Considering Interaction in Live Coding through a Pragmatic Aesthetic Theory

Renick Bell*

*Tama Art University, Japan

renick@gmail.com

Abstract

Live coders can use aesthetic evaluation to improve their work. A pragmatic aesthetic theory, based on the writings of John Dewey (one of the central philosophers of pragmatism), can be employed for such evaluations. In this revised theory, emotional states (affects) are experienced by audience members (affectees) as a result of experiencing a network of percepts (affectors). An affectee assigns value to an experience according to how well it achieves the affectee's intentions. When assembling the network of affectors which will constitute an experience, the live coder can anticipate the resulting affects and use those predictions to attempt to improve the experience. The experience of live coding consists of a network of affectors such as the musical output, programming languages, software libraries, and projection contents. To the extent that affectees, including the performer as first audience member, are aware of the interaction, the interaction method becomes an affector in that network. It can function either indirectly as a result of its influence on other affectors or directly when perceived as a primary element of the experience. An interaction method itself is a compound affector, consisting of various influencing aspects: usability, appearance, and so on. Audience purposes can range from dancing to deep consideration. Performers also have various purposes, including enjoying exploration, elucidating abstractions, or obfuscating for the creation of mystery. Examining live coding interaction techniques and their antecedents for the affects they cause and how well they achieve the intended ends may show potential advances available to contemporary live coders.

Keywords: live coding, aesthetics, pragmatism

1. Introduction

This paper gives a working definition of live coding, followed by a brief history of live coding. The next section presents a pragmatic aesthetic theory derived from the writings of John Dewey, and Dewey's theory of valuation is summarized. Interaction in live coding is then examined. The final section discusses through a concrete example how such interaction might be evaluated in order to improve live coding performances.

2. Defining Live Coding

Live coding is the interactive control of algorithmic processes through programming activity (A. R. Brown, 2007; Collins, 2011; Ward et al., 2004). This paper focuses on interacting with code as a performance and does not deal with public programming as a tutorial. Live coding is by custom projected for an audience to see. It also typically includes improvisation (Collins, McLean, Rohrhuber, & Ward, 2003). The style of music is not fixed, meaning live coding is a performance method rather than a genre (A. R. Brown & Sorensen, 2009).

3. Live Coding History

Sorensen writes that the first documented live coding performance was carried out by Ron Kuivila at STEIM in 1985 (Sorensen, 2005). The piece, "Watersurface", was done in a "precursor of Formula" (a programming language developed by Anderson and Kuivila) and involved Kuivila "writing Forth code during the performance to start and stop processes

triggering sounds through [a Mountain Hardware 16 channel oscillator board]" (R. Kuivila, personal communication, Nov. 6, 2013). "[The] original performance apparently closed with a system crash." (Kuivila et al., 2007)

The Hub used shared data to carry out performances from 1985 to the early 90s (Gresham-Lancaster, 1998). Earlier (1980-82), the League of Automatic Composers did extended performances with programs which were adjusted as they ran (C. Brown & Bischoff, 2002). The degree to which such tuning required coding is unspecified in existing literature.

Gresham-Lancaster cites Cage and Tudor as influences (Gresham-Lancaster, 1998). This impact is supported by Nyman (Nyman, 1999), who contrasts experimental music like that of Cage, Steve Reich, and Fluxus with what he calls the avant-garde. Nyman writes that experimental music focuses on situations in which processes work across a field of possibilities to bring about unknown outcomes (p. 3) and show the uniqueness of particular moments (p. 8). Such music may have an identity located outside of the final audience-perceived auditory material (p. 9), have non-traditional methods for dealing with time (p. 10), and force performers to use skills not typically associated with musicianship (p. 13). Experimental music frequently presents performers with surprise difficulties in performance (p. 14), resemble games (p. 16), and give performers rules to interpret (p. 16). Who the performers are may be ambiguous (p. 19). It may require new ways of listening (p. 20). These issues appear in live coding frequently, such as the need for new performance skills (A. R. Brown, 2007, p.1) and the similarity to games (Magnusson, 2011). The one thing that appears to separate live coding from this experimental music tradition is that Cage, Fluxus, and related composer/performers seem not to have changed the rules during performances, while this is a central concern for live coding. Further examination of the 20th century experimental music tradition for hints about live coding should be done.

Following an apparent gap in the 90s, live coding activity increased in the first decade of the 21st century (McLean & Others, 2010), early examples being Julian Rohrhuber's experiments in live coding with SuperCollider and the duo of Alex McLean and Adrian Ward called slub (Collins et al., 2003). In the previous decade, live coding activity through systems like McLean's feedback.pl, Dave Griffith's Fluxus, and computer music languages SuperCollider and Chuck, through to newer systems like Extempore, Overtone, and the author's Conductive, a growing culture and body of work has developed.

4. A Pragmatic Aesthetic Theory

Live coding can be considered aesthetically with a pragmatic aesthetic framework proposed by Bell based on Dewey's "Art as Experience", published in 1934 (Dewey, 2005). Dewey (1859-1952) was an American philosopher and educator and one of the central figures of the philosophical movement called Pragmatism, along with William James and Charles Sanders Pierce. (Hookway, 2013)

Dewey explains pragmatism as clearly defining an idea and seeing how it works "within the stream of experience" to show how "existing realities may be changed" and to produce plans for effecting such change. In pragmatism, "theories... become instruments" allowing ideas to be judged valuable or not from their consequences (Dewey, 1908).

Before summarizing this author's recent revision, it is useful to consider the main points of "Art as Experience". This summary is basically taken from (Bell, 2013). Dewey calls art "a process of doing or making" (Dewey, 2005) and an engagement with intention. An external product is a potential art experience depending on its audience. For Dewey it is preferable to think of art as an experience, and a painting or performance as a tool through which that experience can be realized. This leads to a "triadic relation" in which the creator produces something for an audience which perceives it. What the creator has produced creates a link between the creator and the audience, though sometimes the creator and the audience are the same (creator as first

audience member). For Dewey, experience also means an interaction with an environment that is unavoidably human and creates a feedback loop in which actions and reactions affect one another. These experiences are always composed of both physical and mental aspects. Experiencing the world means transforming it "through the human context", and equally being transformed. A potentially infinite number of experiences can be derived from a single artifact or situation (Leddy, 2012). Because art is experience, it is always temporal in nature. Dewey discards the distinction between "fine" and "useful" art.

5. A Revised Pragmatic Aesthetic Theory

There are some problematic points to Dewey's theory, and as a result it has been revised by Shusterman (Shusterman, 2000) and by McCarthy and Wright to explain interaction with technology (McCarthy & Wright, 2004), among others. This author presented a revision in (Bell, 2013) and a summary in (Bell, 2013). That summary is presented below with some modifications.

An affect is an emotional state. An affectee is a person experiencing affects in an interaction with affectors. An affector is a percept that stimulates affects in an affectee. It can be a physical object or something abstract. A work of art is an affector which in some way was created, organized, or manipulated with the intention of it being an affector. A person involved with the creation or arrangement of an affector is an artist.

An art experience is the experience of affects in an affectee as the result of the affectee's interaction with a network of affectors, with at least one of those affectors being a work of art. The art experience is the experience of those affectors either simultaneously or in sequence. Experience involves a possibly infinite number of affectors arrayed in a network structure in which they influence each other and influence the affectee either directly or indirectly. Changing the perceived network of affectors changes the nature of the experience.

6. Dewey's Theory of Valuation

Dewey wrote a considerable amount of material on valuation. His theory can be summarized as follows. The value of something derives from how well it suits the achievement of an individual's intentions and the consequences of achieving those ends through those means. The object of an appraisal is also evaluated while considering its consequences with respect to other intentions held by the individual (Dewey, 1939). Everything of value is instrumental in nature. Every end is in turn a means for another intention in a continuous stream of experience. Value cannot be assigned in a disinterested manner (Dewey, 1939). Value is assigned to an experience according to the context of the experience (including but not limited to the culture it takes place in (Dewey, 1939)). Such judgments are always in flux and susceptible to revision based on newly obtained experience. Valuations are instrumental for future valuations and action (Dewey, 1922). These are used to control the stream of an individual's experience (Dewey, 1958).

In addition, some points can be made related to the revised aesthetic theory above. The value of an affector is connected to the value of an art experience in which it is involved. The value of an art experience is determined by the affects experienced (Bell, 2013) and how well those affects and the other consequences of the experience and its affectors suit the intentions of the affectee.

7. Considering Interaction in Live Coding with the Revised Aesthetic Theory

Applying this aesthetic theory to an experience of live coding means:

1. analyzing the intentions held by a performer or audience member

2. determining the network of affectors that are present in a performance

3. examining the consequences of the interaction with those affectors, including resulting affects

4. determining the relationship between those consequences, affects, and the originally-held intentions

5. assigning value to the experience and its affectors according to those intentions or changing intentions

This process bears some similarity to the technique for analysis of the experience of technology described in (McCarthy & Wright, 2004).

8. Intentions

Live coders have expressed a broad and diverse set of intentions (Magnusson, 2011), though the central and common intention is the real-time creation and presentation of digital content (A. R. Brown, 2007), particularly through use of algorithms (A. R. Brown & Sorensen, 2009; Thielemann, 2013). Some want the challenge and the chance to improvise (Collins et al., 2003). Some desire flexibility of expression (Blackwell & Collins, 2005; Magnusson, 2011). Some seek to do so collaboratively (C. Brown & Bischoff, 2002; Sorensen, 2005; Thielemann, 2013). Some aim to communicate algorithmic content to an audience (A. R. Brown, 2007, p. 3), making clear for them the coder's deliberations (McLean, Griffiths, Collins, & Wiggins, 2010; Sorensen, 2005), as well as showing how that activity is guided by the human operator (A. R. Brown & Sorensen, 2009, pp. 9–10). Some want to demonstrate virtuosity (Sorensen, 2005), interact more deeply with a computer (A. R. Brown & Sorensen, 2009; Collins et al., 2003), or discover new musical structures (Sorensen, 2005). This can mean trying to describe generative processes in efficiently (A. R. Brown & Sorensen, 2009), either in terms of computational power necessary or code necessary to express an idea. The intention can even be ironic and in opposition to the goal of clear expression for the audience (Zmoelnig, 2012).

9. Interaction in Live Coding

An interaction method is a compound affector, consisting of various influencing aspects like usability, appearance, and historical position. This comes into relation with what is being interacted with: a programming language and its notation, algorithmic processes, a synthesizer, and so on.

The custom of projecting the coding activity for the audience makes the projected interaction one of the first affectors in the experience of live coding. Though live coding systems are quite personal (Magnusson, 2011), interaction in live coding can be classified into two superficial categories based on this visual display: an orthodox style and idiosyncratic styles. Though not perfectly uniform, the orthodox style involves a text editor and an interpreter, and it can be observed in some live coding performances by McLean and those of Sorensen among others. Idiosyncratic styles may or may not involve the former, but they can include graphics, animation, or other interactive elements. Examples of idiosyncratic live coding interaction styles include some performances and systems by Griffiths (McLean et al., 2010), Magnusson (Magnusson, 2013), and Zmoelnig (Zmoelnig, 2012).

One example of the orthodox style of live coding interaction is that in Alex McLean's performances using his Haskell library Tidal. In addition to its pattern representation and manipulation features, it allows the use of GHCi and Emacs to live code patterns and trigger a synth over OSC (McLean & Wiggins, 2010).

An example of the idiosyncratic style of live coding is Dave Griffith's Scheme Bricks, which is a graphical environment for programming in the Scheme language which trades the signature parentheses of Scheme for colored blocks. It allows the user to graphically manipulate fragments of code in a way the author feels differs from text editing as well as preventing coding mistakes like mismatching the number of parentheses (McLean et al., 2010).

Regardless of the superficial appearance of the live coding, many fundamental aspects are shared. McLean makes clear one of the challenges of interaction in live coding is the higher level of abstraction for making sounds in live coding compared to manipulation of a traditional physical instrument (McLean & Wiggins, 2010). The coder creates those many sounds by means of algorithmic processes. Brown characterizes those processes as "typically limited to probabilistic choices, structural processes and use of pre-established sound generators." (A. R. Brown, 2007, p. 1) Those algorithmic processes are mapped to synthesizers. This creates some tension for the performer, who must juggle two somewhat dissimilar types of interaction: one with the algorithmic processes, and another with the synthesizer.

In order to control these algorithmic processes, a user employees abstractions and the notation defined to express them in a given language. A more complete discussion of these abstractions can be found in (Bell, 2013), but the level of abstraction among live coders can also vary, and there is an extreme diversity in the notations used to express them, such as the S-expressions employed by Sorensen and Griffiths, SuperCollider code used by Rohrhuber and others, and even differences in the Haskell used by McLean and Bell. Naturally the idiosyncratic methods mentioned previously provide somewhat different means for controlling these processes.

The nature of live coding when presented with a projection emphasizes an interaction with the audience in a way that other electronic music does not. McLean and Wiggins question the affects of the audience as a result of experiencing the projection, suggesting that some affectees may experience alienation even while others appreciate the opportunity to see the coder's interaction with the system (McLean & Wiggins, 2011). While their anecdotal evidence says both are possible, that gathered by Brown suggests a positive reaction to be more common. However, he also notes that it can be perceived as showing off or a distraction (A. R. Brown, 2007). This partly depends on the affectee's background (Bell, 2013). Considering their intentions is also important: audience purposes can range from dancing to deep consideration. For example, overemphasis of projected code might be a mismatch for an affectee wanting to dance.

10. Evaluating Interaction in Live Coding

With such a diverse set of intentions and affectees, clearly evaluating the interaction might seem to be an impossible task. The aesthetic theory presented says a potentially infinite number of evaluations could be obtained. However, remembering that the theory is intended as a useful tool for suggesting a plan for change, it seems desirable to apply the system even partially. One strategy may be to focus on one intention at a time for one affectee. Given a particular performance, the affectors involved can be listed out. Such an example follows.

One of the author's recent performances took place on Saturday, May 11 2013 at the Linux Audio Conference in Graz, Austria in the basement of the Forum Stadtpark (Bell, 2013). One of the intentions of the performance was maximizing opportunities to improvise.

At the performance, an audience of perhaps 50 or 60 people stood in front of a low stage. The lights were turned off when the audio began. This performance of bass music emphasized generative rhythms. A custom live coding system, a programming library called Conductive, was used to trigger a simple sampler built with the SuperCollider synthesizer and loaded with thousands of audio samples. An orthodox style was used in which prepared code was loaded into the vim editor, edited, and sent to the Haskell interpreter, where it was executed. Doing so, multiple concurrent processes spawned events and other parameters. Abstractions were used to generate sets of rhythmic figures, which were paired with patterns of audio samples and other synthesis parameters. The concurrent processes read the generated data and used it to synthesize sound events. Such data was generated and repeatedly chosen from to allow improvised music making. By watching the projected interaction, the

audience could to some extent observe the generative processes.

In this case, the dark performance environment made it somewhat difficult to see the audience. As a result, some trepidation about audience reaction (wanting to encourage the audience to dance being another intention) served as a limiting factor to perceived freedom. It occupied some attention that might have been spent otherwise had a dancing audience been observable from the beginning. The software library made switching between patterns and designing time-varying parameter changes simple, but insufficient familiarity with the library functions and the design of the library itself meant there were not opportunities for generating new rhythm patterns during the performance. A sense of restriction resulted to some extent.

Putting the experience to use, it seems that productive changes might include simplifying the generation of new rhythm patterns by changes or additions to the library, somehow obtaining a better view of the audience, and practicing more. The original intention remains an important one.

11. Conclusion

While obtaining a complete analysis for every intention for even one affectee is a very large task and beyond the scope of this paper, it is hoped that this example analysis shows how this theory can be applied. Given enough planning and resources, it is thought that it could be applied more comprehensively to a larger group of affectees with the hopes of obtaining useful evaluations that can then be employed to improve future performances.

References

Bell, R. (2013). Pragmatic Aesthetic Evaluation of Abstractions for Live Coding. *The Japanese Society for Sonic Arts.* Retrieved from http://www.jssa.info/doku.php?id=journal017

Bell, R. (2013). Towards Useful Aesthetic Evaluations of Live Coding. *Proceedings of the International Computer Music Conference.*

Bell, R. (2013, may). *Renick Bell, live @ Linux Audio Conference 2013, Saturday, May 11, Forum Stadtpark, Graz, Austria*. Retrieved from http://www.youtube.com/watch?v=J5TskLgsdBU

Blackwell, A., & Collins, N. (2005). The Programming Language as a Musical Instrument. *Proceedings of PPIG05*. University of Sussex.

Brown, A. R. (2007). Code jamming. *M/C: a journal of media and culture*, *9*(6).

Brown, A. R., & Sorensen, A. (2009). Interacting with Generative Music through Live Coding. Contemporary *Music Review*, *28*(1), 17–29.

Brown, C., & Bischoff, J. (2002). Indigenous to the Net: early network music bands in the San Francisco Bay area. Retrieved from http://crossfade.walkerart.org/brownbischoff/IndigenoustotheNetPrint.html

Collins, N. (2011). Live Coding of Consequence. *Leonardo*, *44*(3), 207–211.

Collins, N., McLean, A., Rohrhuber, J., & Ward, A. (2003). Live coding in laptop performance. *Organised Sound*, 8(03), 321–330.

Dewey, J. (1908). What Does Pragmatism Mean by Practical? *The Journal of Philosophy, Psychology and Scientific Methods, 5*(4), 85–99.

Dewey, J. (1922). Valuation and experimental knowledge. *The Philosophical Review, 31*(4), 325–351.

Dewey, J. (1939). Theory of valuation. *International Encyclopedia of Unified Science*.

Dewey, J. (1958). *Experience and nature* (Vol. 1). DoverPublications. com.

Dewey, J. (2005). *Art as Experience*. Perigee Trade.

Gresham-Lancaster, S. (1998). The Aesthetics and History of the Hub: The Effects of Changing Technology on Network Computer Music. *Leonardo Music Journal, 8*, 39.

Hookway, C. (2013). Pragmatism. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy* (Winter 2013.). Retrieved from http://plato.stanford.edu/archives/win2013/entries/pragmatism/

Kuivila, R., Hub, T., Rohrhuber, J., Mogini, F., Collins, N., Kamensky, V., ... IN SAND. (2007). *A Prehistory of Live Coding*. TOPLAP. Retrieved from http://www.sussex.ac.uk/Users/nc81/toplap1.html

Leddy, T. (2012). Deweyś Aesthetics. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy* (Fall 2012.). Retrieved from http://plato.stanford.edu/archives/fall2012/entries/dewey-aesthetics/

Magnusson, T. (2011). Confessions of a live coder. *Proceedings of International Computer Music Conference.*

Magnusson, T. (2013). The Threnoscope. *Proceedings of the 2013 International Conference on Software Engineering.*

McLean, A., & Others. (2010). *TOPLAP* website. Retrieved from http://www.toplap.org/index.php/Main_Page

McLean, A., & Wiggins, G. (2010). Tidal–Pattern Language for Live Coding of Music. *Proceedings of the 7th Sound and Music Computing conference*.

McLean, A., & Wiggins, G. (2011). Texture: Visual Notation for Live Coding of Pattern. *Proceedings of the International Computer Music Conference.*

McLean, A., Griffiths, D., Collins, N., & Wiggins, G. (2010). Visualisation of live code. *Proceedings of Electronic Visualisation and the Arts 2010.*

Nyman, M. (1999). *Experimental Music: Cage and Beyond (Music in the Twentieth Century) (2nd ed.)*. Cambridge University Press. Retrieved from http://www.worldcat.org/isbn/0521653835

Shusterman, R. (2000). *Performing live: Aesthetic alternatives for the ends of art*. Cornell University Press.

Sorensen, A. (2005). Impromptu: An interactive programming environment for composition and performance. *Proceedings of the Australasian Computer Music Conference 2009.*

Thielemann, H. (2013). Live music programming in Haskell. *arXiv preprint* arXiv:1303.5768.

Ward, A., Rohrhuber, J., Olofsson, F., McLean, A., Griffiths, D., Collins, N., & Alexander, A. (2004). Live algorithm programming and a temporary organisation for its promotion. *Proceedings of the README Software Art Conference.*

Wright, P., & McCarthy, J. (2004). *Technology as experience*. MIT Press.

Zmoelnig, I. M. (2012, dec). *Pointillism*. Retrieved from http://umlaeute.mur.at/Members/zmoelnig/projects/pointillism/