HOME GROWN

Conceived, concocted and cooked-up on these very shores, AudioMulch 2 is winning a swarm of friends both at home and abroad. We talk to its creator, Ross Bencina, about his inspiration for the program, and the elbow grease required to get a world-class application off the ground.

Text: **Brad Watts**

Earlier this year I had the pleasure of meeting Ross Bencina, code-creator extraordinaire, and the brains behind AudioMulch. For those of you who've never heard of the program, this fine audio application melds the concepts of personal computer-based production and digital signal processing into a cohesive tool for performers and recordists looking to push the boundaries of live performance and digital improvisation.

But rather than interrogating Ross about the ins and out (as it were) of AudioMulch, Ross and I chatted about his background in live performance, and what it takes to get a program like AudioMulch off the ground and into the hands of eager-to-improvise musicians. As it stands, AudioMulch has already made permanent inroads into both performance and recording spheres with acts like Girl Talk and Four Tet, and luminaries such as Pete Townshend and Trent Reznor, embracing the program wholeheartedly. For a greater insight into to the nuts and bolts of AudioMulch, see the roundup of features overleaf.

PREPARING THE SOIL

Brad Watts: Ross, can I kick things off by asking you what's led you to becoming involved in software development?

Ross Bencina: Well, to start at the beginning, back when I was a teenager I guess I had two real interests: music and computers. Back then I had a MIDI keyboard – an Ensoniq EPS-16 Plus – and a Mac for sequencing. That's basically how I got into programming. I was very interested in digital sound as a kid, and wrote very basic programs that generated waveforms and the like. I guess it's fair to say I was a bit of a teenage computer nerd.

After high school I went on to study music at La Trobe Uni, which, back in the early '90s, had a great music technology course. This was back in the days when the sequencing programs being taught were Opcode's Vision and Mastertracks Pro. Then there was sound synthesis software like C-Sound… but the really big deal was the fact that La Trobe had a NeXT machine running IRCAM Signal Processing Workstation software – that was super high-end research signal processing stuff.

BW: Was all this sequencing and signal processing easy to manage with a personal computer at that time?

RB: In truth, it was really only beginning to become practical . This was around the time when Apple had just released its PowerPC Macintosh. The first real-time processing I did was on the NeXT platform. These machines were completely discontinued by that stage, but I'd bought a second-hand one regardless, mainly because they had built-in DSP chips. I hacked something together to process some real-time sound with that.

So I was generally neglecting my studies and spending most of my time hacking with computers. That's how it all began really. There's a lot to learn when you're trying to program computers, and it can be a tedious slog at times, but when you've got a real interest in achieving a specific goal – like I

had – you pursue things until you get the one you're after. It was a little bit harder back then, of course. The internet was only just starting up, and gathering information was a little harder. *Some* things could be garnered from the net, but information gathering mostly involved asking people via email how they were doing things. It wasn't like it is now with Google search engines and forum posts. You actually had to contact someone directly to get an answer. Having access to the university library and 20 years of research publications about the formative years of computer music production was probably the biggest advantage I had.

BW: Was the latency monster an issue for you back then?

RB: Latency wasn't too bad. It certainly wasn't good, but it was manageable. I think it was around 20 milliseconds or similar, which is noticeable, but manageable. Like many others though, I saw the potential of computers to do these things, so I was confident the latency issues would become easier to manage over time. The IRCAM hardware I was using at university had less than two milliseconds latency, so I knew the technology would eventually catch up.

COMPUTER MUSIC

BW: How did all this coding relate back to your musical interests?

RB: I guess it all related back to my interest in samplers and hardware synthesisers. This was all occurring around the end of the '80s when the 'analogue' sound was emerging from the dance scene, which is what I was interested in, along with the musique concréte movement. But the technology was also a bit hidden back then, and people seemed to regard it as some sort of esoteric format from a bygone age.

By the early '90s, my perceptions were being bombarded by marketing about digital synths and sample-based systems, like the Roland D-50 and the Yamaha SY series – the analogue-modelling concepts hadn't really begun at that stage. So there was a vacuum of sorts, as I saw it. I wanted to create sounds with the sort of attitude most people now associate with modular 'analogue' synthesis.

> **"It took me a long time to I wanted the kind of fluidity that invites complete improvisation, not just knob-twiddling over already established processing chains. Mulch was all about having the ability to patch stuff together *live*."**

For me, the first really significant machine at the time was the Kurzweil K2000. I remember thinking, "whoa, this thing does everything and it's really deep and powerful."

BW: So when did this focus morph into an ambition to build AudioMulch?

RB: Being interested in processing sound as a musical entity, rather than for sheer experimentation, I started writing programs with some musicality behind them. The first was called Oversyte – a real-time sound granulation program I wrote in about 1994 for the PowerPC Macintosh. Actually, that's not strictly correct; I'll confess the first program I wrote along these lines was a TB-303 Bassline simulation. That floated around for a while between friends and work colleagues, but was never 'released' as such, although it did end up being shoehorned into AudioMulch eventually. But with Oversyte there was a project deadline. I had a gig lined up that involved processing the Astra Choir – a chamber music choir in Melbourne – which had been organised between myself and a few other people at La Trobe. The performance involved three improvising vocalists, although we did rehearse quite a bit so we knew what style of processing we'd be doing.

Leading on from that I collaborated with flautist, Kylee Smith, doing live improvised sound processing with the same software. In a lot of ways that's what got me started with what I do nowadays – live performances using improvised sound. As a result of the collaboration with Kylee Smith, we

ended up doing a series of one-hour fully improvised performances at the Adelaide Fringe Festival. From that point on I was hooked.

The intense improvisational nature of these performances led me to develop a huge laundry list of the tools I'd need to continue. At the time I had software doing some sound processing, while other elements of the performance were pre-prepared and pre-processed as stems in ProTools. These stems would often include pre-production with non-real-time software like C-Sound and C-Mix, which are command line processing programs – completely off-line and non-real-time.

So I guess the turning point for me was realising that I needed software that was more modular; far more configurable for real-time performance, and completely real-time. I wanted to improvise and I wanted to plug lots of stuff together and do it all on-the-fly. I wanted the kind of fluidity that invites complete improvisation, not just knob-twiddling over already established processing chains. Mulch was all about having the ability to patch stuff together *live*.

That concept in itself wasn't a new idea, of course. People had been doing that throughout the '70s with hardware modular synths, but it was nevertheless an idea I really connected with: improvised performance where everything is completely dynamic, with no pre-composition whatsoever.

KEY IDEAS

BW: Can you tell us a bit about the hard work involved with getting a program like AudioMulch developed for commercial release?

RB: Software stability is the main issue with AudioMulch, without a doubt. Getting the program to the point where it's stable for all users – not just for me personally – has been the biggest challenge. I remember early on in Mulch's development, the bug list was literally in the hundreds. That was a real struggle that demanded my constant attention. But my attitude to the program was always: 'Okay if I want this thing to be rock solid I need to be paying attention to these bugs, and if someone says it's crashing then it's not something to just add to a list and think about later'.

It's hard to describe, but basically I like to think of software as a big clockwork mechanism, with a lot of little inter-meshing components all working together. Not just gears, but chains and pulleys and weird tracks where balls run down slots and countless switches turn on and off. Probably every kind of mechanical thing you can think of is in there, but it all has to sync together and function properly. But unlike a clockwork mechanism, where you can sit back and observe the whole thing, with software, it's all electrons flowing through a microscopic circuit. So when something goes wrong, you can't see what's occurred. Even the methods and tools used to find the bugs themselves can be unreliable, which can be extremely frustrating at times.

BW: Presumably you've got to hunt down and tag these bugs somehow?

RB: Yeah, if the software crashes you've got to find out exactly what happened and why. I guess you could liken it to paper jamming up a printer mechanism or a trolley going off the rails on a production line. With software, beyond simply finding it, the real trick is to find out why it went off the rails in the first place. It could be that the rail just led off into oblivion, in which case it's no wonder the trolley fell off, or it could be any number of other less obvious possibilities. One thing can lead to another inside a program, and in an instant, the entire application falls over.

BW: User bug reports must be invaluable to you then?

RB: They are, absolutely. And the program wouldn't be where it is today without that feedback. I've kept a close relationship with the AudioMulch community from the beginning, and the program has really benefited from that. I've always encouraged people to contact me with bugs. That sort of community-based testing process has been a big part of my development process. I mean, if you're a big company like Microsoft you can afford to have whole teams of people doing the testing for you, and you've also got a heavy investment in automated testing. But I'm essentially one guy and I don't have that capacity.

CLEAR AIMS

BW: It sounds like one of the things that must be hard is simply keeping the program on track, and resisting the temptation to try to make it 'all things to all people'. Is it?

RB: I think it's very important, when writing a program, to have a clear idea of what your goals are. That's half the struggle with making anything, really, and it's crucially important when writing software. To be honest, that's something I've learned over time; it's not something that's always been foremost in my mind. It was a lot easier

AUDIOMULCH: WHAT IS IT?

AudioMulch is sometimes hard for people to grasp, so imagine it this way. Close your eyes and picture your favourite rehearsal space. The floor is cluttered with chains of esoteric effects pedals, each tied to their own instrument and amplifier via cables. There's a keyboard player taking up a third of the room (and even more of the mixer channels). He's surrounded by a modular synth, drum machine, sampler and controller keyboard, so there's only a few channels left on the console for vocals, trumpet and percussion mics. A MiniDisc deck in the PA rack is documenting the jam – if you're lucky – and once you're all connected, that's it. Everyone's in his or her own channel and space – locked In.

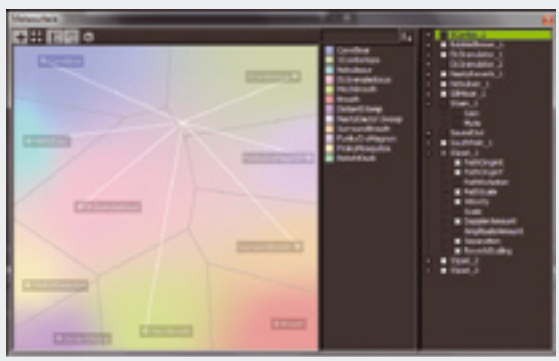
But what if, while you were playing, the guitar could be repatched via the filters in the synth, the trumpet routed through the fuzz pedal and tape delay, the drum machine used to gate the keys? What if the vocal recording from last week's jam could be pumped back into the Fender Twin from the MiniDisc; all on-the-fly, all without interrupting the audio and all spontaneously? What if all this could be done during a single performance, reconfigured to another combination and then returned, all before the coda? This is the domain of AudioMulch.

AudioMulch is an 'interactive modular environment' for improvised performance, composition and sound design. Synthesis and processing can be limited to within a single computer, or alternatively, live instrumental performances can be patched into the program via audio interface for direct interaction. Several laptop performers can even interact in this way running multiple instances of AudioMulch; all locked together via MIDI or network sync. The resulting performances can then be captured as a multichannel audio recording or automation curves to be replicated later or refined.

The process is as simple as plugging two modules together with the program's virtual patch leads, so it's not surprising AudioMulch has been used extensively to teach students about audio signal flow and processing. Almost all of the program can be controlled via MIDI, so the concept of performance can become even more physical. AudioMulch emphasises finding your own non-linear path and removes the conventional structure imposed by the discrete multichannel environment of most DAWs. Feedback loops can be combined with Live Looping processes to create self-generating and evolving soundscapes.

AudioMulch is, above all, a computer instrument. It brings together elements of traditional analogue modelling with effects and routing options only possible within the computer.

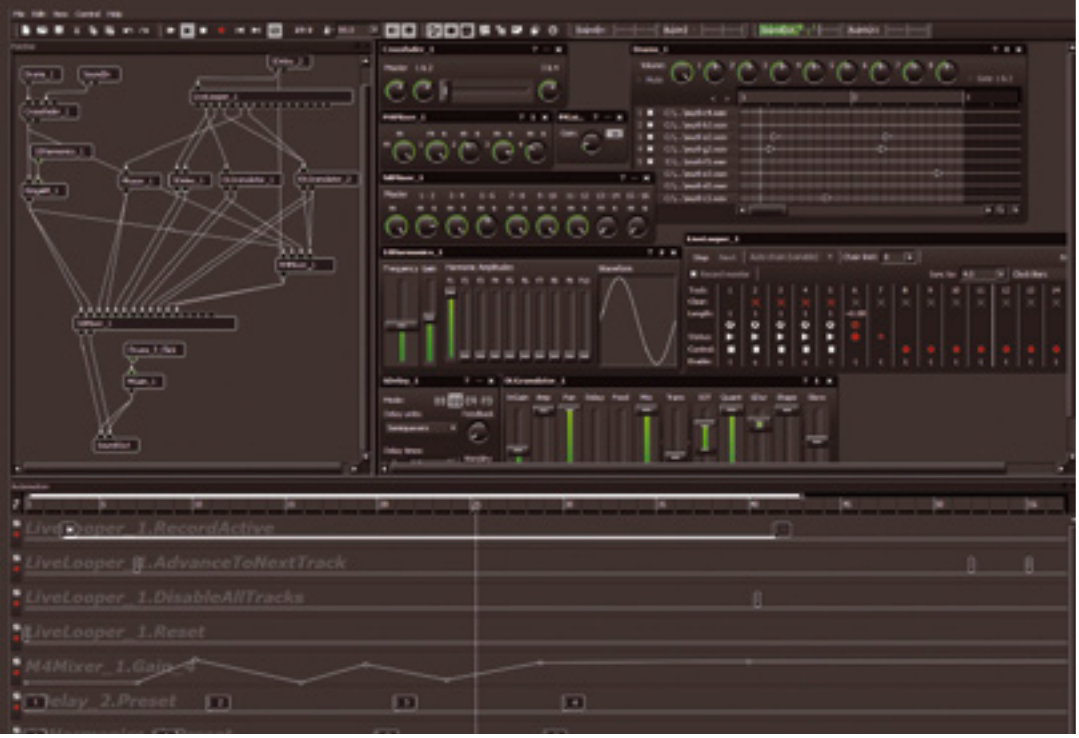
The Metasurface: The Metasurface is a control interface unique to AudioMulch, which allows for the manipulation of many parameters at once through fluid mouse or controller gestures. By placing snapshots of settings around the two-dimensional plane the performer



is able to create a 'geography' of effects. Moving between snapshots with the cursor then smoothly morphs these parameter values, creating shifting sonic effects. The Metasurface can be controlled using an X/Y MIDI controller (Korg Kaoss Pad etc), two individual MIDI controllers mapped to X and Y parameters, or simply with the mouse. If you've got the appropriate interface software you could even use a Wacom tablet, WiiMote or multi-touch device.

Coming Attractions: AudioMulch 2.1 (a free upgrade for registered users) is slated for release in coming months. Aside from the usual host of user improvements and optimisations, 2.1 promises a suite of dynamics processing effects (a compressor, limiter and gate), support for non-4/4 time signatures, and AudioUnit plug-in support for Mac users. To read about and participate in shaping the future development trajectory of AudioMulch visit:

www.audiomulch.com/audiomulch-roadmap-2009-2010



AudioMulch is an 'interactive modular environment' for improvised performance, composition and sound design.

Ross Bencina: "The real challenge for me is in promoting the fact that AudioMulch isn't for everyone and doesn't do everything."

when I started building the program for my own performances, before I went public with it. Even after I first started calling it AudioMulch, sharing it over the web, and making it available for other people to use, it was foremost in my mind that this was software I was writing for myself, and that I had to be happy with its performance.

Now that the program has been around for more than a decade in some form or other, the things I mostly think about are issues like marketing – communicating the value of the product I know so well to people who may want to use it. It's very much about communicating clearly what it is I'm trying to do, and being clear about what AudioMulch is designed for. As you say, it's very easy to just grab onto whatever the latest hype is and attempt to ride that bandwagon, whatever it happens to be. The real challenge for me is in promoting the fact that AudioMulch *isn't* for everyone and *doesn't* do everything. It's not a tracking program and it's certainly not a DAW.

I still remember the first time I got an email from someone saying, "Hey man I love your product." I was quite confronted by the concept of my software being referred to as a 'product'! I didn't relate to it like that at all. These days I can look at the program in that light, and I think that's helped a lot with keeping the concept of AudioMulch clear in my own mind, and consequently relay that concept more effectively to the user-base.

BW: What's the estimated size of your user-base now, do you know?

RB: That's a difficult question to answer, mainly because of software piracy. I don't actually think most of the downloads are from my website. The other day I was at a bar listening to these guys from Argentina play and one of them was doing this laptop experimental stuff and I said, "I liked your set… I make audio software, a program called AudioMulch." He was like, "AudioMulch! Wow! I use that software all the time, but I didn't pay for it. I always use a pirated version!" So I think there's a lot of people like that out there. Fortunately, there are enough people using legitimate versions to allow the program to continue.

FUTURE DEVELOPMENT

BW: So what does the future hold for AudioMulch?

RB: There's plenty of stuff planned for this year, but in terms of crystal ball predictions, the future for Mulch is all about creating something that's easy to use and even better suited to live performance. Performing live with laptops is much more commonplace nowadays than when I started with AudioMulch. Back then I was carting a desktop computer around to gigs, so I think there's still a lot of potential in paring down the equipment required for processing audio. I'm also very interested in physical interfaces – I actually spent some time in Spain, working with the research team that developed reacTable (www.reactable.com) – getting away from that sitting-behind-a-computer-with-a-mouse paradigm. Things like the Novation Launchpad, the Monome, and so forth all look to be viable alternative control systems for live performance. I see that whole interface trend as something that's going to expand greatly during the next few years, and I'm interested in taking that beyond mere control surfaces, into wearable and movement-sensing controllers, taking things more in the direction of the WiiMote idea. I've actually done some experiments using WiiMotes to control AudioMulch already – there are videos of that stuff online if you want to check it out. It's looking quite promising. Whether that's something I just end up doing for my own musical interest, or something that becomes part of AudioMulch, or indeed a separate product altogether, I'm really not sure yet. ■

NEED TO KNOW

Price
AudioMulch 2: US$189 or $189 to Australian residents via email: (info@audiomulch.com)

Contact
info@audiomulch.com
www.audiomulch.com

Supported Operating Systems

Windows XP, Vista, Windows 7.

Mac OSX (Intel processors only) 10.4, 10.5, 10.6.

Any computer capable of running a supported OS.

A fully functioning 60-day trial version of the program is available from the AudioMulch website.

Key Features

AudioMulch contains a range of built-in signal generation and sound processing modules, including studio and performance classics like delay, parametric EQ, reverb, phaser, flanger, a drum machine and arpeggiator, as well as unique digital synthesis and processing effects like granulators, shifters & shapers, risset filters and tone generators...

There's multitrack live looping functionality, a unique Metasurface effects morphing interface, timeline automation of contraption and program parameters, and VST plug-in support for further expansion of the audio and MIDI processing possibilities.

There's support for up to 256 channels of real-time audio I/O, multichannel file recording and playback, along with playback and recording support for WAV an AIFF files up to 32-bit/192k.

There are assignable MIDI input ports, patchable MIDI routing, along with quick mapping of MIDI controllers to contraption and program parameters. There's support for MIDI & network sync with other devices and MIDI control of document switching.