Alex McLean and Adrian Ward

Live Coding: I think in text

Alex McLean

Alex McLean hacks Perl for the state51 conspiracy, is half of the live code band "slub" and also part of a new collaboration called "table".
http://yaxu.org/

The term 'live coding' is usually meant to describe the coding of music on the fly. It seems a process of unveiling the (running) machine to manipulate it, resounding accordingly. Which are your main concerns while performing live?
When it's good I have no concerns, and can just get on with developing the music. I'm really just switching focus between what Ade and Dave (the other 'slub' members) are doing and what I'm adding to that. Whether I need to stop doing something to give them room, or whether they're reaching a conclusion with their stuff and I need to get ready to take the lead with some new code.

Live coding is "deconstructing the idea of the temporal dichotomy of tool and product" as it's stated in the TOPLAP website. So the tool mutates into a product. In your opinion is it regaining its status of magmatic digital data? Or is it mutating into a hybrid powerful machine-oriented code?
I'm not sure what you mean by 'magmatic digital data'. I think though that live coding isn't about tools or products, but instead about languages and musical activity. Tool doesn't really come into it.

With the commercial model it goes:

[code] -> compiled -> [tool] -> used -> [music]

the dichotomy comes because the person making the tool is different from the person making the music.

With livecoding it goes

[code] -> interpreted -> [music]

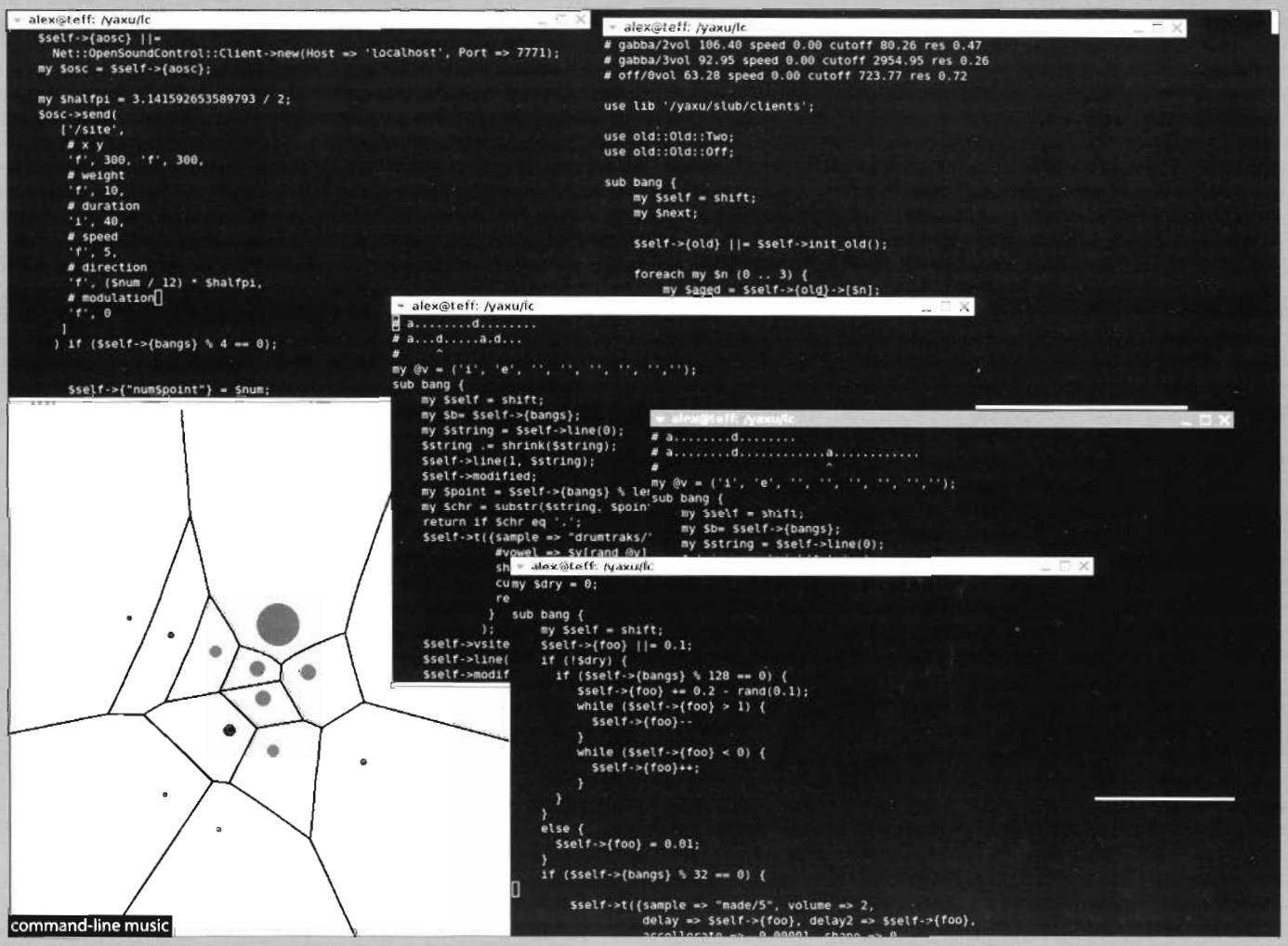
where the code is modified to change the music

So the code and the music comes closer together by missing out the tool stage. Of course the big secret about the commercial model is that a lot of the music comes from the code, as compiled into the tool. As Kim Cascone says, "The tool is the message." Well in the case of livecoding the tool isn't the message - there is no tool. The code is the message!And the music is the message... And the music is the code...

And how do you feel the ambivalent code evolution / music so generated? Is it a parallel (but conceptually linked) flow or a digital cause/effect relationship?
There is a feedback loop. The livecoder writes code, that makes sound, which the livecoder hears and perceives as music, which they then react to, by editing the code. The code is a kind of notation for the music. Unlike traditional notation the code describes the sound completely, because it is read by a formal interpreter that is in turn described as code. Lovely!

Changing the code while it runs seems similar to composing phrases on the fly (as we humans are used to do). Do you think that live coding has some 'semiotic' characteristics that can be compared to poetry live improvising?
No, but I believe it will go in this direction. In fact I have become very interested in articulatory speech synthesis, which makes sound from models of the human body. My current research project is to apply techniques from speech synthesis to musical sounds, not necessarily human-like sounds. There is rich history of people talking about writing down musical sounds as 'vocable' words, for example Canntaireachd for bagpipe and Bols for tabla sounds. I want to make a synthesis system for livecoding so I can type the word "krapplesnaffle" and have it turned into sound and immediately placed into livecoded rhythmic structure.
[http://speechless.lurk.org/] contains my experiments towards this...

Another key point of live coding performances is to have no backup (MiniDisc, DVD, safety net computer). Is this meant to legitimate the (eventual) accident as a


alex@teff: /yaxu/lc
$self->{aosc} ||=
Net::OpenSoundControl::Client->new(Host => 'localhost', Port => 7771);
my $osc = $self->{aosc};
my $halfpi = 3.141592653589793 / 2;
$osc->send(
['/site',
x y
'f', 300, 'f', 300,
weight
'f', 10,
duration
'i', 40,
speed
'f', 5,
direction
'f', ($num / 12) * $halfpi,
modulation
'f', 0
]
) if ($self->{bangs} % 4 == 0);
$self->{"numspoint"} = $num;
alex@teff: /yaxu/lc
gabba/2vol 106.40 speed 0.00 cutoff 80.26 res 0.47
gabba/3vol 92.95 speed 0.00 cutoff 2954.95 res 0.26
off/0vol 63.28 speed 0.00 cutoff 723.77 res 0.72
use lib '/yaxu/slub/clients';
use old::Old::Two;
use old::Old::Off;
sub bang {
my $self = shift;
my $next;
$self->{old} ||= $self->init_old();
foreach my $n (0 .. 3) {
my $aged = $self->{old}->{$n};
alex@teff: /yaxu/lc
a........d........
a...d.....a.d...
#
my @v = ('i', 'e', '', '', '', '', '','');
sub bang {
my $self = shift;
my $b= $self->{bangs};
my $string = $self->line(0);
$string .= shrink($string);
$self->line(1, $string);
$self->modified;
my $point = $self->{bangs} % le
my $chr = substr($string, $poin
return if $chr eq '.';
$self->t({sample => "drumtraks/
#vowel => $v[rand @v]
sh
cu my $dry = 0;
re
} sub bang {
); my $self = shift;
$self->vsite $self->{foo} ||= 0.1;
$self->line(if (!$dry) {
$self->modif if ($self->{bangs} % 128 == 0) {
$self->{foo} += 0.2 - rand(0.1);
while ($self->{foo} > 1) {
$self->{foo}--
}
while ($self->{foo} < 0) {
$self->{foo}++;
}
}
}
else {
$self->{foo} = 0.01;
}
if ($self->{bangs} % 32 == 0) {
$self->t({sample => "made/5", volume => 2,
delay => $self->{foo}, delay2 => $self->{foo},
alex@teff: /yaxu/lc
a........d........
a........d............a............
#
my @v = ('i', 'e', '', '', '', '', '','');
sub bang {
my $self = shift;
my $b= $self->{bangs};
my $string = $self->line(0);
alex@teff: /yaxu/lc
command-line music

part of the performance?
Yes, a little danger is good, adding an edge to the performance both for us and the audience... There are three of us though, and if one of our systems go down the others can take over. Then the audience has some fun watching our boot up proceedure :)

You also used to play live (as half of 'slub' band) with your own 'command-line music'. Why you choosed to use the minimal command line interface? Which software was involved?
Correction: since 2006 there are now three of us: Adrian Ward, Dave Griffiths and myself Alex McLean. I use the UNIX shell because I think in text. It's fast, there's a beautiful relationship between data and code, and it's easy to recall and modify past actions - you don't have to repeat yourself all the time like with GUIs. When interactive commandline shells were first developed they were called "conversational languages," part of a field of research called "conversational computing." It's a shame that this terminology fell out of use.

What's 'moving' in your performance is not an arm that plays a violin, but the shape of your algorithms, forcing your fingers to move fast on the keyboard. Even if this is barely seen by the audience, there's a gesture, more evident and theatrical than the usual laptop performance. How important do you consider the gesture in your live set?
Well, livecoders always project their screens, so people can see the typing gestures, which I think are really beautiful even if you can't see the fingers that are typing them. Jaromil and Jodi's "time based text" (http://tbt.dyne.org/) highlight this really well. As there are three of us improvisers there are human gestures between us too. I think all this is important, if you can see someone is on a stage, but you can't see any movement making the music, then there is no performance.

Performing live coding, you feel to purely 'improvise'? Here, do you feel a substantial difference with the improvisation music school? If yes, which one?
Improvisation is the creation of work while it is being performed, so it's clear that livecoding is a form of that. I have had really enjoyable improvisations with vocalists, guitarists, rappers and drummers as well as other laptopists, so don't see much difference. The only real difference is that livecoding is quite new, and I think has a bit more developing to do...

TOPLAP (whose acronym has a number of interpretations, one being the Temporary Organization for the Proliferation of Live Audio Programming) is advocating live coding practices in different areas. In its 'draft' manifesto it's written: "Programs are instruments that can change themselves." Do you think that software is the ultimate music instrument?
No, I don't. I think computer languages are great mediums for making instruments though, and livecoding allows you to change those instruments while you're playing them in some interesting ways. But you can make amazing sounds with an egg whisk. Who am I to say that Perl or Haskell is better than an egg whisk? In fact if I was to pick an ultimate instrument I think the human voice would be it.

The TOPLAP crew also stated that they advocate the "humanization of generative music." What's wrong with 'classic' generative music software?
According to Brian Eno, generative music is like making seeds and sitting back seeing what they produce. There's nothing at all wrong with this idea, I love gardening. But livecoding is something a bit different - it's instead more like modifying the DNA of plants while they're growing, by hand. In this way, generative music is nature and livecoding is nurture, in fact it's possible to have a combination of the two.