



COVER SHEET

This is the author-version of article published as:

Brown, Andrew (2007) Code Jamming. *M/C Journal* 9(6).

Accessed from <http://eprints.qut.edu.au>

©2007 M/C

Code Jamming

Andrew R. Brown
Queensland University of Technology
a.brown@qut.edu.au
2006

Jamming culture has become associated with digital manipulation and reuse of materials. As well, the term jamming has long been used by musicians (and other performers) to mean improvisation, especially in collaborative situations. A practice that gets to the heart of both these meanings is live coding; where digital content (music and/or visuals predominantly) is created through computer programming as a performance. During live coding performances digital content is created and presented in real time. Normally the code from the performers screen is displayed via data projection so that the audience can see the unfolding process as well as see or hear the artistic outcome. This article will focus on live coding of music, but the issues it raises for jamming culture apply to other mediums also.

Live coding of music uses the computer as an instrument, which is “played” by the direct construction and manipulation of sonic and musical processes. Gestural control involves typing at the computer keyboard but, unlike traditional “keyboard” instruments, these key gestures are usually indirect in their effect on the sonic result because they result in programming language text which is then interpreted by the computer. Some live coding performers, notably Amy Alexander, have played on the duality of the keyboard as direct and indirect input source by using it as both a text entry device, audio trigger, and performance prop. In most cases, keyboard typing produces notational description during live coding performances as an indirect music making, related to what may previously have been called composing or conducting; where sound generation is controlled rather than triggered. The computer system becomes performer and the degree of interpretive autonomy allocated to the computer can vary widely, but is typically limited to probabilistic choices, structural processes and use of pre-established sound generators.

In live coding practices, the code is a medium of expression through which creative ideas are articulated. The code acts as a notational representation of computational processes. It not only leads to the sonic outcome but also is available for reflection, reuse and modification. The aspects of music described by the code are open to some variation, especially in relation to choices about music or sonic granularity. This granularity continuum ranges from a focus on sound synthesis at one end of the scale to the structural organisation of musical events or sections at the other end. Regardless of the level of content granularity being controlled, when jamming with code the time constraints of the live performance environment force the performer to develop succinct and parsimonious expressions and to create processes that sustain activity (often using repetition, iteration and evolution) in order to maintain a coherent and developing musical structure during the performance. As a result, live coding requires not only new performance skills but also new ways of describing the structures of and processes that create music.

Jamming activities are additionally complex when they are collaborative. Live Coding performances can often be collaborative, either between several musicians and/or between music and visual live coders. Issues that arise in collaborative settings are both creative and technical. When collaborating between performers in the same output medium (e.g., two musicians) the roles of each performer need to be defined. When a pianist and a vocalist improvise the harmonic and melodic roles are relatively obvious, but two laptop performers are more like a guitar duo where each can take any lead, supportive, rhythmic, harmonic, melodic, textual or other function. Prior organisation and sensitivity to the needs of the unfolding performance are required, as they have always been in musical improvisations. At the technical level it may be necessary for computers to be networked so that timing information, at least, is shared. Various network protocols, most commonly Open Sound Control (OSC), are used for this purpose. Another collaboration takes place in live coding, the one between the performer and the computer; especially where the computational processes are generative (as is often the case). This real-time interaction between musician and algorithmic process has been termed hyperimprovisation by Roger Dean (2003).

Jamming cultures that focus on remixing often value the sharing of resources, especially through the movement and treatment of content artefacts such as audio samples and digital images. In live coding circles there is a similarly strong culture of resource sharing, but live coders are mostly concerned with sharing techniques, processes and tools. In recognition of this, it is quite common that when distributing works live coding artists will include descriptions of the processes used to create work and even share the code. This practice is also common in the broader computational arts community, as evident in the sharing of flash code on sites such as Levitated by Jared Tarbell, in the Processing site (Reas & Fry 2003), or in publications such as Flash Maths Creativity (Peters et al. 2004). Also underscoring this culture of sharing, is a prioritising of reputation above (or prior to) profit. As a result of these social factors most live coding tools are freely distributed.

Live Coding tools have become more common in the past few years. There are a number of personalised systems that utilise various different programming languages and environments. Some of the more polished programs, that can be used widely, include Supercollider (McCartney 1996), Chuck (Wang & Cook 2003) and Impromptu (Sorensen 2005). While these environments all use different languages and varying ways of dealing with sound structure granularity, they do share some common aspects that reveal the priorities and requirements of live coding. Firstly, they are dynamic environments where the musical/sonic processes are not interrupted by modifications to the code; changes can be made on the fly and code is modifiable at runtime. Secondly, they are text-based and quite general programming environments, which means that the full leverage of abstract coding structures can be applied during live coding performances. Thirdly, they all prioritise time, both at architectural and syntactic levels. They are designed for real-time performance where events need to occur reliably. The text-based nature of these tools means that using them in live performance is barely distinguishable from any other computer task, such as writing an email, and thus the practice of projecting the environment to reveal the live process has become standard in the live coding community as a way of communicating with an audience (Collins 2003).

It is interesting to reflect on how audiences respond to the projection of code as part of live coding performances. In the author's experience as both an audience member and live coding performer, the reception has varied widely. Most people seem to find it curious and comforting. Even if they cannot follow the code, they understand or are reassured that the performance is being generated by the code. Those who understand the code often report a sense of increased anticipation as they see structures emerge, and sometime opportunities missed. Some people dislike the projection of the code, and see it as a distasteful display of virtuosity or as a distraction to their listening experience. The live coding practitioners tend to see the projection of code as a way of revealing the underlying generative and gestural nature of their performance. For some, such as Julian Rohrerhuber, code projection is a way of revealing ideas and their development during the performance. "The incremental process of livecoding really is what makes it an act of public reasoning" (Rohrerhuber 2006). For both audience and performer, live coding is an explicitly risky venture and this element of public risk taking has long been central to the appreciation of the performing arts (not to mention sport and other cultural activities).

The place of live coding in the broader cultural setting is still being established. It certainly is a form of jamming, or improvisation, it also involves the generation of digital content and the remixing of cultural ideas and materials. In some ways it is also connected to instrument building. Live coding practices prioritise process and therefore have a link with conceptual visual art and serial music composition movements from the 20th century. Much of the music produced by live coding has aesthetic links, naturally enough, to electronic music genres including musique concrète, electronic dance music, glitch music, noise art and minimalism. A grouping that is not overly coherent besides a shared concern for processes and systems. Live coding is receiving greater popular and academic attention as evident in recent articles in *Wired* (Andrews 2006), *ABC Online* (Martin 2006) and media culture blogs including *The Teeming Void* (Whitelaw 2006).

Whatever its future profile in the boarder cultural sector the live coding community continues to grow and flourish amongst enthusiasts. The TOPLAP site is a hub of live coding activities and links prominent practitioners including, Alex McLean, Nick Collins, Adrian Ward, Julian Rohrerhuber, Amy Alexander, Frederick Olofsson, Ge Wang, and Andrew Sorensen. These people and many others are exploring live coding as a form of jamming in digital media and as a way of creating new cultural practices and works.

References

Andrews, R. 2006. "Real DJs Code Live." *Wired, technology news*. 06 July 2006. <http://www.wired.com/news/technology/0,71248-0.html>

Collins, N. 2003. "Generative Music and Laptop Performance." *Contemporary Music Review* 22(4): 67-79.

Peters, K., Tan, M. and Jamie, M. (2004). *Flash Math Creativity*. Berkeley, CA: Friends of ED.

- Tarbell, Jared. *Levitated*. <http://www.levitated.net/daily/index.html>
- Fry, Ben and Reas, Casey. *Processing*. <http://processing.org/>
- Martin, R. 2006. "The Sound of Invention" *Catapult*, ABC Online. <http://www.abc.net.au/catapult/indepth/s1725739.htm>
- McCartney, J. 1996. "SuperCollider: A new real-time sound synthesis language." *The International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 257-258.
- Reas, C. and Fry, B. 2003. "Processing: a learning environment for creating interactive Web graphics." *International Conference on Computer Graphics and Interactive Techniques*. San Diego: ACM SIGGRAPH pp. 1-1.
- Rohrhuber, J. 2006. In a post to a live coding email list. livecode@slab.org. 10 September 2006.
- Sorensen, A. 2005. "Impromptu: An interactive programming environment for composition and performance." In A. R. Brown and T. Opie (eds.) *Proceedings of the Australasian Computer Music Conference 2005*. Brisbane: ACMA pp. 149-153.
- TOPLAP. <http://toplap.org/>
- Wang, G. and Cook, P. R. 2003. "ChucK: A Concurrent, On-the-fly, Audio Programming Language." *International Computer Music Conference*. ICMA, pp. 219-226
- Whitelaw, M. 2006. "Data, Code & Performance." *The Teeming Void*. 21 September 2006. <http://teemingvoid.blogspot.com/2006/09/data-code-performance.html>