ABSTRACT

Flow (2015) is a system for solo live coding performance that playfully explores and subverts aspects of cognitive flow and virtuosity in Live Coding. The piece was developed as part of a collaborative residency at Digital Media Labs, UK in September 2015. Flow builds on my own practice in live coding which focusses on live coding synthesis in SuperCollider and working in ways which are likely to induce performer error and unpredictable results. In the performance an EEG monitor is used to access the performer's engagement levels, allowing the development of a performance narrative relating to both the physical and cognitive aspect of live coding performance.

1. VIRTUOSITY IN LIVE CODING

In instrumental music practices, virtuosity is commonly related to physical accuracy, and precision. In Live Coding, however, the performer's ability to physically interface with the music production is reduced to the act of typing. The TOPLAP Draft manifesto (Toplap wiki) proposes the 'glorification of the typing interface' may be part of the live coding performance, and Nilson (2007) suggested typing exercises to develop typing skill and accuracy in the performer. It follows then, that building a performance interface which requires virtuosic performance would include the possibility of drastically increasing the need to type quickly and accurately.

A second aspect of virtuosity in live coding performance is proposed by Sawyer (2015), that is, that live coding performance relies more heavily on cognitive virtuosity than physical virtuosity. Rohrhuber and de Campos (2009) conversational programming proposition suggests that live coding practice involves a feedback process of translating aesthetic thought into code, perceiving the result, and decoding the difference between imagined and actual sound. A second proposition then is that a performance system requiring cognitive virtuosity may intensify the requirements for the performer to comprehend and make logical changes to code, and their necessity to maintain concentration levels.

Flow addresses virtuosity in live coding by reducing time to comprehend and edit algorithmically generated code, and by using the performer's concentration levels as part of a feedback loop to moderate the time given to modify code. The sections which follow address aspects of Brain Computer interfaces in music, human-algorithm interaction narratives, and system design and evaluation.

2. BRAIN COMPUTER INTERFACES IN MUSIC

In order to explore cognitive virtuosity in live coding performance I was interested in developing a direct link between the cognitive actions of the performer and the difficulty of performing with the system.

Brain Interfaces have been used in music performance and composition since Lucier's 1965 Music for Solo Performer, where the performer is required to produce alpha waves in order to vibrate percussion instruments (Lucier 1998). Many other examples exist of e.g. sonifying brain activity (Lutters and Koehler 2016), or using particular emotional states to trigger sounds or navigate sound environments (Pearlman 2015).

Most systems which interface with the brain for musical or aesthetic purposes deal, directly or implicitly, with biofeedback. That is, as it is not possible for the performer to control all cognitive
reaction to visual and auditory stimuli, the output of the BCMI impacts the brain activity, building a feedback loop between brain state and musical/aesthetic result (Lutters and Koehler). As we have seen, feedback loops are integral to human-algorithm interaction in live coding performance (Rohrhuber and de Campo), therefore I was interested in replicating this more complex interaction with the use of the EEG monitor in Flow. Another important aspect of using the EEG monitor in Flow was to develop a narrative of cognitive interaction, which is otherwise inaccessible for audience observation.

The next section discusses narratives of interaction in Live coding Performance.

3. REVEALING NARRATIVES OF HUMAN-ALGORITHM INTERACTION

Live Coding performance practice often includes projecting the performance interface, in order to reveal the process of the music production to the audience (toplap wiki). In much the same way that the physical actions of an acoustic musician and their relationship to the resulting sound is observable to the audience, projecting the laptop screen gives the audience insight into how the performer is interacting with the code.

Regardless of code literacy levels in audience members, displaying this interaction reveals something of the dynamic interplay between human action and algorithmic output, and the inevitable, and potentially dramatic, inclusion of human or technical error as part of the performance narrative.

In 2014 I became interested in finding more transparent ways – than direct screen projection – to graphically represent the interaction between human, interface, system, and algorithm, in order to present the human narrative as an explicit concern of the live coding performance. Alongside several aspects of the code and system state, I represented the human interaction with the performance system through visualising the typing density. A video including this initial attempt at graphic representation can be seen here: https://vimeo.com/114298725.

Since 2014, I have experimented with many iterations of these graphics, refining the aesthetic aspects and aiming to reduce the opacity of the visual representation. The representation of typing action, however has remained a central concern of all iterations of the visualisation, as this gives the audience an insight into the ebb and flow of the performer's physical interaction with the keyboard interface, perhaps also implicitly suggesting a timeline of cognitive interaction.

Another aspect of live coding revealed through typing representation is the asynchronous and uncorrelated relation between performer action and sonic result in live coding. As with other algorithmic and electronic music systems, physical exertion does not directly equate to sonic complexity. For example, furious typing from the performer may result, on future code execution, equally in smooth drones or complex rhythmic structures.

This disparity between physical intervention and sonic result proves fertile ground for exploration in developing performance systems for live coding. In 2016 Joanne Armitage built vibrating devices which vibrate in synchronisation with the typing, and are held by audience members while the performer live codes (Redhead 2016), and Andrew Brown (Brown 2013), Chris Kiefer (Kiefer 2015), and others have projected webcam images of the performer's interaction with the interface.

In Flow, a further narrative was explored, that is, the narrative of the performer's cognitive interaction with the performance interface. In Flow, the performer's EEG readings are visualised alongside the code, typing and keyboard, giving a multi-narrative view of the performer-algorithm interaction. In the next section we discuss the development, testing and future work of the performance system.

4. DEVELOPING AND TESTING THE SYSTEM

5. Initial Experiments

The first iteration of Flow was developed as part of a residency I took part in, in September 2015. Digital Media Labs is a collaborative digital arts residency where ten media artists from different disciplines spend a week in a co-working space expanding their practice and developing new works. In the initial presentations I saw synergies between my work on narrative in live coding performance
and work by Gemma May Latham on cognitive flow in textiles work. We conducted experiments using an industrial EEG monitor on my levels of attention and meditation and valence in 2 situations: (1) solo live coding and (2) live coding in collaboration with another live coder (Alex McLean). We recorded the brain signals, my physical interaction with the keyboard, and the performance interface, in order to deduce correlation between brain activity and performer action. Experiment 1 can be seen here: https://vimeo.com/161947655. Unfortunately a technical error resulted in experiment 2 not being recorded properly, however comparisons carried out during the experiments revealed that cognitive flow (i.e. consistently high levels of attention with moderate levels of valence (Ulle 2010)) was more likely to take place in a solo performance than when collaborating. This research served as the foundation for building a system which uses measurements relating to cognitive flow as an input into a live coding system.

6. The System

The piece consists of a set of 10-15 algorithmically generated snippets of supercollider code which simultaneously begin playing at the start of the performance. The snippets are presented to the performer one at a time in a randomised order. The performer has a short time frame of 15-40 seconds to comprehend and edit each snippet of code. This process is modulated by a live EEG reading of the performer's concentration level (an important measure of cognitive flow (Nakamura and Csikszentmihalyi 2002)) which is used to make the system more difficult to play with by extending the time frame of the code editing time when the performer's concentration is low and shortening the time frame when the concentration levels are high. i.e. time to edit a piece of code is inversely related the performer's engagement levels.

The performance is accompanied by a visualisation which shows the performer's physical and cognitive interaction with the code and the performance. The physical interaction is presented through a webcam image of the performer's typing which appears more clearly when typing speed is higher. This is overlaid with the EEG readings for attention and meditation represented with grayscale colour blocks (white = high, black = low) and a 0-100 number reading. The current code snippet is shown, and white lines are used to represent typing density for each code snippet.

The sound is projected in multichannel with the sounds relating to each code snippet equally spaced in the sound field to aid the performer and audience in relating edits to code to changes in sound.

The system creates a narrative that relates to the cognitive flow of the performer as well as their interaction with the system and code. The performer's impact on the musical output depends on their ability to quickly interpret and edit code under time and concentration pressures.

7. Testing and Evaluation Constraints

Since the initial development phase I have performed with the system on two occasions. These performances revealed that test conditions in the studio are not sufficient to determine the correct difficulty level for the system. In the studio I consistently record attention and meditation levels of between 30-70 with an average at around 50. However in performance situations the fluctuations between readings are much higher, recording between 0-100 in both measures and with attention being generally higher: ca. 60-90 and meditation being often low: under 40. It would be reasonable to assume the stress levels induced in performance situations impact strongly on levels of attention and meditation, and that maintaining consistent values is much more difficult.

Given that my attention levels in performance were much higher than expected, this also made the system much more difficult to perform with, further compounding the issues relating to performance stress, creating a frustration feedback loop. Although this is intended to be part of the way the system functions, I was unable to reach a level of control of my stress levels sufficient to interact with the system in a musically satisfying way.

As it became clear that I needed a way to adjust the system in real time to deal with relative, rather than absolute, stress levels, in the second of the two performances I adjusted the trigger which changes the code snippet, such that the system keeps a running average of the two measures and triggers a jump to the next code snippet when the values are above the running average. This seemed to ease the difficulty level somewhat, and as a performer I felt more able to interact in an
adequate way with the code, however the performance did still suffer from long periods of fast code changes reducing my ability to edit the sound scape in a musically pleasing way.

8. Future Development

The limitations of higher stress levels in performance make system modifications difficult to prototype and test in the studio, therefore the piece is likely to develop slowly, as performance opportunities provide further test conditions. In the next performance I plan to raise the trigger threshold to 10% above running average in an attempt to ease the difficulty level and regain a sufficient level of musical control of the performance system.

9. CONCLUSION

Flow proposes that increasing time intensity for live coding activity may increase virtuosic requirements of the live coding performer. Giving the performer unknown code for short durations increases the need for fast cognitive and physical abilities. Coupling this with a process which reduces time according to levels of concentration creates a bio-feedback loop which creates an ultimately subversive system which works against performers'attempts to engage high levels of cognitive capacity in attempting to understand and edit code snippets. Presenting the data relating to this process as part of a visualisation gives the audience access to the performer's cognitive and physical interactions with the algorithmic processes.

Acknowledgments

With thanks to Digital Media Labs, the Frozen Music Collective, Gemma May Latham and Alex McLean for their involvement in the research and development of Flow.

REFERENCES


