Live Coding Towards Computational Creativity

Alex McLean and Geraint Wiggins

Goldsmiths, University of London, United Kingdom
ma503am@gold.ac.uk,
WWW home page: http://doc.gold.ac.uk/∼ma503am/

Abstract. Live coding is a way of improvising music or video animation through live edits of source code, using dynamic language interpreters. It requires artists to represent their work within a formal computer language, using a higher level of abstraction than is the norm. Although the only creative agents in live coding performance are human, this abstraction makes the practice of interest to the field of computational creativity. In this paper live coders are surveyed for their thoughts on live coding and creativity, related to the aims of building creative agents.

1 Introduction

Live coding is the writing of rules in a Turing complete language while they are followed, in order to improvise time based art such as music, video animation or dance. This is a relatively new approach, receiving a surge of interest since 2004, through both practice and research [1–6].

Live coding is most visible in performance, however the 'live' in live coding refers not to a live audience but to live updates of running code. Conventionally humans write code followed by software, although some experimental dance improvisations have used both human rule makers and rule followers. Whether human live coders can be replaced by software creative agents is a question for the field of computational creativity, which we hope to have at least clarified by the end of this paper.

In contrast to live coding, generative art is output by programs unmodified during execution, which often have no user interface at all. The lack of control over such programs has led to a great deal of confusion around the question of authorship. When watching a piece of software generate art without guidance, onlookers ask "is the software being creative?" There is no such confusion with live coding, there is a human clearly visible, making all the creative decisions and using source code as an artistic medium. In fact there is no difference of authorship between live coded and generative art. A programmer making generative art goes through creative iterations to, only after each edit they have to restart the process before reflecting on the result. This stuttering of the creative process alone is not enough to alter authorship status.

If the computer's role in a live coding performance is uncreative, then what is this paper doing submitted to a computational creativity conference? Well, as a new way of producing art using formal systems, it is hoped that live coding

can give a unique clarifying perspective on issues of computational creativity, and perhaps even become a stepping stone towards a creative software agent.

2 Live coders on computational creativity

A survey was carried out with the broad aim of gathering ideas for study of computational creativity. The members of TOPLAP [3], an active live coding community, were asked to fill out an on-line survey, and 32 responded. To avoid prejudice, the word 'creativity' was not used in the invitation or survey text, and pertinent questions were mixed with more general questions about live coding.

2.1 Results

The subjects. Users of the six pre-eminent live coding environments were represented, between five and fourteen for each system (many had used more than one). Background questions indicated a group with a generally rich musical background. There were a diverse range of approaches to the question of how to define live coding in one sentence, and the reader is referred to the on-line appendix to read the responses (http://doc.gold.ac.uk/∼ma503am/writing/icccx/). While the responses show some diversity of approach, because the subjects had all used at least one of the main languages it seems safe to assume that they are working to largely the same technical definition.

Creating language. Computer users often discuss and understand computer programs as tools, helping them do what they need efficiently. For a programmer it would instead seem that a computer language is an immersive *environment* to create work in. It is interesting then to consider to what extent live coders adapt their computer languages, personalising their environments, perhaps to aid creativity. Over two thirds (69.0%) collected functions into a library or made an extensive suite of libraries. This is analogous to adding words to a language, and shows the extent of language customisation. A smaller proportion (20.7%) had gone further to implement their own language interpreter and fewer still (17.2%) had designed their own language. That these artists are so engaged with making fundamental changes to the language in which they express their work is impressive.

Code and style. From the perspective of computational creativity, it is interesting to focus on the code that live coders produce. Their code is not their work, but a high level description of how to make their work. A creative computational agent would certainly be concerned with this level of abstraction. An attempt at quantifying how live coders feel about their code was made by asking "When you have finished live coding something you particularly like, how do you feel towards the code you have made (as opposed to the end result)?" Over half (56.7%) indicated that the code resulting from a successful live coding session

was a description of some aspect of their style. This suggests that many feel they are not encoding a particular piece, but how to make pieces in their own particular manner. Around the same number (50.0%) agreed that the code describes something they would probably do again, which is perhaps a rephrasing of the same question. A large number, (83%) answered yes to either or both questions. There are many ways in which these questions can be interpreted, but the suggestion remains that many subjects feel they have a stylistic approach to live coding that persists across live coding sessions, and that this style is somehow represented in the code they make.

Live coding as a novel approach. The subjects were asked the open question "What is the difference between live coding a piece of music and composing it in the sequencer (live coding an animation and drawing one)? In other words, how does live coding affect the way you produce your work, and how does it affect the end result?" Some interesting points relevant to computational creativity are selectively quoted for comment here, the reader is again directed to the on-line appendix to read the full responses.

> "I have all but [abandoned] live coding as a regular performance practice, but I use the skills and confidence acquired to modify my software live if I get a new idea while on stage."

The admission that getting new ideas on stage is infrequent, makes an important and humble point. In terms of the Creative Systems Framework (CSF) [8, 7] we can say that live coding is useful in performance if you need to transform your conceptual space (the kind of work you want to find or make), or your traversal strategy (the way you try to search for or make it). However, as with this subject, transformational creativity is not always desirable in front of a paying risk-averse audience.

> "When I work on writing a piece ... I can perfect each sound to be precisely as I intend it to be, whereas [when] live coding I have to be more generalised as to my intentions."

Making the point that live coders work at least one level of abstraction away from enacting individual sounds.

> "Perhaps most importantly the higher pace of livecoding leads to more impulsive choices which keeps things more interesting to create. Not sure how often that also creates a more interesting end result but at least sometimes it does."

Live coding allows a change in code to be heard or seen immediately in the output, with no forced break between action and reception. This would be a surprise to those whose experience of software development is slow and arduous.

> "Live coding has far less perfection and the product is more immediate. It allows for improvisation and spontaneity and discourages over-thinking."

This may also come as a surprise; live coding has a reputation for being cerebral and over technical, but in reality, at least when compared to other software based approaches, the immediacy of results fosters spontaneous thought.

> "Live Coding is riskier, and one has to live with [unfit decisions]. You can't just go one step back unless you do it with a nice pirouette. Therefore the end result is not as clean as an "offline-composition", but it can lead you to places you [usually] never would have ended."

This comment is particularly incisive; the peculiar relationship that live coders have with time does indeed give a certain element of risk. Thinking again within the CSF [7], such riskier ways of making music are more likely to produce aberrant output, providing the opportunity to adjust your style through transformational creativity.

> "... while Live Coding is a performance practice, it also offers the tantalising prospect of manipulating musical structure at a similar abstract level as 'deferred time' composition. To do this effectively in performance is I think an entirely different skill to the standard 'one-acoustic-event-per-action' physical instrumental performance, but also quite different to compositional methods which typically allow for rework."

This really gets to the nub of what live coding brings to the improvising artist – an altered perspective of time, where a single edit can affect all the events which follow it.

Live coding towards computational creativity. The subjects were given a series of statements and asked to guess when each would become true. Regrettably there was a configuration error early on in the surveyed period, requiring that the answers of two subjects were discarded.

Optimism for the statement *"Live coding environments will include features designed to give artistic inspiration to live coders"* was very high, with just over half (51.9%) claiming that was already true, and two fifths (40.7%) agreeing it would become true within five years. This indicates strong support for a weak form of computational creativity as a creative aide for live coders. Somewhat surprisingly, optimism for the stronger form of creativity in *"Live code will be able to modify itself in an artistically valued manner"* was also high, with two fifths (40.7%) claiming that was already possible. If that is the case, it would be appreciated if the live code in question could make itself known, although it seems more likely that ambiguity in the question is at fault. A little more pessimism is seen in response to *"A computer agent will be developed that produces a live coding performance indistinguishable from that of a human live coder"*, with a third (34.6%) agreeing this will never happen. This question is posed in reference to the imitation game detailed by Alan Turing [9]. However as one subject commented, "the test indistinguishable from a human is very loose and there can be some very bad human live coding music." That would perhaps explain why half (50.0%) thought the statement was either already true or would become so within five years.

3 Conclusion

What if a musicology of live coding were to develop, where researchers deconstruct the code behind live coding improvisations as part of their work? Correlations between expressions in formal languages and musical form in sound could be identified, and the development of new ways of expressing new musical forms could be tracked. If successful, the result need not be a new kind of music, but *could* be a music understood in a novel way. It is this new computational approach to understanding music that could prove invaluable in the search for a musically creative software agent.

In looking at creativity through the eyes of live coders, we can see some promise for computational creativity even at this early stage of development of both fields. Live coders feel their musical style is encoded in the code they write, and that their language interfaces provide them with inspiration. They are actively developing computer languages to better express the music they want to make, creating computer language environments that foster creativity. From here it is easy to imagine that live coding environments could become more involved in the creation of higher order conceptual representations of time-based art. Perhaps this will provide the language, environment and application in which a computational creative agent will thrive.

References

1. Blackwell, A., Collins, N.: The programming language as a musical instrument. In: Proceedings of PPIG05. University of Sussex (2005)
2. Wang, G., Cook, P.R.: On-the-fly programming: using code as an expressive musical instrument. In: NIME '04: Proceedings of the 2004 conference on New interfaces for musical expression, Singapore, Singapore, National University of Singapore (2004) 138–143
3. Ward, A., Rohrhuber, J., Olofsson, F., McLean, A., Griffiths, D., Collins, N., Alexander, A.: Live algorithm programming and a temporary organisation for its promotion. In Goriunova, O., Shulgin, A., eds.: read_me — Software Art and Cultures. (2004)
4. Collins, N.: Live coding practice. In: NIME. (2007)
5. Collins, N., McLean, A., Rohrhuber, J., Ward, A.: Live coding in laptop performance. Organised Sound **8**(03) (2003) 321–330
6. Sorensen, A., Brown, A.R.: aa-cell in practice: An approach to musical live coding. In: Proceedings of the International Computer Music Conference. (2007)
7. Boden, M.: The Creative Mind. Abacus (1990)
8. Wiggins, G.A.: A preliminary framework for description, analysis and comparison of creative systems. Journal of Knowledge Based Systems (2006)
9. Turing, A.M.: Computing machinery and intelligence. Mind **LIX** (1950) 433–460