Proceedings of the 7th International Conference on Live Coding (ICLC 2023)
April 19th - 23rd – Utrecht, Netherlands

The Meaning of Live: From Art Without Audience to Programs Without Users

Alex McLean
Then Try This
alex@slab.org

Julian Rohrhuber
Institut für Musik und Medien
rohrhuber@protonmail.com

Renate Wieser
rwieser@mail.uni-paderborn.de

ABSTRACT

The concept of an 'art without audience' has informed live coding since its beginnings. Live Coding concentrates on collective work and questions the division between producers and consumers. This understanding of art has enabled a parallel strategy in the understanding of programming: just as an audience is not necessary for art, a user isn't necessary for programming. In the same sense as we question the separation between developer and user, we question the juxtaposition of artist and audience. This gives us occasion to recall some aspects of live coding which we have always found central to this practice: the displacement of the relation between programmers and programs, and the emancipatory potential of public thought.

1 Art Without Audience

Art is assumed to happen for an audience. Despite the importance that theory often gives to the authorship and agency of the observer or listener, still, most art is produced to be consumed. However, when in 1960 Kurd Alsleben experimented with an analog computer, it struck him that a computer drawing is not a product, but something that would become part of a conversation, a mutual exchange. Alsleben, together with Antje Eske, brought the concept of conversation art to life as part of the Hamburg data art movement of the 1980s, combining the ideas of 17th and 18th century salon culture with the possibilities of hypertext and network technology (Eske and Alsleben 2003). They took conversation art to be an art without a stage. As *art without audience* (*Kunst ohne Publikum*), it didn't need one.

Art without audience takes place in a semi-official space ("offiziöser Raum") between the personal and public sphere. As Eske and Alsleben pointed out at the 10th Chaos Computer Congress, this space *"[...] is the place that makes exchange possible. Data art, Net art is art without audience. There is nothing to be offered: no sender addresses a receiver. Rather what matters is forms/conventions, working out a common code."* [1] Just removing the spectator may seem to produce nothing but a lonely misunderstood genius. This practice, however, aims for something altogether different: for a participatory community whose members are on equal footing.

Eske and Alsleben were based at the Hamburg art academy, where ten years later, the *changing grammars* symposium took place. This symposium brought together live coding practitioners and collectives for the first time. From today's perspective, it is difficult to imagine the speculative nature of live coding at that time, which from the outside could be seen as impractical and very much against the dominant interests of seamless, gestural interfaces and software engineering workflows. This was the first meeting where we revealed to each other that live coding was feasible as an alternative practice, and shared perspectives on why it was interesting to pursue. The symposium moved programming from being a production of a program, to programming as an integral part of the program - and part of a public thinking process (Blackwell et al. 2014, 17). Against the background of conversation art, we perceived changing programs at runtime as a conversational experience, which naturally fell into place with a displacement of the programming practice into participatory public spaces. Where art is a communal praxis, there is not much need for a third party that consumes it. In retrospect, it is obvious that the possibilities of an art without audience has been an important ingredient in live coding from the very beginning. From the perspective of a practice that doesn't require artworks to be received by an audience, programming need not be directed at a finished program applied by users.

[1] „Er ist der Ort, in dem Austausch möglich ist. Datenkunst, Netzkunst ist Kunst ohne Publikum. Es wird nichts geboten: Kein Sender richtet sich an Empfänger. Es geht vielmehr um Formen/Konventionen, einen gemeinsamen Code zu erarbeiten." (translation R.W.) Eske and Alsleben (2010)

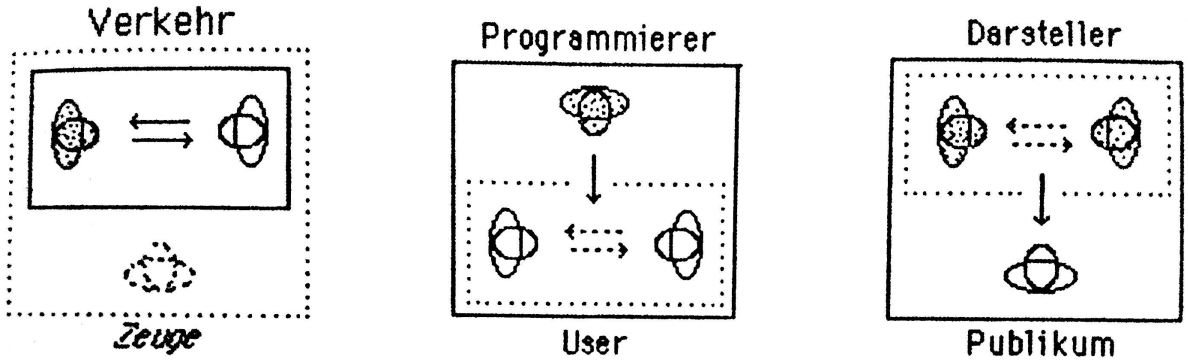

Figure 1: *The so called* "DreiDrei-Schema" *(1992) lays out three forms of communicative polarity in the art. In the right image, the performers' (Darsteller) communication is witnessed by the audience (Publikum). In the middle, the programmer (Programmierer) controls the users' communication (User). The leftmost is the schema of art without audience, or art as interchange: the artist is like the other, and the audience becomes a mere coincidental witness (Zeuge). Image by Kurd Alsleben (Alsleben and Lehnhardt 1994)*

1.1 Collective Exploration

To resolve the division of the spectator is not a single act but a never ending learning process, and various attempts have been made by different live coding groups. PowerBooks_UnPlugged, for example, was a live coding network band formed around 2007 which played its last concert in 2011. Two of the present authors (Wieser and Rohrhuber) were part of it. We played concerts only using laptop speakers and used the supercollider-based Republic system, so we could iterate sound through each others' machines. It was a displacement of agency: what everybody could hear on one of the laptops was not necessarily what the person behind it was playing, and the code that person would run would be spread out to different laptops through the space (Rohrhuber et al. 2007). Not only was the sound distributed but also the code, one could catch, rearrange and reuse it as it passed around the wireless network.

The group would sit distributed within the audience space, and if there was one, the stage remained empty. With this setup, the concerts were relatively quiet but spatialised and the people around could look over the shoulders of the performers. We remember sitting there at one occasion, getting ready, while listening to people around us. They were wondering and mocking the concept, because they didn't understand that we weren't checking emails, but as the sounds started to fill the space, they realised that they were closer to the performance than expected. In general, this slightly awkward exposure was met with friendly irritation. It has something to do with the performance of distributed authorship, which is the basis of programming in any case (one always continues the work of others). Sitting within the audience was just a little shift of the rules of the game. It didn't abolish the notion of a third party, but rendered the division of roles porous.

This theme of a deconstruction of the conventional performer-audience relationship continued to be challenged through the development of live coding culture. While the concept of the 'laptop orchestra' (or LorK) translated traditional Western musical hierarchies from instrumental to laptop performance, live coding has instead generally explored more participatory 'laptop ensembles' such as PowerBooks_UnPlugged and BiLE. Even where the word 'orchestra' appears in the name, such as with the Cybernetic Orchestra, a wholly participatory and communal approach to music making has been taken (Ogborn 2014).

Famously, this participatory and communal spirit is also found in the Mexican live coding scene. After its founding through events at Taller de Audio del Centro Multimedia CENART from 2010 onwards, it quickly became the most active geographically-centred live coding community of its time. The scene developed special practices: live coding 'from scratch' for exactly 9 minutes, people played in audio/visual pairs in short performances, which ended in applause by the others (Villaseñor-Ramírez and Paz 2020).

These early live coding initiatives replace the focus on the audience with a focus on collective exploration, often referred to as *public thought*. This shift explains why also the programming activity had to be completely included within the program runtime: there is no user who would be waiting for the programmer to finish programming and ship the software.

2 Disappearance of Programming

The end of computer programming has been prophesied throughout its short history. Most recently it is seen as an activity that can be turbo-charged with a large language model (LLM) 'co-pilot' that can write chunks of code for the programmer, or even one that could be replaced entirely with a neuralink device, where thoughts can be directly beamed into the computer. Whether this is or is not possible or imminent is not necessarily interesting for live coders, who will want to involve themselves in music and other artworks, whether or not automated processes are able to mimic them. However, there is an interesting assumption at play here - that the purpose of programming is to optimise the transmission of thought into the computer. Clearly, live coding is a counterexample which undermines this assumption.

The optimisation of the one-way transmission of thought into applications can be seen as a motivation in the development of live debuggers. On the surface, debugging appears to share common aims with live coding, but there are hidden but profound philosophical differences between them. Live debugging aims to make programming efficient, so that it almost disappears in the fluid transmission of a designed idea from the programmer to a working application. Live coding on the other hand aims to share and distribute the dynamics of the thought process, with all of its uncertainty, anticipation, experimentation and surprise, and where the experience of coding happens in parallel with music, dance, moving image, or discovery and insight.

In a live debugging environment, the program is placed on a workbench and operated on while it 'lives', until it is deemed ready to be distributed to end-users. Conversely, in a live coding environment, the programming doesn't operate on the program, rather the programming is part of the program. This stands for very different motivations for dynamism. For many 'future of coding' and 'programming language experience design' researchers, 'dynamic media' is a meta-medium encompassing all others, where 'computational thinking' melts into the direct transmission of thought into material. However, art forms such as sound or dance are not absorbed into code as an all-encompassing meta-medium, rather live code, as meta-material, allows a fresh yet respectful approach to these established practices.

By engaging with material via code, we are able to combine and transform structural elements with outcomes that would be well beyond our imaginations, if those outcomes weren't immediately presented to our senses, continually interlacing abstraction with perception. Each speculative edit made by a live coder asks the question 'what if?', each question informed by what went before, and each answer informing what edit comes next. By comparison, the idea of beaming digital artefacts from our imaginations into the outside world, fully formed, seems beside the point.

So while 'liveness' has become an increasingly cherished topic in the software engineering community, this is generally only seen as a means for increase in programming efficiency, to the point that the thinking process of programming becomes obsolete. Because public thought, mostly realised in free exploration in small groups, doesn't scale well, the point of live coding has mostly been lost in the mainstream notion of such 'liveness'. However for us, the 'meaning of live' stands not for this disappearance of thought in the presence of a seamless tool, but quite the opposite – for a whole thinking practice.

3 Public Thought

Live coding challenges dominant modes both in music and software engineering culture. Through mass media headlines, live coding has either been contrasted from or associated with DJ culture. Headlines such as "DJs of the Future Don't Spin Records—They Write Code" (Wired Magazine, 2019) suggest that DJs will either be replaced by coders, or have to learn to code themselves – supposedly because typing code is a superior way to selecting and mixing music to using turntables or DJ controllers. These attention-grabbing headlines have bemused live coders, who do not wish to replace DJs, and in any case are not live coding 'in the future' but now already, as part of a decades-old practice.

However on a different level, the comparison of live coding to DJ culture is interesting to pursue. At a point where the different geographical and cultural strands of DJing are now better known and historicised, the parallels with the different ways that live coding is practiced and understood are coming into view. One parallel is the notion of *control*: is a DJ seeking to control the experience and behaviour of a crowd? Early DJ Jimmy Saville, who (falsely) claimed to be the first to DJ with twin turntables and microphone in the early 40s, is perhaps the most extreme case of a DJ who explicitly sought to control crowds, a desire which extended to profound, abusive control of the vulnerable people around him. However academics, DJs and party organisers Jeremy Gilbert and Tim Lawrence contrast this notion of the controlling DJ with the work of figures such as David Mancuso, whose legendary 'Loft' house parties created a radical Black and Queer-led space for counterculture, away from heavily regulated commercial venues, looking to build a cultural revolution one lounge at a time[2]. Although we would not claim that live coding is comparable as a countercultural force, from our perspective, we can draw some comparisons to the alternative and partly semiofficial

[2]For the Love Is The Message podcast, see https://www.loveisthemessagepod.co.uk/, with David Mancuso a recurring reference throughout the first series.

venues in which live coding practice developed, such as London's Foundry pub, Public Life bar, and Open Lab events, Mexico City's Centro Multimedia, the Access Space media lab, the and the 'Le Placard' headphone festival which also emerged from events in apartments in Paris and Tokyo. What these venues and events have in common is an explicit openness, that created space for live coders to find each other and develop a collective community of practice.

Drawing on these cultural reference points brings attention to live coding's capacity for rethinking rules and conditions, and sharing the collective wealth that commercial software works so hard against, despite all of its talk of industry and innovation. Whether a live coder brings carefully prepared code to work with, or works from scratch, unless they confront the capacity for change are they really live coding? More than ever, we need to create spaces and practices with a sense that alternatives are possible, in order to imagine a different future. Rejecting commercial licenses, and instead sharing code in free communities away from the extractive economy, feels like a pre-requisite for this (although of course, not enough).

In rejecting a certain kind of virtuosity then, we reject the idea of channeling the virtues of genius for an audience. In the staging of the DJ, this might be expressed by notorious 'christ' poses and awkward mixing console gestures. Following Ursula Franklin (1999)'s characterisations in her lectures 'The Real World of Technology', technologies of control can be abandoned in favour of technologies of craftwork: a live coder may follow code into unexpected places, writing code to create space for an audience to explore together, rather than to control them.

Lendl Barcelos' project CUE opens up the experience of DJing in installation form, where visitors are able to hear in one room the headphone mix of a DJ cueing and mixing two records and the main mix in a second room, so the 'seams of the seemingly seamless become audible', against the background of what seems to be without gap. This offering of the private world of craft and reasoning reminds us what is offered in a live coding performance, where despite some experiments (e.g. Kirkbride 2020), cueing code on headphones has not developed as a practice. The searching for and preparing of material is shared as part of the end result, with little possibility for inauthenticity and every possibility for surprise, whether through errors, or emergence of form.

4 Programs Without Users

Just like the idea of an audience is oriented towards the existence of the artist, the idea of a user of a program is centred on that of a developer. Programming work is hidden in order to isolate the situations of use from the situations of development. Thus, software becomes a commodity. Furthermore, while programs can be given away for free, this does not give access to any decisions that were made through them.

It has always been a challenge for those who do free and open source software to give access to the programming activity itself. One can say when everybody is computer literate, the user disappears. Live coding performs this ideal, first of all in making programming a participative and public affair. There is another aspect, however, which is harder to make sense of, especially for those who have learned to be a professional software engineer. So, even though interactive and live programming techniques are increasingly integrated into the tool bench of professional programming, they still only serve the purpose of more efficiently finishing the development and reaching the program as a final product.

This second aspect is that in live coding, the very idea of a program has changed. A program is not programmed to be used later, but is used by programming it. This is why it makes sense to say that, just like an artwork without audience, it is a program without user. And indeed, the notion of the user as opposed to the developer is misleading insofar as it implies a merely passive receiver, a consumer. By contrast, using, rather than being used, is an active participation; and in this sense, just as when we are involved in a conversation with the computer, we are all audience, when programming is the user interface, we are all users. [3]

In such art
there is no artist author
there is no artwork
there is no distribution
there is no public

(Kurd Alsleben, „Urheben", in: Kongressband, 1. Kieler Netztage '93,Verlag Claus Schönleber, Kiel,1993, p. 197–198, our translation.)

[3] "When it comes to computing (not least), the term user seems at first to be rather limited and functionary, even when describing well-meaning notions such as user-centered or user-friendly, which are both part of a more general shift toward seemingly inclusive and participatory forms. However, as much as it is clear that the user is open to subtle forms of exploitation, we argue that the user might also be a force for reinvention."

References

Alsleben, Kurd, and Antje Eske. n.d. “Urheben, Vortragsgespräch auf dem 10. Chaos Computer Congress in Hamburg am 29.12.1993.” In.

Alsleben, Kurd, and Matthias Lehnhardt. 1994. *Gesänge über dem Lerchenfeld: Beiträge zur Datenkunst ; Propemptikon & Apopemptikon*. 2. Aufl. Material. Hamburg: Material Verl. der Hochsch. für Bildende Künste Hamburg.

Blackwell, Alan, Alex McLean, James Noble, and Julian Rohrhuber. 2014. “Collaboration and Learning Through Live Coding (Dagstuhl Seminar 13382).” Edited by Alan Blackwell, Alex McLean, James Noble, and Julian Rohrhuber. *Dagstuhl Reports* 3 (9): 130–68. https://doi.org/http://dx.doi.org/10.4230/DagRep.3.9.130.

Eske, Antje, and Kurd Alsleben, eds. 2003. *NetzKunstWörterBuch*. Norderstedt: Books On Demand.

———. 2010. “Hypertext. Kunst Ohne Publikum.” http://konversationskunst.org/index.php/texte/18-eske-a-alsleben-hypertext-kunst-ohne-publikum.

Franklin, Ursula. 1999. *The Real World of Technology*. 2nd edition. House of Anansi Press.

Kirkbride, Ryan Philip. 2020. “Collaborative Interfaces for Ensemble Live Coding Performance.” PhD thesis, University of Leeds. https://doi.org/ch6_1a-Rehearsal-26_04_17.mpg.

Ogborn, David. 2014. “Live Coding in a Scalable, Participatory Laptop Orchestra.” *Computer Music Journal* 38 (1): 17–30. https://doi.org/10.1162/comj_a_00217.

Rohrhuber, Julian, Alberto de Campo, Renate Wieser, Jan-Kees van Kampen, Echo Ho, and Hannes Hölzl. 2007. “Purloined Letters and Distributed Persons.” In *Music in the Global Village Conference*. Budapest.

Villaseñor-Ramírez, Hernani, and Iván Paz. 2020. “Live Coding From Scratch:The Cases of Practice in Mexico City and Barcelona.” In *Proceedings of the International Conference on Live Coding*, 59–68. Limerick, Ireland: University of Limerick. https://doi.org/10.5281/zenodo.3939206.