What is the question to which Live Coding is the answer?

Keynote speech for the Coded Matter(s) symposium – 23 March 2014
by Marcel Wierckx

First, I want to acknowledge what a thrill it is to be giving the keynote for this Coded Matter(s) event at STEIM. As most of you probably know, STEIM was the venue for the very first live coding performance ever: a half-hour performance by Ron Kuivila in 1985 using the programming language FORTH, which, incidentally, ended with a system crash[1].

My first exposure to Live Coding was in 2005, and I was immediately smitten. At the time, I was heavily involved in performing live computer music, and the whole idea of Live Coding suddenly put everything I had accepted as 'the rules' for performing live electronic music under a microscope. I was reminded of Robert Henke's famous line: "The last century was about how to *create* electronic art. This century is about how to *perform* it."

All aspects of live performance in the broadest sense – talent, skill, virtuosity, risk, concentration, excitement, transience – come together in Live Coding in a way that no other performance art can boast.

Most performance arts can be considered as existing 'within brackets' (in the literary sense): the performance is part of the story and yet separated from it. The brackets act like a thin shell between the larger narrative in which the audience exists and the smaller narrative in which the performer exists. The most exciting modes of performance however do not have this shell. Think of the most memorable jazz improvisation you've ever seen live. It probably affected you because you shared the same cognitive space as the performer, even if it was just for a moment. Either you were invited in to their brackets or they escaped theirs.

The primary appeal of Live Coding to me has always been this sharing of cognitive space. It's only possible because Live Coding performances lack the brackets we normally associate with performance arts. When Live Coders perform well it is as if there is a magnifying glass upon the reality of the moment: the fragility, the excitement, and the authenticity of the performance become fully transparent and sharply focused. A good Live Coder does not need to get 'into character' before a performance– the Live Coder's performance *is* the person.

But enough by way of introduction.

There are two things I want to talk about here today: first, what is the question to which Live Coding is the answer, and second, how can we broaden the appeal of Live Coding (and why should we)?

To understand the significance of this first question we need to remind ourselves of Kim Cascone's critique of the emerging genre of Laptop Music, stated in an article from 2002. He writes:

> Music performed on a laptop is lacking in one element: its unique existence at the place where it happened to be created ... the performance feigns the effect of presence and authenticity where none really exists. The cultural artifact produced by the laptop musician is then misread as 'counterfeit,' leaving the audience unable to attach value to the experience. ... it is difficult for an audience to perceive the value of a performance where the artist could simply be playing back soundfiles... Consequently, the standard codes of musical performance are violated: the laptop is doing the work, no skill is required or demonstrated, and the artist could just as easily be any one of the audience faking a performance. [2]

A similar critique, targeted at the digital arts in general, was voiced by Kurt Ralske in 2004. He writes:

> For the classical pianist, the tedium of endless hours of practicing scales takes on an aura of nobility; it's a virtuous, character-building activity. Instead of practicing scales, the digital artist learns software and hardware, learns programming languages, learns the techniques of creating digital models of sound, image, information, and intelligence. *But generally, the 'aura of nobility' that falls to the studious pianist does not currently fall to the disciplined digital artist.* Currently, the attitude of the digital art world seems closer to the punk aesthetic: concept is everything, technique is secondary: a distracting necessity. There is perhaps even a prejudice that digital technique and artistic vision are mutually exclusive: an individual can have one or the other, but not both. [3]

Cascone and Ralske both captured a sentiment that was in everyone's thoughts at the time and expressed it succinctly. And lots of us were searching for ways to address these issues at the time. I looked for the answer by exploring the ways a laptop could be used as an extension to acoustic instruments, something that I continue to do now with my students at the Conservatory of Amsterdam. Others invested heavily in developing novel controller technologies, like Eboman's sensor-packed body suit [4], or took a radical approach, like Hans Koch and his literal laptop instruments: the Bandoneonbook and Electroviola [5]. Gradually however a view of the computer as a general-purpose device, as simply a link in a larger performance system, seemed to become pervasive.

That's why the answer that Live Coding provided was special, because it allowed us to keep the focus on the laptop as an instrument *in and of itself*, rather than seeing it as some kind of extension or appendage to some other more important instrument or device. And more significantly, it saw the laptop as an enabler of artistic expression as realized intellectually rather than physically – a means to share that cognitive space that I referred to earlier.

To come back to Cascone, Live Coding seems to fulfill his demands for returning live electronic music back to artistic growth: it re-engages the historical context of performance

practice by reintroducing spectacle as a guarantor of presence and authority, it builds an awareness of audience expectations, and, perhaps most importantly, it develops non-distracted modes of reception in an audience. [2]

Whether or not Live Coding fulfills Ralske's demands on the digital artist is less certain. I will return to Ralske in a moment, when we get to the second topic of this talk.

And so now, roughly ten years since the expansion of Live Coding into a feasible mode of performing computer music for a lay audience, attention has suddenly been focused on it from all directions. Recently I was approached by a mainstream news network here in the Netherlands who were excited about the idea of the Algorave and wanted to know everything about Live Coding for an article.[7] The music culture program *Tracks* aired a piece on Live Coding and Algorave on the TV channel Arte. Students who were once at best indifferent to the whole idea of writing computer code because of its inherent nerdiness and perceived difficulty are now asking me to give courses in Live Coding. And the renewed interest in the activities of STEIM, the rise of festivals and events dedicated to artistic coding in all its forms, and the slow yet noticeable shift in gender balance at Live Coding events, are all indications to me that a good thing is happening, right now.

Not all the attention is positive however.

Can you imagine my dismay at reading Jacqueline Oskamp's dismissive remarks about Live Coding in her 2011 book *Onder Stroom,* a history of Dutch electronic music.[Ambo/Anthos, 2011] She summarizes our craft as "technical showing-off" that delivers an "impoverished musical result" (*"technische spierballenvertoon" levert "pover muzikaal resultaat"*). The implicit criticism here is that Live Coders are an elitist club of specialists that show off their ability to do really difficult things without regard to whether or not the result is musically interesting.

Oskamp's comments were even published in the widely read Dutch newspaper NRC Handelsblad.[8] Now, I'm familiar with the concept 'all press is good press,' but I'm not going to be facile about this. There are two possible reactions to this criticism. The most tempting is to agree wholeheartedly with Oskamp, and wear that elitist hermeticism proudly as a badge of accomplishment and legitimacy. But if there's just one thing that the Dutch artistic community has learned in the past few years, it's that this approach just won't fly anymore.

So then the other possible reaction to Oskamp brings me to the second topic of this talk: How can we broaden the appeal of Live Coding, and why should we.

To start on a personal note, I am proud to say that I am an amateur Live Coder. I am even willing to go so far as to admit that this keynote is in fact my very first paid Live Coding gig.

This is in no way an admission that I'm an inexperienced or inept Live Coder. On the contrary, I think I'm pretty damn good at it. It's merely the acknowledgment that I do Live Coding out of a passion and love of the craft, and that my interest in it is personal and not professional.

I say this because I believe that to take part in a community as an informed amateur is not only good but in fact essential for the health and growth of that community. In the words of Edward Said:

> ...the intellectual's spirit as an amateur can enter and transform the merely professional routine most of us go through into something much more lively and radical; instead of doing what one is supposed to do one can ask why one does it, who benefits from it, how can it reconnect with a personal project and original thought. [9]

But now to come back to Ralske, I feel that Live Coding still has to develop further before it can be said to fully address his remarks. There are a couple of reasons why I believe this to be the case. The most obvious reason has to do with the average audience member's difficulty in imagining for themselves what the craft of Live Coding actually entails.

Here's an analogy to clarify what I mean: ask a random selection of people if they've ever pressed the keys on a piano and listened to what happens. Chances are most or all will have done that. Hence, pretty much everyone can easily imagine the "tedium of endless hours of practicing scales" that bestows an "aura of nobility" to the concert pianist. One need not be a pianist to understand this.

Would we find a similar situation with laptop music? In a random selection of people you'd be hard pressed to find anyone who's never used a laptop. And I don't think it's a stretch to say that a significant number in that group would have tried making music on a laptop: we all know about the ease and ubiquity of music creation software. So perhaps it's not unreasonable to think that someone who's whipped together some drum and bass tracks (or whatever) in Ableton Live or Garage Band might not be so impressed by a Live Coder's frantic typing of code that results in some meagre bleeps.

But here I've intentionally mis-stated my analogy in order to highlight a general misunderstanding of Live Coding, a misunderstanding that we should take seriously. The more accurate analogy to my casual piano-key-presser would be with someone who interacts with the laptop in a more fundamental sense, and with this I mean at the coding level. Because there is absolutely no real correlation between pressing down a piano key and pressing down a key on the laptop's keyboard. The only meaningful correlation between my two examples, and the correlation that we should be making, would be between pressing a piano key and typing a line of computer code: giving the laptop an instruction in its own language.

But now the analogy is incongruous. In a random selection of people, how many do you think will have ever typed in and executed a bit of computer code? Not many. Hence the often bewildered reactions to Live Coding, even from audiences permeated by supposedly 'experienced' computer users. They watch someone do something which clearly involves intellectual exertion, yet what they hear is (to them at least) not substantially different from what they can do themselves with a couple of mouse clicks. Bye bye aura.

Let me return to my concert pianist. Human beings are inherently lazy, and pianists are no exception. Therefore if a pianist can find repertoire that requires fewer tedious hours of practice yet still awes an audience into believing in their aura, they will be tempted to favour that repertoire over repertoire that is more difficult to play but less clearly impressive for an audience.

I contend that the laptop performers ten years ago were no different. Since the audience could not tell the difference between a bleep produced from a soundfile and a bleep produced by a line of computer code, it was tempting to take the shortest path to audience satisfaction and easy aura. Hence the Toplap mantra "Obscurantism is dangerous. Show us your screens!"[6]

But I am not convinced by one line in the Toplap manifesto:

> It is not necessary for a lay audience to understand the code to appreciate it, much as it is not necessary to know how to play guitar in order to appreciate watching a guitar performance. [6]

What is 'knowing how to play the guitar'? Knowing a few chords? Scales? Pieces? Being able to pluck a string, or even just being able to *imagine* plucking a string? Is zero-knowledge of guitar playing actually possible? There is a big difference between knowing how to play the guitar and knowing how a guitar is played. The former is not necessary to being able to appreciate a guitar performance, the latter is. Do you see what I'm saying: the minimum amount of understanding necessary to appreciate a guitar performance is so negligible that it is difficult to conceive of anybody lacking such understanding. As my example above of the casual piano-key-presser, the aura of authenticity is properly earned by the virtuoso pianist because everyone understands the vast difference between simply hitting a key and making a great performance.

But there is no comparable minimal knowledge of coding that we can accept as universal in order to make this comparison. There is even a vast spectrum of misleading levels of so-called 'expertise' with computers as to make any assumptions meaningless. There are scores of people who have used computers for a large portion of their lives who have never executed a piece of code, who would probably not even know where to begin writing code. If we are to reach these people as an audience for Live Coding, we need to find a way to earn our aura. They need to have that minimal amount of understanding of coding, and we need them to challenge us to continually set the bar of virtuosity and authenticity higher.

To return to Ralske's dilemma, I ask the question: why is there a wealth of intricate, difficult and virtuosic piano music existing today? Because audiences would not be fooled. I would even go so far as to suggest that there might have been something like an arms race between pianists and audiences. The more expert the audiences became, the less likely it was that a performer could pull the wool over their eyes and fake an aura.

Why I am I telling you this? Because coding is making a comeback. Kids are learning to code in schools, a general interest in coding is expanding everywhere I look, and eminent scientists and tech billionaires are pushing policy makers to reintroduce fundamental computer science into all levels of education and society. This means that there are more coders coming, a more

critical audience is on its way, and we should participate in and embrace this new arms race in order to bring to life a wealth of intricate, deep, virtuosic and noble Live Coding practice in the future.

I also have a deeper reason for telling you this. Live Coding relies on systems and systems-thinking: it's what makes our craft possible. But if the post-digital period is, as Florian Cramer claims, a "reaction to the crisis of the paradigm of the system," [10] then we should be asking: where does Live Coding stand in the context of the post-digital period? Is it an "over-identification with systems," or is it a covert "rejection of systems," a rejection only possible by the hijacking of systems for an artistic purpose?

When I started using computers back in the '80's I never imagined that one day I would be asking myself if my computer was affecting my sense of agency, or that my own personal sense of agency could even be affected by a machine. At that time the computer seemed to be an enabler of independent thinking, a device that would empower me to realize new ideas and discover new forms of expression. But today I have a real love-hate relationship with the computer. While accepting its necessity for pursuing my craft, I struggle with the machine's capacity to nudge me into doing things I might rather not do if given the chance to reflect critically on the moment.

Long before Edward Snowden, writers and thinkers had already alerted us that somewhere along the way we stopped using computers and they started using us. Now I think we're finally waking up to that reality. The fear that we are unconsciously surrendering our own personal sense of agency to a system that sees us merely as means to an end has become real and relevant.

This is another reason why I am a fan of Live Coding: to me it represents a rejection of systems as dictating outcomes, and thereby a rejection of the way systems can be used as a method to enslave and stifle creativity. To me Live Coding has the potential to bend systems such that they can be used in the pursuit of personal notions of excellence and originality.

Some closing thoughts. Recently I've been working with some really fantastic jazz musicians who are exploring the field of live electronics. They've all told me the same thing: that the jazz establishment sees musicians that use live electronics as inferior, and that live electronics is only a way to mask the fact that they can't play their instruments. While I believe this to be patently untrue, the prejudice is real and prevalent. There is a real misconception that people gravitate to laptop music because they are not talented enough to make 'real' music. Ironically, this kind of attitude was prevalent in the early days of jazz. White musicians labelled black music with derogatory terms like 'scat,' 'funk' and 'jazz'. I am still waiting for an adequately inflammatory derogatory term to arise for live electronic music. Hopefully one that the community can adopt with pride.

Finally, when Aristotle was asked which musicians in society should be given the best-made flutes, he responded that naturally the best musicians should. His answer seems obvious to us, but his reason is revealing: the *virtue* of the flute is to be played, so the best possible playing is the fulfillment of the highest possible function of that flute. Let us ask: what is the virtue of a

laptop?

[1] Alan Blackwell and Nick Collins, 2005 "The Programming Language as a Musical Instrument" *Proceedings of PPIG05*
[2] Kim Cascone, "Grain, Sequence, System: Three Levels of Reception in the Performance of Laptop Music" *Contemporary Music Review:* Vol. 22, Issue 4. London: Routledge: 101-104
[3] Kurt Ralske, "The Pianist: A Note on Digital Technique" http://retnull.com/pianist.html (accessed 13 March 2014)
[4] http://www.eboman.info/wiki/index.php (accessed 13 March 2014)
[5] http://www.youtube.com/user/hwkoch (accessed 13 March 2014)
[6] http://toplap.org/wiki/ManifestoDraft (accessed 13 March 2014)
[7] http://nos.nl/op3/artikel/580854-nieuw-in-de-club-en-t-conservatorium-live-coding.html (accessed 13 March 2014)
[8] http://www.nrc.nl/handelsblad/van/2011/juli/15/alles-zit-nu-in-een-laptopje-12025844 (accessed 13 March 2014)
[9] http://downloads.bbc.co.uk/rmhttp/radio4/transcripts/1993_reith4.pdf (accessed 13 March 2014)
[10] http://www.aprja.net/?p=1318 (accessed 13 March 2014)