular synthesis technique. I was able to understand it fairly quickly and use it in several extended electroacoustic pieces. One of these was a 20' piece entitled 'Beyond the Saying.'

In this work the well-known saying " you reap what you sow" is transformed in such a way that the listener no longer hears the words and therefore comprehends their meaning but instead is transported into a dramatic and fantastic soundworld. In this musical environment speech becomes singing.

'Beyond The Saying' uses five main types of sounds:

1. A brush stroke-like gesture which is heard at the beginning and which articulates the beginning of new sections. It is later developed into a series of comical gestures.
2. Those which are based on the dipthongs of the text 'ou','ea', etc. The sequences which use these sounds function in a similar way to the thematic material in a large scale work such as a symphony.
3. Displays of percussive consonants especially 'p', 't' and 's' from the words 'reap', 'what' and 'sow'.
4. Arch shapes which consist of pure sine wave-like sonorities.
5. The stretching in time of individual words. Here the inflexions of the spoken voice are magnified up to one hundred times the original recording. This type is particularly noticeable in the final section where my speech is transformed into a male voice choir.

John Rimmer and Warren Burt, New Zealand 2006

Back in Auckland in 1991, having made computer transformations of the sound of an oboists reed while at SFU, I composed 'Crow' for oboe and CD. This piece was performed at the ICMC in Banff in 1995. It's a theatre piece in that the oboist appears onstage with all the warm-up and adjusting sounds that oboists make offstage. In 'Crow' these sounds are integrated into the work. This can be heard in the opening where in a short reed sounding 'ritual' the player begins to interact with prerecorded reed sounds which assume prominence as the work proceeds.

In this granulated soundworld tiny grains of sounds provide the basis for a wide variety of textures ranging from single reed sounds as in the opening to a dense, bubbling 'reed dance' heard near the end of the piece. The sounds for the transformation by granulation were derived from the following resources:
1. Various sounds made on the oboe reed, including the oboists 'crow'.
2. An oboe 'multiphonic' or type of chordal sonority.
3. The word "crow" spoken by the Canadian oboist Lawrence Cherney who kindly provided the reed and oboe sounds and much of the inspiration behind the work.
4. A recording of a rooster crow, an item in the extensive sound archives of The World Soundscape Project held at Simon Fraser University.

Combining instruments and voices with electronic music has long been a passion of mine and I've composed nearly 20 pieces of this mixed music variety. I love the challenge of deciding how to combine and blend the different sounds, when to contrast the live performer with the recorded sounds and when to allow for an instrumental, vocal or electronic display. And the rich aural possibilites take on a dramatic effect when the live performer is confronted by the loudspeakers.

In 1995 I encountered further extensions to Truax's granular synthesis programme and used these in 'Pacific Soundscapes with Dancing'

This piece is based on computer transformations of environmental sounds derived from the following: the sea shore, a gentle stream, cicadas, bell birds and kakas all recorded on Little Barrier Island(Hauturu) and volcanic activity on White Island (Whakaari) including ominous sounds from a vent in the side of the main crater and delicate bubbling sulphur streams.

The dancing referred to in the title is an electronic image of boisterous Pacific drumming ensembles and is derived from a tiny fragment of the opening of the composer's "Composition 4 for Flute and Electronic Sounds". These sounds are extended beyond their environmental settings by granulation and time stretching.

During the composition of 'Pacific Soundscapes...' I was aware of the spatial aspects of the original recordings in particular the stereo sound of one parrot(kaka) flying from branch to branch. I decided to incorporate this spatial quality into the computer transformed sounds. In addition the rhythmic feel of the bubbling stream is reflected in many of the drumming sounds and in the musical material before the main climax.

Later in 1995 I worked on a commission from the well-known Institut Internationale de Musique

Electroacoustique in Bourges, France. 'La Voce di Galileo' was the result and this piece was planned as a forerunner or compositional warm-up to my chamber opera about Galileo which I completed in 1998.

'La Voce di Galileo' is based on a short text from the opera and is intended to stand as a separate electroacoustic piece. At this point in the opera, Galileo has been sentenced to house arrest for the rest of his life, his books placed in the Catholic Index of forebidden books and he has been told that he no longer exists. He utters his legendary aside "Eppur si muove"(It still moves) referring to the movement of the Earth. He then meditates on his fate and dreams of the universe dancing. "Eppur si muove" spoken by the librettist, Witi Ihimaera a prominent New Zealand novelist, provides the main sound source together with recordings of an active volcano in New Zealand and planetary tones as a my version of the medieval theory of the 'Harmony of the Spheres'. At each main climax, Galileo's breathing and his voice appear in an elliptical orbit representing the Earth, the planet of life and vitality in contrast to the rather bland planetary tones which pulse along at different speeds.

Incidentally the opera represents a culmination of all my interests in vocal, instrumental and electronic music. Most of the electronic music is heard between the scenes when the 'Music of the Heavens' takes over the drama.

An important part of my electroacoustic music education has been to attend International conferences and festivals. I went to International Computer Music Conferences in Vancouver1985, Glasgow1990, San Jose1992, Banff 1995 Thessaloniki1997, Havana 2001 and Gothenburg 2002

I went also to the electroacoustic festival 'Synthese' in Bourges, in 1990, 1995 and 1999. From 1993 in Sydney, I have been to several of the Australasian Computer Music conferences. At these events it is always stimulating and exciting to meet other like-minded composers, interact with programmers and researchers and find out what current, affordable equipment is available for one's compositional projects.

On a personal note, electroacoustic music becomes another musical instrument. I love the medium as it enables me to realise musical subtleties and colours that I'm not able to achieve with other more conventional musical forces. Not that I have stopped writing for instruments. In fact since I have retired from the University of Auckland in 1999, I have composed mostly orchestral and brass band pieces.

Electroacoustic music offers many opportunites for composers to be inventive, daring, adventurous and imaginative. I look forward to listening to these kinds of pieces.

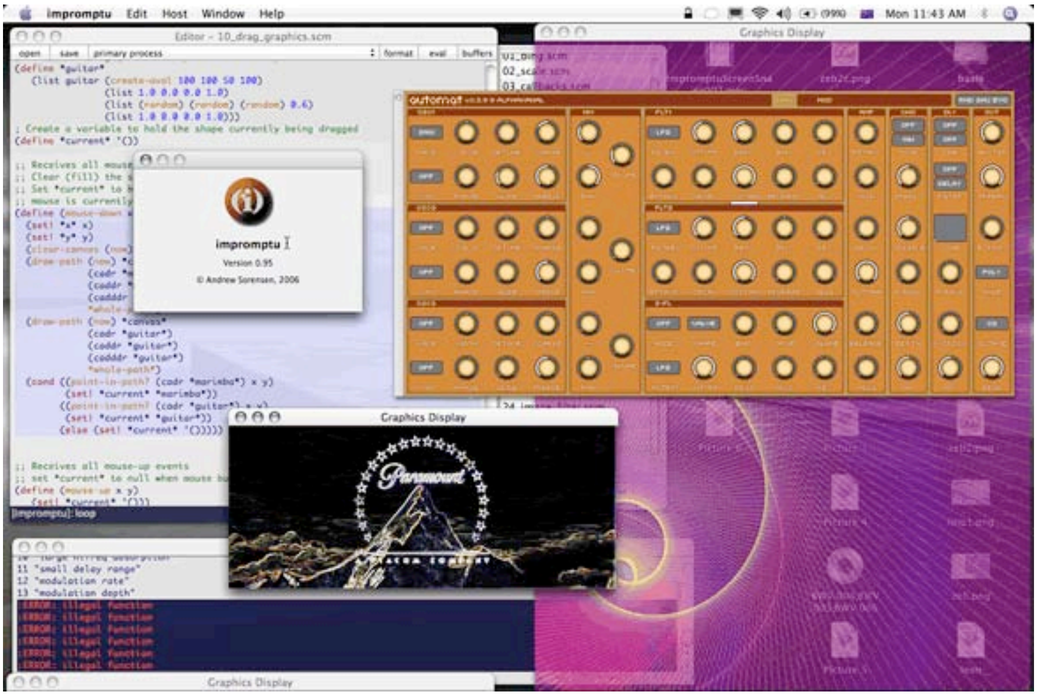
My list of electroacoustic pieces:

Instruments, Voice and Electronic Sounds
COMPOSITION 1 (1968) horn 5 min.
COMPOSITION 2 (1969) wind quintet 12 min.
COMPOSITION 3 (1973) harp 9 min.
COMPOSITION 4 (1972) flute 7 min.
COMPOSITION 5 (1971) perc. 11 min.
COMPOSITION 6 (1972) piano 11 min.
COMPOSITION 7 (1973) bassoon, piano 10 min.
COMPOSITION 8 (1974) violin 15 min.
COMPOSITION 9 (1976) soprano voice 10 min.
COMPOSITION 10 (1977) double bass 17 min.
VISIONS (1975) choir, soprano, baritone solos 15 min
SOUNDWEB (1977) trombone 14 min.
SEASWELL (1979) trumpet 12 min.
TIDES (1981) horn 10 min.
PROJECTIONS AT DAWN (1985) clarinet 12 min.
VOYAGER (1986) recorders 10 min
CROW (1991) Oboe 17 min.
OTHER FLUTES (2002) Baroque Flute 6 min.
COSMIC WINDS (2004) Maori Instruments 'Taonga Puoro' 15 min.

Electroacoustic Music
CANZONA PER ELETTRONICA TONI (1967) 3 min.
HOMAGE TO PAGANINI (1974) 6 min.
WHITE ISLAND (1974) 7 min.
WHERE SEA MEETS SKY 1 (1975) 8 min.
COLDER FAR THAN SNOW (1977) 6 min.
POI (1979) 5 min.
DE MOTU NATURAE (1985) 16 min.
FLEETING IMAGES (1985) 12 min.
BEYOND THE SAYING (1990) 20 min.
AN INNER VOICE (1991) 9 min.
A VOCALISE FOR EINSTEIN (1991) 15 min.
REDUNDANCIES (1993) 6 min.
SHATTERED (1995) 7 min
PACIFIC SOUNDSCAPES WITH DANCING(1995) 15 min
LA VOCE DI GALILEO(1995) 10 min.
PEOPLE, POWER AND POLLUTION(1999) 7 min
ANCESTRAL VOICES (2001) 15 min
MASKED RITUAL (2006) 7 min

Impromptu

Andrew Sorensen



```
(define *guitar*
   (list guitar (create-oval 100 100 50 100)
                (list 1.0 0.0 0.0 1.0)
                (list (random) (random) (random) 0.6)
                (list 1.0 0.0 0.0 1.0)))
; Create a variable to hold the shape currently being dragged
(define *current* '())

;; Receives all mouse
;; Clear (fill) the s
;; Set *current* to b
;; mouse is currently
(define (mouse-down x
  (set! *x* x)
  (set! *y* y)
  (clear-canvas (now)
  (draw-path (now) *c
             (cadr *m
             (caddr *
             (cadddr
             *whole-
  (draw-path (now) *canvas*
             (cadr *guitar*)
             (caddr *guitar*)
             (cadddr *guitar*)
             *whole-path*)
  (cond ((point-in-path? (cadr *marimba*) x y)
         (set! *current* *marimba*))
        ((point-in-path? (cadr *guitar*) x y)
         (set! *current* *guitar*))
        (else (set! *current* '()))))

;; Receives all mouse-up events
;; set *current* to null when mouse bu
(define (mouse-up x y)
   (set! *current* '()))
```

```
11 "small delay range"
12 "modulation rate"
13 "modulation depth"
:ERROR: illegal function
:ERROR: illegal function
:ERROR: illegal function
:ERROR: illegal function
:ERROR: illegal function
:ERROR: illegal function
```

By the time you read this article Impromptu will have just completed its first year of development. A lot of ground has been covered in that 12 months and in this article we take a look at some of the core ideas behind Impromptu's development.

Impromptu is a programming environment for musicians and animators. It is a dynamic, real-time environment where time plays a central role. The core idea behind Impromptu is to tightly integrate a dynamic interpreted language (Scheme) with a real-time scheduling engine in order to develop a dynamic, temporal programming environment. Impromptu leverages Apple's audio and graphics libraries and so it is only available for the Macintosh platform.

So that's Impromptu in a nutshell, but there were a lot of terms used in the description so in this article we'll unpack what some of them mean and how the various design decisions reflect particular perspectives on the nature of creative practice. Let's begin by looking at the basic architecture.

ARCHITECTURE

Impromptu is an amalgam of five primary components, a real-time scheduler, a Scheme language interpreter, an Integrated Development Environment (IDE), an audio plugin architecture and a vector based graphics engine. In this section we'll take a brief look at each of these components.

At its most fundamental level, Impromptu's real-time scheduler is a priority queue whose tasks are linked to C++ instance methods. At this level of operation, a method of any C++ class instance can be registered with a task and scheduled on the priority queue. This is the basis for all precisely timed activity in the Impromptu system. Scheduling in Impromptu is sample based and it attempts to guarantee sample accuracy (more on this later).

The scheduling engine is driven by the currently selected audio device which is automatically started when Impromptu initialises. Any default audio device settings, including samplerate and channel configurations will remain active for the duration of the Impromptu session At this stage in development, applying changes to the default audio device or it's parameters midway through an Impromptu session requires a restart. Impromptu uses Apples CoreAudio engine, and more precisely Apple's AudioUnit plugin architecture for all audio processing; it is an AudioUnit host. Apple's AudioUnit library provides a plugin architecture allowing individual AudioUnit's to be connected together to form signal processing graphs of arbitrary complexity. The AudioUnit standard is well supported and allows Impromptu to access an enormous array of commercial and non-commercial virtual instruments, filters, delays, reverbs, mixers, generators and so on.

AudioUnits

Once Impromptu is initialized the AudioUnit graph is started with a single default output unit. An impromptu preference specifies whether this should be a stereo or surround sound output. Once started the AudioUnit graph will run for the duration of the session, pulling audio data directly from the default output unit, which in turn pulls data from any units that are attached to it. Impromptu is always "on" not unlike Max/MSP or Supercollider server. There is more information on the AudioUnit graph in Apple's developer documentation for the interested reader. AudioUnit's can be instantiated during the session and freely attached or detached from the Graph at run-time. The AudioUnit graph runs in a real-time thread and is responsible for audio production, updating Impromptu's internal sample clock and notifying the scheduling thread each buffer cycle. The scheduler is run in an independent thread to ensure that the execution of scheduled tasks takes a lower priority to audio processing. Impromptu's internal clock is incremented in buffer size increments where the buffer size is an Impromptu parameter that defaults to 128 samples. At it's worst this means that scheduled events can be out by up to the buffer size. However, the AudioUnit specification includes an event offset that allows AudioUnit calls to be offset from buffer boundaries. This results in guaranteed sample accuracy for most audio events (note-no's, note-off's, control changes, program changes, AudioUnit parameter changes and the like..).

Scheme

During Impromptu's initialisation a default Scheme interpreter is started. Impromptu uses a modified version of the TinyScheme interpreter, a light-weight Scheme interpreter written in the C programming language. The interpreter runs as a service allowing local or remote connections. This provides a flexible architecture whereby multiple remote users can share a single interpreter or a single user can run multiple local interpreters. Each new interpreter started on a single host shares the same AudioUnit Graph and internal clock but runs within in it's own Mach kernel thread. A shared clock ensures that multiple local interpreters can be temporally synchronised. Impromptu also provides the ability to set the internal clock, allowing remote interpreters to synchronise across a network. Impromptu supports OSC messages for network communications.

A large library of native calls provide the Impromptu scheme interpreter with direct access to the Impromptu scheduling engine, audio engine and graphics engine. Of primary importance among these is the CALL-CC function which allows Scheme to schedule C++ methods for later execution. All C++ methods available from within the Scheme environment are defined as global constants when the interpreter is initialised. The CALL-CC function uses these constants to build tasks to send to the scheduling engine. Along with the C++ method a Scheduled task must also contain any arguments that the method may require. C++ methods can access Scheme values directly from the interpreters heap. The interpreters garbage collector has been modified to allow this to occur in a thread safe manner. The Scheme interpreter also ensures that any Scheme values currently stored as arguments on the Scheduling queue are "protected" from garbage collection. This is required so that Scheme values passed as arguments to C++ methods are not garbage collected until after the scheduler executes the method call. The delay between the initial Scheme call and the scheduled C++ call could be samples, minutes, hours or days. Impromptu users are hidden from this layer of complexity and should never need to call CALL-CC directly. Instead, Scheme wrapper functions are used to hide this scaffolding.

Editor

In order to evaluate Scheme expressions 'on-the-fly' Impromptu includes a dynamic code editor that is designed to assist real-time, dynamic programming. As well as supporting standard features such as colour-syntax highlighting, bracket matching, auto-indenting and code markers, the editor also supports the ability to send code to the Scheme interpreter for evaluation. This is achieved by pressing the evaluate button (or equivalent keyboard shortcut) while the cursor is located within a valid Scheme expression. The editor will pass the expression to the currently selected Scheme interpreter (this is usually the default interpreter) across a TCP connection (remember that interpreters can be local or remote). The Scheme interpreter will then evaluate the expression and display the result in the editors result bar. The Impromptu IDE also supports custom AudioUnit User Interfaces, allowing users to modify AudioUnit parameters in real-time via their User Interface as well as programatically through Impromptu library functions.

The Impromptu IDE also enables graphics canvas's. Impromptu supports bezier paths, font rendering, image manipulation and QuickTime movie support. The complete set of CoreImage filters are supported along with UI event capture including mouse down, mouse up, mouse drag, QWERTY key up and down etc.. Impromptu's real-time scheduling provides interesting opportunities for animators who are interested in breaking free from the traditional timeline. Animators can schedule drawing commands and implement multiple time independent animations using temporal recursion.

SCHEDULING CODE

The ability to precisely schedule events in real-time is not uncommon for computer music languages. What is less common is the ability to precisely schedule code execution. Impromptu provides programmers with the ability to precisely schedule the evaluation of arbitrary Scheme functions asynchronously. SuperCollider language also schedules code blocks asynchronously and the Chuck language offers similar capabilities but uses a synchronous approach. Impromptu attempts to integrate this functionality in as Scheme friendly a manner as possible.

The CALLBACK function (a wrapper function for CALL-CC) is responsible for Scheduling a task that calls another Scheme function. Scheme functions are first class values and can be passed as arguments to other Scheme functions. Let's look at a simple example.

```
(CALLBACK (+ (NOW) 44100)
          (LAMBDA () (PRINT "Hello World")))
```

This example will call the anonymous function (LAMBDA () (PRINT "Hello World")) in exactly 44100 samples from (NOW). The function (NOW) returns the number of samples that have been processed since Impromptu was initialised (i.e., the current internal clock time). Evaluating this expression results in "Hello World" being printed to the log view one second after evaluating the expression. Lets reuse the example but this time use a function defined in the global environment (i.e., bound to a symbol) and pass the string to be printed as an argument to the function.

```
; First define my-function taking one string argument
(DEFINE my-function
  (LAMBDA (str-arg)
    (PRINT str-arg)))

; Call my-function in one second from now passing string
; argument
(CALLBACK (+ (NOW) 44100) my-function "Hello World")
```

Notice that CALLBACK also takes any arguments that are required by the function that we want to call back to. The ability to precisely schedule the execution of arbitrary code blocks is a useful technique, particularly when we begin to think recursively.

TEMPORAL RECURSION

Impromptu's ability to schedule code encourages temporal recursion. Using the CALLBACK function it is possible to schedule function recursions at defined intervals. Let's take a look at the definition of a plain old recursive function in Impromptu.

```
(DEFINE my-func
  (LAMBDA ()
    (PLAY-NOTE (NOW) my-instrument 60 80 1.0)
    (my-func)))
```

This function will recursively call back into itself playing a middle C each time my-func is called. The problem with this example is that the function will call back into itself as fast as it can execute. The result of this example will be hundreds of notes all trying to play at approximately the same time. What we need to do is to schedule the call back into my-func. The CALLBACK function we just learnt about provides this facility. Adding CALLBACK to the previous example:

```
(DEFINE my-func
  (LAMBDA ()
    (PLAY-NOTE (NOW) my-instrument 60 80 1.0)
    (CALLBACK (+ (NOW) 44100) my-func)))
```

This time instead of calling directly back to my-func we schedule a call back to my-func by placing my-func on the scheduling queue at now+44100 samples. This sets up a temporal recursion (or temporal loop) with precisely timed increments that can be understood also as a control rate for the recursion. There is an added benefit here - CALLBACK is asynchronous. Once we have scheduled the CALLBACK, my-func completes and we are free to process other functions. This is somewhat like a multi-threaded environment where each temporal recursion operates in parallel. As a further example, we'll set up a quick drum loop with three independent temporal recursions.

```
(DEFINE kick-drum
  (LAMBDA ()
    (PLAY-NOTE (NOW) my-drumkit kick 80 1.0)
    (CALLBACK (+ (NOW) 44100) kick-drum)))

(DEFINE snare-drum
  (LAMBDA ()
    (PLAY-NOTE (NOW) my-drumkit snare 80 1.0)
    (CALLBACK (+ (NOW) 88200) snare-drum)))

(DEFINE hi-hats
  (LAMBDA ()
    (PLAY-NOTE (NOW) my-drumkit
             (RANDOM '(hat-open hat-closed)) 80 1.0)
    (CALLBACK (+ (NOW) 11025) hi-hats)))
```

Of course we also need to make an initial call to our new functions to start them off.

```
(kick-drum)
(snare-drum)
(hi-hats)
```

We now have a very simple drum loop. Of course our temporal recursions could just as easily animate graphics or sweep an AudioUnit's cutoff frequency.

IMPROMPTU AS CONTROLLER

Impromptu is primarily an algorithmic composition environment and does not directly support DSP in Scheme. However, the rich controller interface provided by modern plugins (particularly AudioUnits), mixed with Impromptu's scheduling capabilities, provides sophisticated parametric control over all aspects of the signal processing chain, from sound spatialisation through to envelope adjustment, filter frequency and oscillation depth. It is also worth mentioning that while Impromptu does not support synthesis directly users can combine and control low level AudioUnit components (basic oscillators, filters, delays etc. can all be found as AudioUnit plugins) in a similar manner to the UGen's available in other Music V style environments.

Temporal recursion again provides a powerful tool for controlling AudioUnit parameter data. The ability to programatically control hundreds of independent AudioUnit parameters in real-time, each at their own control-rate, offers very fine control over virtual instrument performance. Lets look at an example that sweeps a filter cutoff back and forth. In this example we use the AU-MIDI-OUT function to send control change messages (control change 1 in this example) to an AudioUnit "my-instrument". This example updates my-instrument every 441 samples (100 updates per second) completing a full oscillation once every second.

```
(DEFINE cutoff
   (LAMBDA (phi)
      (DEFINE val (+ 64 (* (SIN (* 2PI phi)) 63)))
      (AU-MIDI-OUT (NOW) my-instrument
                   *midi-cc* 0 1 val)
      (CALLBACK (+ (NOW) 441) cutoff (+ phi 0.01))))
```

Impromptu provides four primary options for I/O, MIDI in and out (both external MIDI-IN and MIDI-OUT and internal AU-MIDI-IN and AU-MIDI-OUT), OSC in and out, Audio in and out and internal AudioUnit parameter setters and getters. Additionally Impromptu supports a number of UI options including mouse and keyboard event capture. This provides a range of options for communicating with sound sources either internal (AudioUnits) or external (Max/MSP or external synths for example). It is a trivial process to setup Impromptu for routing I/O data. Here is an example that accepts incoming midi data from an external source and uses that data to trigger an external OSC host.

```
; first define host IP address and port for OSC udp
(DEFINE host (cons "192.168.9.1" 7009))

; define midi-in function to start receiving midi data
(DEFINE midi-in
   (LAMBDA (type channel a b)
      (COND ((= type *note-on*)
            (OSC-SEND host "/inst/syntha/start" a b))
           ((= type *note-off*)
            (OSC-SEND host "/inst/syntha/stop" a b)))))
```

GRAPHICS

Impromptu’s graphics functions are scheduled in exactly the same way as its audio functions. To demonstrate, we’ll add some simple graphics to the drum example from Section 3.

```
(DEFINE kick-drum
   (LAMBDA ()
      (PLAY-NOTE (NOW) my-drumkit kick 80 1.0)
      (DRAW-PATH (NOW) my-canvas
                 (GET-RANDOM-CIRCLE)
                 my-red my-blue *whole-path*)
      (CALLBACK (+ (NOW) 44100) kick-drum)))
```

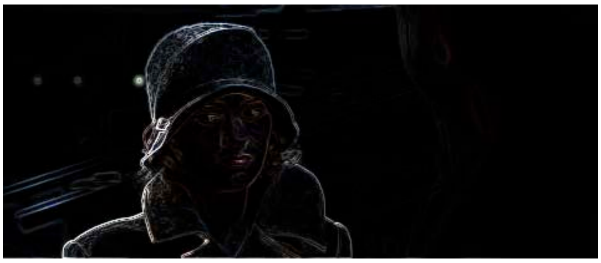
Impromptu’s graphics support leverages the CoreImage library to provide an extensive range of image manipulation filters. Apple ships this library of professional-quality image filters as a standard part of the OSX 10.4. Impromptu supports CoreImage filter “pipelines” of arbitrary complexity, providing the ability to produce sophisticated image-manipulation effects.

Let’s take a look at a complete example that applies a CoreImage edge-detection filter to a QuickTime movie.

```
(DEFINE canvas (CREATE-CANVAS))
(DEFINE movie (LOAD-MOVIE "/tmp/myfilm.mov"))
(DEFINE filter-params '(("inputIntensity" . 5)))

(DEFINE loop
  (LAMBDA (time movie-time)
    (LET* ((image (GET-MOVIE-FRAME movie movie-time))
          (filtered-image (FILTER-IMAGE image "CIEdges" filter-
params)))
        (DRAW-IMAGE time canvas filtered-image 1.0 75 150)
        (CALLBACK (+ time 4000) loop (+ time 4410) (+ movie-
time 0.1)))))

(loop (NOW) 0.0)
```



We start by instantiating a canvas. This automatically opens a window suitable for drawing on. We then load a QuickTime movie and define the "inputIntensity" parameter for an edge-detection filter. Next, we define a “loop” function, utilising temporal recursion to play back the movie, at a frame-rate defined by the callback frequency. The function “loop” applies the image filter to each frame, prior to drawing it to the canvas, then "loop" plays back filtered frames at “normal” speed, skipping frames as required.

Note that the “time” parameter to “loop” is sample-oriented, as per Impromptu’s scheduling interface, whilst the “movie-time” parameter is time-based (in seconds). GET-MOVIE-FRAME returns the corresponding movie frame, based on the elapsed-time parameter passed to it. The “loop” function passes “movie-time” to GET-MOVIE-FRAME as the elapsed-time value. Each new invocation of “loop” increments “time” by 4,410 samples (i.e., 0.1 seconds), and “movie-time” by 0.1 seconds, thus ensuring that movie play-back runs at “normal” speed, and at a frame-rate of 10 frames per second. GET-MOVE-FRAME is used to skip intervening frames. Thus, the frame-rate could be doubled to 20 frames per second, whilst retaining play-back at the same speed (i.e., skipping fewer intervening frames) by incrementing “time” by only 2,205 and “movie-time” by 0.05 for each invocation of “loop”.

Note that the “loop” invocation is scheduled for execution at (time + 4,000) samples, whilst the “time” argument to be passed to “loop” is incremented by 4,410. In other words, “loop” will execute (just) ahead of the time at which the events it schedules are required to be executed. This is to give “loop” enough time (in this case, 410 samples) to construct and schedule its events. The only requirement is that execution of “loop” precedes the events it wishes to schedule with enough time to evaluate the function.

CONCLUSION

Impromptu has matured rapidly over the past 12 months and now provides a reliable platform for algorithmic software development and real-time performance. Scheme is an elegant and extensible language with a long history in the fields of machine creativity and algorithmic art. There are a number of free, high-quality Scheme tutorials available both online and in print. A list of Scheme resources can be found on the Impromptu website, http://impromptu.moso.com.au, which also hosts an extensive collection of example Impromptu code and an introductory video. You can download a copy of Impromptu for OSX 10.4 from the website.

ACMA Products

The following items will be available for sale in Adelaide at ACMC06, or can be ordered from the Publications Officer.

ACMC05 Conference Proceedings:	
Printed (181 pages)	$20
CD-Rom	$5
ACMA CDs:	
Machine Message	$15
Assembly	$15
acma2004	$15

Upcoming events:

ACMC 2006: Medi(t)ations

11 - 13 July 2006

Elder Conservatorium of Music, University of Adelaide
Adelaide, Australia

http://www.acmc06.org

ICMC 2006: Multidimensionality

6 - 11 November 2006

Tulane University Music Department,
New Orleans, Louisiana, USA.

http://www.icmc2006.org/

Generate and Test at QUT

Gordon Monro
www.gommog.com

This is a report on the 2005 Australasian Computer Music Conference (ACMC05), held on 12-14 July at the Creative Industries Precinct, Queensland University of Technology; the conference theme was "Generate and Test". The conference was ably organised by Greg Jenkins, assisted by Andrew Brown and Tim Opie (responsible for the conference proceedings), Jason Zadkovich (resident technician), and a team of about 20 student volunteers. Usual disclaimer: what follows is a personal view of a complex event.

In the three days we heard about 30 presentations and 30 pieces of music; in addition there were installations, a poster session, a forum about computer labs, and the after-hours Conference Club. After the conference finished, there was a day of workshops, which I did not stay on for. The conference was well attended, and there were quite a number of new faces as well as the usual regular attendees.

Papers

Keynote addresses

There were three keynote addresses. The first was by Paul Doornbusch, outgoing ACMA President. He started by describing his work on reconstructing the music of the Australian CSIRAC computer, probably the first in the world to play music. Paul went on to discuss the work of various composers, including Hiller, Brün and Xenakis, from the point of view of "generate and test". He made a plea for artists not to get completely wrapped up in the algorithms ("generate"), but to develop their aesthetic judgement as well ("test").

The second keynote address was by Katharine Neil, who has spent the last seven years as a computer game sound designer and programmer. She pointed out that avant-garde compositional practices such as aleatoric music and music that responds to the environment are common in game music, and reach a huge audience. The artistic and technical challenges are considerable, and sometimes bizarre: one game music composer spends his time creating bridge passages linking any one theme from the "Star Wars" movies to any other, to allow for fluid gameplay. Katharine's entertaining talk was the first of a small group with a commercial focus, something fairly new for ACMA. In this context it occurred to me that the Sydney Conservatorium of Music, where I am currently a student, has what amount to strong commercial links, as it is closely connected with the orchestras and the opera.

The third keynote address, from Ross Bencina, came right at the end of the conference. Ross started by giving a short performance using a glove and his well-known AudioMulch software. He then gave a thoughtful discussion of the process of developing software for creative purposes. Ross's software was written to support his own artistic practice, but it contains close to 100,000 lines of code; a project of this size requires a disciplined "engineering" approach, and much of the code is either implements the user interface or provides more or less general "infrastructure". The DSP code is only a small part of AudioMulch. Any piece of music software imposes some stylistic bias and constraints (despite aspirations to the contrary); Ross said that Miller Puckette has recently called for the models and abstractions used in music software to be made explicit and documented. Incidentally, the tension between creativity and disciplined production is not specific to music software, but surely has been around at least as long as the craft workshop.

Contributed papers

These comments do not attempt to cover all the papers — order the conference proceedings for your library! The conference covered a wide range of topics; the papers did not fall neatly into subject groups, though many had some connection with performance. Those papers with a potential commercial connection were concerned either with computer games or with ways to help non-musicians or beginners to make music. There was one survey paper, Jim Barbour's timely account of the current state of delivery methods for multi-channel audio.

Steven Livingstone, Ralf Muhlberger and Andrew Brown presented an ambitious scheme to mark up music with "emotional metadata", used to influence MIDI playback. The ultimate aim is for the emotional state of the listener to influence the emotional tone of the playback, with the main intended application being computer games, though Steven hoped the project might be useful for educational purposes as well.

Garth Paine (in joint work with Ian Stevenson) talked about mappings for a new performance controller, the "Thummer", being developed by Jim Plamondon. Jim was at the conference and ran a post-conference workshop on the Thummer. Unfortunately Garth could not be specific about the Thummer, as Jim wants to protect its details for the time being (I assume that the workshop attendees had to sign a non-disclosure agreement). So Garth's talk was rather too abstract. We do know that the Thummer has a large number of buttons, concertina-style, laid out so that the fingering is the same in any key. It also has 10 or so continuous controllers, but what they are remains a mystery, and therefore so does the potential of these controllers for expressive performance. Although the Thummer is a controller, it is not restricted to MIDI; I understand that the controller sends raw data to an attached computer, from whence it can be sent on in various formats, of which MIDI is only one. The Thummer is also claimed to be well adapted to microtonal music. Unfortunately Ian Stevenson wasn't at the conference to present his more general paper on “Design issues for new performance systems”; it is in the conference proceedings.

A paper by Greg Schiemer and Mark Havryliv described a system for using mobile phones as musical performance instruments, via Bluetooth and Java applications running on the phones. Performers can modify the sounds coming from their own phones of from other participants' phones, including changing the tuning system. Greg and Mark have developed a PD to Java conversion tool, allowing PD patches to run on the phones. In this context Warren Burt commented on the perverse tendency of electronic musicians to continually revert to using crappy-sounding devices.

Rene Wooller and collaborators described their work on a “participatory electronic dance music environment”. built on Rene's LEMu program for generating electronic dance music in real time. Rene's group have built various controllers, including robust floor mats and the “Seuss Stick”, an instrument with a telescopic action like a bicycle pump, the ability to rotate one part relative to the other, and three pressure-sensitive pads. Rene discussed the problems involved in having a group of people collaboratively controlling a dance music engine.

Some other papers concerning performance are mentioned below in connection with the concerts. I would also like to mention briefly Alistair Riddell's inspiring talk about a performing ensemble he was involved in, using glove-like sensors, and Steven Campbell's talk about PlaY+SPaCE, a system similar to Rolf Gehlhaar's long-established SOUND=SPACE in that it uses multiple ultrasonic sensors, but having the advantages of being much cheaper and of being controlled from Max/MSP.

A small group of papers concerned algorithmic or generative composition. Luke Harrald presented his work on using the Iterated Prisoner's Dilemma as a compositional engine. In the Prisoner's Dilemma, two (guilty) prisoners are interrogated separately. If both remain silent they each receive 1 year in jail. If one confesses and dobs his mate in, the dobber gets a pardon, and the other prisoner gets 5 years. If they dob each other in, they both get 3 years. In the Iterated Prisoner's Dilemma, this is repeated many times, and the “prisoners” can make choices based on what happened to them in previous rounds. Luke has made several pieces using the actions of simulated prisoners to control musical material in various ways.

Other papers involving algorithms were Dave Burraston's paper presenting his ongoing work with cellular automata as compositional generators, and the paper by Francis Soddell and Jacques Soddell, who are respectively a computer scientist and a microbiologist, on using L-systems to make music for an installation. L-systems produce tree-like structures, and in this case an L-system was constructed to make branching shapes like those generated by a particular fungus. Traversing the structure produced a MIDI file; the choice of timbres was made freely for aesthetic reasons. In this context Warren Burt commented that in general sonifications succeed or fail as artworks according to the choice of timbres, and this work was no exception: an example with a piano timbre was unimpressive, but the final result sounded good, and well-suited to the environment of the installation (a botanical conservatory).

Peter McIlwain presented a paper (joint work with Jon McCormack) on a deceptively simple network traversal algorithm for generating melodies. An “activation point” moves around the network according to some simple rules, triggering a note at each node. The time interval between two notes depends on the the length of the line joining the nodes. The idea has some possibilities for analysis as well as melody generation, which Peter demonstrated with a discussion of “Three Blind Mice”. A couple of the papers discussed below in connection with the concerts were also related to generative composition.

There were three papers on analysis. Tim Opie presented a paper on aspects of sound amplitude as part of an ambitious study of the various attributes of natural sounds, with a goal of compositional applications. Barry Hill compared and contrasted two contemporary electronic genres, Trance and Drum n Bass, backed up by transcriptions. Unfortunately David Hirst was unable to attend the conference to present his work on computerised study scores for electro-acoustic music. This work represents a practical outcome of David's earlier more theoretical work on the analysis of electro-acoustic pieces. David's study score for a piece (Denis Smalley's “Wind Chimes”) consists of two spectrograms at different levels of detail together with a meticulously marked diagram showing all the significant sonic events. The whole thing is interactive, being realised in Flash, so one can listen to a section of the piece at the same time as scrolling through the corresponding visual representation.

Forum on computer music laboratories

There was a forum on computer music laboratories, with Alistair Riddell, Garth Paine, Tim Kreger and Andrew Brown, and contributions from the audience. The question of the role computer music labs play in an era when most students have their own machines has been around for some time. A partial consensus did seem to be reached by the forum, along the following lines. (Apologies: most of the time I didn't note who said what.)

A computer music lab will be needed for non-specialists who may only take one or two courses. Students who are specialising in computer music can be expected to buy their own machines. Garth Paine indicated that in the UK, Apple have a leasing scheme for students, and there are some moves towards setting up such a scheme here. Garth also said that the University of Western Sydney is looking at licensing deals so that students can be provided with a CD-ROM of appropriate software. This would answer a comment of Alistair's, that students needed to be weaned off simple-minded software, and fits with a general view that a common environment makes for a more cohesive student group. (Though someone defended having a multiplicity of software.)

Alistair also commented that students undertaking animation or other digital arts projects may still need more powerful computers than they can be expected to buy; in general there may be a need for some special-purpose dedicated computers. The convergence of sound and vision will need to be addressed, if that is the way our community is moving.

Given that students can do most of their work at home, why would they come in to Uni? There was a strong consensus that they *should* come in, as they do learn from one another. The consensus was that the money saved by having fewer lab machines should be used for facilities which the students cannot easily replicate at home, such as good recording facilities and a high-quality multi-channel playback space. It was also suggested that the typical computer lab is a fairly unpleasant environment, and that the shared spaces for students should be made as pleasant as possible. Someone also suggested that students should be given team projects, which means that they have to collaborate.

The participants said that getting centralised IT support to understand the needs of computer music was very difficult; for example IT is likely to object to computers with CD burners because of piracy concerns. Good tech support is very important, and it is not clear what the responsibilities are if students are *required* to provide their own machines, as opposed to choosing to provide them. The legal issues also need to be well sorted out.

Installations and Performances

Installations

There were several installations, but I only engaged with two, Amanda Cole's “Sine of the Tones”, a Max/MSP audio-and-video piece with changing coloured stripes, and Michael Yuen's “Atrium” in a foyer, where small speakers caused the glass walls to emit soft hissing and scratching sounds. At least one of the others wasn't working when I was free to see it.

Concerts

By my count there were thirty pieces in the six concerts; 11 were live performances, two were abstract videos and the rest were tape pieces. Most of these were multi-channel works (for between four and eight channels), but there were several stereo pieces, showing that the genre is not as extinct as it appeared to be a couple of years ago. Only one piece was “diffused” live, Camilla Hannan's soundscape of apparently unpeopled factories.

The highlight among the tape pieces was Robin Fox's “The Third Space” in three movements. The second movement consisted of a pulsing sound which was swept around the ring of speakers, at first slowly and then faster and faster. Some strange phasing effects resulted. The outer movements consisted of small fragments of sound carefully arranged and spatialised. Navin Dolosawa's short tape piece “Elevator Music” was made using a gadget, designed for paragliders and the like, which gives out beeps which indicate whether it is rising or falling. Navin gave a talk about this, in which he explained that the aural cues from such a device are essential for paragliding.

Warren Burt's piece “Beneath the Slopes of Mt Corrimal” was a recording of a live performance by several computer-controlled microtonal instruments located at the Logos Foundation in Ghent, Belgium, and designed by Godfried-Willem Raes. Warren composed this piece by remote control, sending MIDI files to Belgium and receiving recordings back, The effect was of many music-box-like sounds arranged in a dense texture, with occasional clangs from a computer-controlled piano. The strange, almost wooden sounds in the piece came from a quarter-tone pipe organ with one bellows per pipe, which only gives out a short puff of air per note.

Warren talked about this piece, and indicated that although the remote composition process was successful for the instruments he used, he felt unable to use the instruments in the collection that have freely variable pitch, namely sirens and a computer-controlled musical saw. Warren considered that to learn how to use these instruments successfully it would be necessary to actually go to Ghent.

Of the live performance pieces, the highlight for me in terms of artistic quality was Brigid Burke's “Grit”. Brigid is a composer, performer and visual artist, and she combined all of these talents in this piece, playing her clarinet through an effects unit along with a DVD with recorded processed clarinet sounds and visuals consisting of largely of semi-abstract collages, together with a video of a wave ebbing and flowing over sand.

Several of the pieces raised questions about performance practice. The piece “Invisible Territory” by Mark Pedersen, with Ralf Mulhberger on shakuhachi, was a sonification of a historic game of Go. Mark and Ralf gave a talk explaining the principles of the sonification, which used spatialised shakuhachi samples to indicate the distribution of the pieces, and other information. The live shakuhachi controlled the tempo of the playback of the game, but the game was not projected during the performance. Several people asked Mark why not; the reply was that this was an experiment to see if the piece had a strong enough structure without such a projection. The answer would appear to be “no”.

Greg Schiemer (who unfortunately couldn't be at the conference) had his piece involving mobile phones (mentioned above) performed, the performers swinging the phones around their heads. For me this piece didn't work: it lacked the elegance of its distant predecessor, Greg's “UFOs”, small sinewave oscillators also swung around the performers' heads. There all the interest came from the effects generated by the swinging, while with the phones it was difficult to tell what was happening. The start of the performance was delayed because it turned out there was another Bluetooth phone active in the room (belonging to Greg Jenkins).

New to ACMA was the practice of “live coding” or “live programming”, the creation of the algorithm for the piece during the actual performance. This was bravely demonstrated during the last concert by the “Andrews Sisters” (Andrew Sorensen and Andrew Brown), using an environment developed by Andrew Sorensen. We saw Andrew Sorensen's Scheme code projected up as he typed it in, complete with syntax errors. Only basic constructs were used (no high-level predefined functions). Nonetheless the two Andrews constructed a complex ostinato, with some evolving structure. Opinions differed as to the point of the exercise; at least with this audience a substantial proportion of it could more or less follow the computer code.

Also new to ACMC (I think) is the practice of on-the-fly score generation, where the players (of conventional instruments) are sight-reading the material as the computer creates it. Two such pieces were performed. The first was by Andrew Brown, for five players. Each performer was reading standard notation from a computer screen, with the computer generating material a few bars ahead. Andrew acted as “page turner”, periodically pressing a button to display the next phrase; necessary, as the tempo was set by the performers. The actual piece was boring, being produced by a process with essentially no memory. The piece should perhaps have been held over until a more convincing compositional algorithm is developed.

The second piece with a score generated in real time was “Community Aid: Resonant Energy”, by American David Birchfield. It was brought to us by David's collaborator, percussionist Robert Esler. The composition process was much more sophisticated, involving a multi-level genetic algorithm generating a complex-looking graphic score. As well as Robert's live playing, the computer was playing back percussion samples. Robert was seated with a large cymbal on his lap and a collection of smaller instruments (shakers, bowls, etc.) at each side. This particular performance was fairly relentless, with little obvious variation. Robert, who gave an informative talk about this piece, said that the algorithm can produce a much more varied result. He also indicated that he had quite a degree of interpretative freedom, in particular regarding choice of instruments. I thought that the performance looked like someone trying to solve the Towers of Hanoi puzzle, involving moving objects among three piles, and perform a ballet at the same time.

The Conference Club

This year there was an off-campus Conference Club. Held in a small inner-city space, with wine and beer available, it provided a venue for laptop improvisation, starting at 8 pm or so, and kicking on to late. On the first night people did more or less listen, but as the conference went on the crowd and, in reaction, the playing got successively louder. My own laptop performance, which was intended to be mostly quiet, took place straight after the free drinks that marked the end of the conference proceedings. The crowd was on an end-of-conference high, the noise from the room poured into my mike, and my performance ended up completely derailed. Fittingly, the night ended with an ear-splitting feedback performance by Cat Hope from Perth. It actually sounded good from my position, outside the door wearing earplugs.

So this year ACMC offered us two different performance contexts, a concert-hall sit-quiet-and-listen and a noisy club. Some of the laptop gigs I have been to in Sydney offered an intermediate environment, moderately quiet, but with some freedom to move around and talk softly. Several of the ACMC “club” performances would have been suited to such an environment, my performance and that of Simulus (Steve Adam and Tim Kreger on this occasion) for two. On the other hand two of the “concert” performances didn't really fit in the concert-hall environment, namely the laptop impro by Aaron Hull and the performance by Botborg (Scott Sinclair and Joe Musgrove) involving audio and video feedback, treating audio signals as video and vice versa.

These comments about venues are not intended as a criticism of the conference organisers. The Conference Club generated its raucous dynamic spontaneously, and the provision of more than one performance environment reflects the broadening of the ACMC in recent years to include more than just traditional “academic” computer music. (This notion of “academic” needs revision; now, the more adventurous popular genres are comfortably accommodated in at least some academic environments.)

Certainly, we were presented with a quite a range of performance practices at the conference, from playback of acousmatic tape music to through to Cat Hope's feedback performance. Some of these performance practices have more or less established social conventions surrounding them. Two certainly do not, the “live coding” of Andrew Sorensen and Andrew Brown, and the two performances where the score was generated in real time. Since in these last two pieces the performer(s) had no influence over the score, it is a fair question to ask what the difference would have been if the score had been generated in advance; in other words, what the point of this performance practice is. The answer, if there is one, must come from the social relations surrounding the performance. Maybe the natural home for this practice is the computer game, not the concert hall.

Greg Jenkins and his team are to be congratulated on a very successful conference, the third ACMC to be held at Queensland University of Technology. QUT's support for ACMC, and for computer music in general, is something to be very grateful for.

CLATTERBOX
Attention experimental instrument builders in Australia

clatterbox is undertaking a survey of people building experimental music/sound instruments in Australia.

Completing the survey is easy. Download the form at : http://www.clatterbox.net.au/survey.doc

Fill out your responses and SAVE the form.

Email your SAVED form back to : sean@clatterbox.net.au

Your responses to this survey will help me continue to develop the clatterbox website and other possible projects.

Want to contribute to Chroma?

Chroma is always accepting articles, reviews, previews, information, concert details, conference stories, poems, musical scores, code, pictures, place mats, and pears.

So submit now:
acmachroma@gmail.com

"..the media have broken down in their traditional forms, and have become merely puristic points of reference. The idea has arisen, as if by spontaneous combustion throughout the entire world, that these points are arbitrary and only useful as critical tools, in saying that such-and-such a work is basically musical, but also poetry. This is the intermedial approach, to emphasize the dialectic between the media."

~ Dick Higgins (August 3, 1966)

To be addressed in Adelaide……………

Medi(t)ations

Australasian Computer Music Conference 2006

11 - 13 July 2006
Elder Conservatorium of
Music, University of Adelaide
Adelaide, Australia

Call for Submissions

All submissions are due Tuesday, 18th April 2006

All submissions to be submitted via this web site:

http://www.acmc06.org/registration.html

Details of how to make submissions for:

- * Musical works
- * Papers
- * Posters
- * Artist talks
- * Studio reports
- * Workshops

Musical works

Due Date: Tuesday, 18 April 2006

Submit: http://www.acmc06.org/registration.html

ACMC06 invites the submission of musical works that reflect the nature and diversity of computer music. These can include:

- * Electro-acoustic works
- * Real-time processing
- * Computer assisted compositions
- * Sound Installations
- * Submissions that explore the conference theme are encouraged

Presentation configurations and formats

ACMC 2006 welcomes submissions of recorded, acoustic and mixed works with any combination of electronic and acoustic elements. Playback formats available include:

- * Stereo works on CD
- * 5.1 authored DVD video and DVD audio
- * AC3 files on optical media
- * Multichannel - such as 4 and 8 works as individual mono files on CD/DVD
- * Live electronic works
- * Other replay formats will need to approved by the concerts selection committee
- * Information for works requiring musicians will be provided shortly

Submission requirements and formats

All submissions will include the following:

- * Completed musical works submission form (postal submissions only)
- * Recorded work (please include a 2 channel downmix of multichannel works for audition purpose). Preferred format for audition is audio CD.
- * Liner notes (250 words or less)
- * Artist's biography (250 words or less)
- * Complete scores for acoustic performers (as necessary)
- * Technical requirements list
- * Any submitted materials such as DVDs and CDS should be labelled both on the case and disk
- * Please note this selection process is separate from other submission selections

Papers

Due Date: Tuesday, 18 April 2006

Submit: http://www.acmc06.org/registration.html

ACMC06 invites the submission of papers for presentation at the conference and peer review and publication in the conference proceedings. Topics for papers include but are not limited to:

- * Experimental music with computers
- * New sounds and new sources
- * Computer assisted composition
- * Digital instrument building and software development
- * Interactive performance
- * Computers and creativity
- * Submissions that explore the conference theme are encouraged

Submission requirements and formats

Papers may be in one of the following categories:

* Research Paper - Fully refereed (6 - 8 pages)
* Paper - Abstract refereed (500 word abstract)
* Please note this selection process is separate from other submission selections

Papers should be formatted using the ACMC Paper Template

Posters

Due Date: Tuesday, 18 April 2006

Submit: http://www.acmc06.org/registration.html

Posters that describe work in progress or smaller scale studies are encouraged, these may be particularly suitable for presentation of student projects.

Topics for posters include but are not limited to:

* Experimental music with computers
* New sounds and new sources
* Computer assisted composition
* Digital instrument building and software development
* Interactive performance Computers and creativity

Submission requirements and formats

* To propose a poster please submit 500 words outlining your project
* Please note this selection process is separate from other submission selections

Artist talks

Due Date: Tuesday, 18 April 2006

Submit: http://www.acmc06.org/registration.html

An artist talk is a short presentation given during the conference in a manner similar to a formal paper presentation. We wish to encourage opportunities for artists to discuss previous, ongoing or developing projects, with particular emphasis on projects that relate to the conference theme and topics. Abstracts or short papers will be published in the conference proceedings, as provided by the presenter.

Submission requirements and formats

* To propose an artist talk please submit a 500 word abstract outlining your work
* Please note this selection process is separate from other submission selections

Studio reports

Due Date: Tuesday, 18 April 2006

Submit: http://www.acmc06.org/registration.html

A studio report is a short presentation given during the conference in a similar manner to a formal paper presentation. We wish to explicitly encourage discourse on the following topics:

* Case studies of computer music facilities in primary and secondary school music education
* The role of computer laboratories in tertiary institutions

Abstracts or short papers will be published in the conference proceedings, as provided by the presenter.

Submission requirements and formats

* To propose a studio report please submit a 100 word abstract outlining your presentation
* Please note this selection process is separate from other submission selections

Workshops

Due Date: Tuesday, 18 April 2006

Submit: http://www.acmc06.org/registration.html

A workshop is an educational session run by a conference delegate or partner on a particular area of their expertise. There will be opportunities for workshops the day following the conference. Proposals for workshops can be on any topic likely to be of interest to the conference participants.
Submission requirements and formats

* To propose a workshop please submit 500 words or less detailing the topic, scope and likely resource requirements for the workshop
* Please note this selection process is separate from other submission selections