Routledge
Taylor & Francis Group

Contemporary Theatre Review

ISSN: 1048-6801 (Print) 1477-2264 (Online) Journal homepage: http://www.tandfonline.com/loi/gctr20

Sound Choreography <> Body Code: Software Deployment and Notational Engagement without Trace

Kate Sicchio & Alex McLean

To cite this article: Kate Sicchio & Alex McLean (2017) Sound Choreography <> Body Code: Software Deployment and Notational Engagement without Trace, Contemporary Theatre Review, 27:3, 405-410, DOI: 10.1080/10486801.2017.1343244

To link to this article: https://doi.org/10.1080/10486801.2017.1343244

Contemporary Theatre Review, 2017
Vol. 27, No. 3, 405–410, https://doi.org/10.1080/10486801.2017.1343244

Sound Choreography <> Body Code: Software Deployment and Notational Engagement without Trace

Kate Sicchio and Alex McLean

Computer programs describe interwoven systems of data flows and organising processes. These flows and structures are generally hidden, but computer programmers conceptualise them as mental imagery through their work and it is possible to visualise them through technologies including visual programming languages.[1] This leads us to consider the writing of both software source code and choreographic scores as organisational practice, and look for ways of connecting them by linking together their different notational systems through technology.

A number of choreographers have developed, and to some extent formalised, their own notational systems, discussed by Rudolph Laban, Scott deLahunta, and Kate Sicchio, and systems that use scores to produce live works.[2] These notations have practical aims of supporting the composition of new choreography, or documentation and analysis of a performance, but have in the process brought code and choreography closer together, making their correspondences and differences visible. More recent projects have introduced digital technology to support the creation of movement and dance works during composition and rehearsal sessions, including the Software for Dancers project[3] and the collaborations between Marc Downing, Nick Rothwell, and Wayne McGregor in the *Becoming* artificial intelligence agent.[4] William Forsythe's company's collaboration with Ohio State University *Synchronous Objects* and its follow-on project *Motion Bank*, sit somewhere between an archive of scores and annotated videos of choreography, and the use of movement to create digital artworks.[5]

Live coding practice is distinctive for manipulating code 'on the fly', where code is created and

For material which compliments this article please also see Kate Sicchio and Alex McLean's contribution to the the journal's Interventions website: '*Sound Choreographer <> Body Code*' <https://www.contemporarytheatrereview.org/2017/kate-sicchio-and-alex-mclean>.

1. Marian Petre and Alan Blackwell, 'Mental Imagery in Program Design and Visual Programming', *International Journal of Human–Computer Studies*, 51 (1999), 7–30.
2. See Rudolph Laban, *Laban's Principles of Dance and Movement Notation*, 2nd edn (London: MacDonald and Evans, 1975); Scott deLahunta, 'The Choreographic Resource: Technologies for Understanding Dance', *Contact Quarterly*, Chapbook 1, 35.2 (Summer 2010), 18–27; Scott deLahunta, *Software for Dancers: The Users Guide* (2002) <http://www.sdela.dds.nl/sfd/scott.html> [accessed 6 June 2016]; and Kate Sicchio, 'Hacking Choreography: Dance and Live Coding', *Computer Music Journal*, 38.1 (2014), 31–39.
3. *Software for Dancers* (2002) was a project facilitated by Scott deLahunta, bringing together software engineers and choreographers to develop tools for the choreographic process.
4. *Becoming* is an artificial intelligence agent that animates 3D drawings used within the rehearsal process for dancers to respond to and improvise movement with. Marc Downie, Nick Rothwell, and Wayne McGregor, 'Becoming', Wellcome Trust Exhibition, London, 3 October 2013.
5. See William Forsythe, 'Choreographic Objects', *Synchronous Objects* (2009) <http://synchronousobjects.osu.edu/media/inside.php?p=essay> [accessed 6 June 2016]; and 'Motion Bank' (2013) <http://motionbank.org/en> [accessed 6 June 2016].

developed during a performance, and often made visible to the audience via video projection of the coder's screen.[6] Crucially, the process continually interprets the code while it is edited, so that the code becomes a live user interface. Typical live coding performances involve a human as the programmer, and a computer as interpreter. However, live coding is a set of techniques that may be applied to any time-based work, including choreography. In live choreography the tables are turned somewhat, in that the code is not generally interpreted by an electronic computer, but by a human dancer. To support discussion in the present article, we coin the term code-score, which refers to any symbolic notation of rules constraining or guiding a live, improvised performance, whether it is followed by an electronic computer or a human.

Through the following, we will focus on live code-scores which are themselves created or modified during performance. Work bringing live coding and choreography together has become an active topic over the past few years, developed through the work of Nick Collins and Teresa Prima, Kate Sicchio, and Shelly Knotts and Konstantinos Vasilakos.[7] These works have explored different approaches to changing code-scores during their performance; for example, Collins and Prima employed a chalkboard to change the performer's instructions and the others include the use of programming languages and projected screens.

Collaborations between live coding and choreography may work to highlight the deployment of notation and scoring, transform choreographic work into new formats such as digital animation, or provide fuel for improvisational experiments within a rehearsal setting. The interpretation of a code-score is open to varying degrees of engagement, and heavily dependent on whether a human or machine takes the role of interpreter. Our own work explores both human and machine interpretation in the same piece, providing ground to explore the different ways that human bodies and computational processes can become intertwined in a notational system. Our approach connects two code-scores, one musical and the other choreographic, influencing each other through sound and body.

6. Nick Collins and others, 'Live Coding in Laptop Performance', *Organised Sound*, 8.3 (2003), 321–30.
7. See Nick Collins, 'Live Coding of Consequence', *Leonardo*, 44.3 (2011), 207–11; Sicchio, 'Hacking Choreography'; and Shelly Knotts and Kostantinos Vasilakos, 'Agonyart', *YouTube*, 16 May 2013 <https://www.youtube.com/watch?v=2Pk1nmIAoQs/> [accessed 6 June 2016].

Sound Choreography <> Body Code

Sound Choreography <> *Body Code* follows two strands of individual research: by Sicchio on the relationship between choreography and technology, and by McLean on designing programming languages for expression in live performance. To achieve confluence between them, we needed to engage aesthetic and technical aspects on both sides, finding balance. The solution we arrived at, and so far deployed through five performances, maintains a clear distinction between choreography/dance and code/music, but connects them via their notations. As a result, the music is not coded for the dancer, and the dancer does not move to the music; but still a feedback loop is created that encompasses body and code, via audio signal processing on one side and computer vision on the other.

The piece begins with both performers simultaneously creating live action, and projections of both code-scores within the performance space. A diagrammatic code-score is displayed for interpretation by the dancer (Sicchio), which at first consists of a small number of instructions (right, left, up, down, loop, if) and numbers (one till three), which are automatically connected to form a network (as the minimum spanning tree) (see Image 1).[8] The dancer chooses a set series of gestures, which she then organises and performs in response to how the instructions and numbers are connected. The instructions continually move in response to sound events, where each instruction responds to a particular part of the audio frequency spectrum, causing the spanning tree to be continually recalculated. As the performance progresses, these movements are amplified, and additional instructions are added so that the diagram becomes increasingly complex and fluid. The number of instructions and amount of movement is graphed in the background of the code-score, as 'intensity' and 'change' respectively, using polar coordinates. Both increase over time on this set incremental pattern, which peaks at a point where the dancer is completely overwhelmed. The diagram then returns back to a simpler form to end the performance. Because the reconfigurations are triggered by the instructions moving in response to sound, the diagram responds to the sound that McLean produces via his own code-score. This score is deployed computationally, with

8. The minimum spanning tree is the most efficient way of connecting a set of points with paths between them.

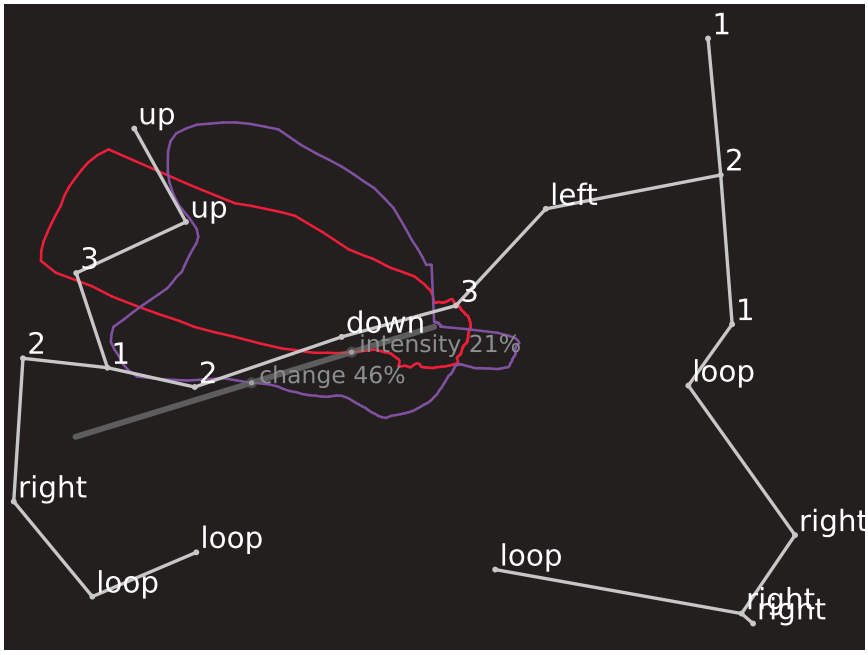

Image 1 The Sound Choreographer, showing instructions right, left, up, down, and numbers, connected in a minimum spanning tree. The line extending from the central point sweeps through a single cycle during the performance, and the shapes outlined in red and purple show 'intensity' and 'change' respectively, graphed over time using polar coordinates. 'Intensity' gives the number of instructions, and 'change' the size of each movement that is made in response to sound onsets.

engagement from both performers that will change that score.

The movement vocabulary within the piece is gestural and limited to a small set of simple movements that are repeated throughout the performance. The dancer decides upon these before each performance, which might include raising one arm, lunges, falling to the floor, and circling both arms. There is a pedestrian quality to the movement and it is very much situated in a post-modern dance cannon, almost as a homage to performance scores from that era of dance. The limited vocabulary of movements helps the dancer focus on following the notation, but also allows the audience to follow the patterns and transformations of the choreography as the piece progresses: this way a trace may emerge.

In *Sound Choreography <> Body Code* (*SC<>BC*), the movement of the dancer is tracked by a Microsoft Kinect sensor via the Isadora software, to detect the location and shape of the dancer's body in space. These data are then sent via the network protocol Open Sound Control (OSC) to Texture (see Image 2), an experimental, visual programming environment, and used to move a function within the code.[9] This motion tracking not only helps to facilitate the feedback loop, but also provides one of the two points of contact between the movement and the sound. So the movement of the dancer changes the code, and the sound produced by the live coding changes the dance score.

The Texture programming environment is visual in a stronger sense than in conventional visual programming languages.[10] In particular, its syntax is based on Euclidean distance, whereas in many other visual systems such as Max/MSP or PureData the programmer makes connections between functions manually. As the function assigned to Sicchio's position on stage moves around McLean's screen it interferes with and disrupts the syntax tree of the running code. Because of the strict nature of Texture's underlying system, the resulting program is always syntactically correct.

9. Alex McLean and Geraint Wiggins, 'Texture: Visual Notation for the Live Coding of Pattern', *Proceedings of the International Computer Music Conference*, 2011, pp. 612–28.
10. A visual programming language is one which uses visual elements in its presentation or syntax, other than plain text.

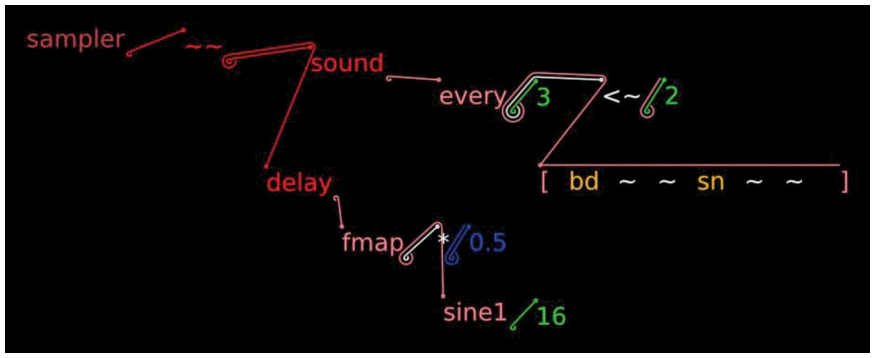

Image 2 The visual programming environment Texture. Words are automatically connected (and re-connected) based on proximity and type compatibility.

In practice, this means that the program is rarely disrupted to the point that it falls silent.

The second point of contact between the choreography and code is via audio processing, as we have already described; the Sound Choreography software performs onset detection on the sound produced from Texture. The words within the choreographic score move in response to the sound, resulting in connections switching between words in the spanning tree. In this sense, the choreographic structure is dancing more directly to the rhythm of the sound than the human dancer.

In *SC<>BC*, the technology has become instrumental in associating the two different practices into a coherent composition. During a previous collaboration by the authors, both performers were focused on their individual scores, creating a co-located rather than combined performance. In these previous improvised works, Sicchio had not noticed developments in the sound as she was performing her code-score, and likewise McLean had been too focused on his code-score to focus on her movements, let alone the influence of his sound upon them. In *SC<>BC*, the technology intervenes, becoming a choreographer organising interaction between the performers, who are otherwise unable to sense and respond to each other. The technology links the feedback loop, bringing bodily movement into the code of the sound on one side, and the sound into the movement of the code on the other. As the piece develops the feedback loop begins to bring elements of uncertainty into the system and allows for the connection to feel as improvised as the creation of the sound and the movement.

Choreography is a time-based organisational process that may be set, improvised, scored, or notated. Within *SC<>BC*, the dance score becomes an active form of programming, and the human dancer the interpreter. The human body is an interpreter performing the machine-generated code. However, embodied human thought processes interpret code and engage and deploy movement in very different ways than a computer. The human has agency, and so is able to make decisions about the code and how to deploy it within the timespan of the performance. For example, when performing a section of the score that links *loop*, *four*, and *right*, the dancer might perform a gesture with the right arm four times while facing away from the audience. However, if they turn and face the audience, the right could be read by the audience as left and the dancer may respond by changing the arm they are moving. The dancer-interpreter does not have direct control over the code, but is able to reflect upon, define, and redefine the semantic meaning of it, deciding upon the meaning of the words through the bodily process of interpreting them.

The nature of time in this process is also key. The choreographic code-score is changing in real-time based upon the sound, meaning that the dancer's decisions are improvised, instantaneous reactions. While the code-score is computer-generated it is executed by a human, and therefore a personal interpretation of the code is performed. The process of reading, interpreting and performing is interrelated in each moment and the work becomes as much a cognitive awareness exercise as it is a physical one. The audience may also engage with these thought processes as the code-score is projected within the piece, allowing them to see the score change, and the dancer respond. This ephemeral approach implies that code may exist as a momentary trace, rather than a fixed notation. Code may fade away just as quickly as a trace made by moving the body.

The choreographic code-score grows in complexity over time, becoming more complicated than a human

dancer can perform. By the end section of *SC<>BC* the dancer will always fail to follow the code-score as it is written. They must make a decision on how to follow and interpret fragments of the score, because there is too much, changing too quickly. This highlights both the different quality and scales of operation between human and machine interpreter, and the human agency to make decisions about the score. A human can embrace failure and redefine the situation to continue beyond it, rather than stopping with an error message, at least until claimed by physical exhaustion.

There is an analogous relationship in the live coding of the sound; the more movement the dancer performs, the more the code for the sound is disrupted by the function within Texture. Within the performance, changes to the musical code-score are made not only on the accord of the live coder but also in response to the dancer's movement. The human programmer is then forced to abandon any sense of planning and control, and just work to influence the code-score that is in a state of flux.

Conclusion

As an improvisatory work with a notation that is written and then discarded during the piece itself, *SC<>BC* leaves no trace but in memory. The system is deployed in two parts, which engage with one another through mutual intervention within a feedback loop, and it is through this live process of engagement that the running system generates and becomes the work. *SC<>BC* involves software at its core, but in a sense is itself a piece of software: it is a system incorporating both human and computer actors, with an outcome that only becomes apparent through performance. We can therefore borrow the notion of deployment that is common amongst software engineers, namely, the point at which a system is placed 'in the wild'. This is where imagination meets reality, and indeed, where the work has the opportunity to border at the seams between imagination and reality and potentially affect both.

SC<>BC engages two sides; two practices (music, dance), two notations (musical, choreographic), two bodies (live coder, dancer) in mutual influence; the dancer's body interfering with the musician's sonic notation, and the live coder's sound interfering with the dancer's choreographic notation. This creates a feedback loop which passes through dancer, computer vision, live code, sound, machine listening, choreography, and back to the dancer, but the coder sits outside the loop, their body apparently disengaged apart from a fixed gaze into their laptop, and their typing fingers (see Image 3). The aim

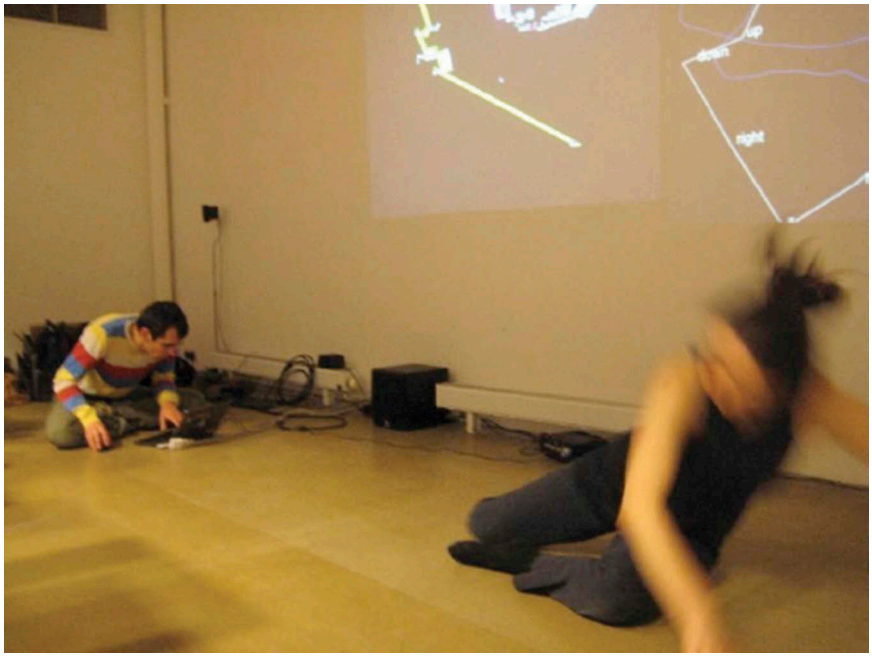

Image 3 Photo from March 2014 performance at Site Gallery, Sheffield, UK. Photograph by Susanne Palzer.

of this technological engagement is to create balance between the two practices, however in this sense the piece has not fully met this aim, and it seems that further development is needed to bring the programmer's body into the feedback loop too.

The choreographic and sonic notations are key to *SC<>BC*, but themselves only exist for the duration of the performance, the arc following the building up and taking apart of the code-scores, leaving no tangible trace. This is live notation, a kairotic practice,[11] which by its nature responds to the work as it unfolds, the notation depending on the notated and vice-versa. In this sense we can follow a line through the work, but as it leaves no trace behind, the notations hold no meaning outside of a particular performance.

11. Emma Cocker, 'Live Notation: Reflections on a Kairotic Practice', *Performance Research*, 18.5 (2014), 69–76.