# Generative Content Co-creation:

## Lessons from algorithmic music performance

Andrew R. Brown
Interactive Media Lab
Griffith University
Brisbane, Australia
email: andrew.r.brown@griffith.edu.au

**Abstract—This article examines features of algorithmic music performance practices and considers how these might be applied to other generative content creation contexts. Based on the assumption that all generative processes are performative, the article draws from an analysis of live algorithmic music to outline lessons that may be more widely applicable to content co-creation with algorithmic systems. Methods discussed include algorithm selection and expression, the architecture of algorithmic system design, the effects of materiality on algorithmic performance, and how co-creative strategies openly embrace the influence of humans as agents in generative content systems. Having distilled and articulated these methods in this article it is anticipated that future research will apply them to generative media system beyond music performance for evaluation of their generalizability.**

***Keywords-Generative; Media; Creative; Content; Music; Algorithmic.***

## I. INTRODUCTION

Algorithmic content creation for fixed and dynamic works has a rich history, both in academic circles and in commercial contexts. For example, in the use of procedural content in computer games and interactive installations. Even if generative media outcomes are not dynamic the process of algorithmic content creation is, both in the system design and in the computational rendering. Algorithmic content creation always has a performative element. With this in mind, this article will reflect on the characteristics of performative algorithmic practices, in particular live coding and interactive computer music, and highlights aspects of these practices that are relevant for the design of generative content systems more broadly defined.

Generative media have been used in many contexts, from graphic design to architecture, and employ many techniques, from rule-based models to generative adversarial networks. Uses for generative media include customizing individual products based on templates or stylistic patterns, the design of complex artefacts with many dimensions of components, and the production of emergent or evolving experiences that adapt to changing contexts.

Within contemporary societies the production of media content is generally considered to be a creative act. Within this context a creative computational system can been defined as "a collection of processes, natural or automatic, which are capable of achieving simulating behavior which in humans would be deemed creative" [1]. Following Csikszentmihalyi [2], it is generally accepted that creative practice involves the production of novel and useful outcomes from work done within a conceptual space of acceptable outcomes. In the creative computation literature, there is often a distinction between (mere) generation for human selection from outcomes and generation for (self) evaluation by the machine [3]. In this article, the focus is on algorithmic systems used for generative co-creation [4] between humans and machines, such as those used in generative design or adaptive game music engines. Co-creation, in these contexts, is understood as "collaborative creativity where both the human and the computer take creative responsibility for the generation of a creative artefact" [5].

Section II will discuss approaches and considerations in algorithmic music practices that are relevant to generative media content creation. Section III will explore how performative interaction with generative systems is the basis for co-creation between people and machines.

## II. LESSONS FROM LIVE ALGORITHMIC MUSIC

Generative algorithms are used in a range of music performance practices. These include those characterized as interactive music systems [6], live algorithms [7], networked music performance [8], or live coding [9]. Such practices will be collectively attributed here as live algorithmic music, and their practitioners as algorithmic musicians.

According to Lewis, live algorithmic music systems are interesting because they "produce a kind of virtual society that both draws from and challenges traditional notions of human interactivity and sociability, making common cause with a more general production of a hybrid, cyborg sociality that has forever altered both everyday sonic life and notions of subjectivity in high technological cultures" [10]. More specifically, it has been suggested that live algorithmic music practices convey three important attributes; algorithmic thinking, real-time creativity, and networked collaboration [11]. In the following sections each will be explored in turn.

### *A. Algorithm choice and design*

Live algorithmic music is typically performed from memory and without time to consult reference materials, so it is sensible that algorithmic musicians seek to identify a small cohort of functions that are widely applicable to a variety of

 ISBN: 978-1-61208-707-8 

musical circumstances. An advantage of focusing on a small set of coding patterns means these can be thoroughly understood and flexibly employed. This approach contrasts with many digital media software applications that boast the extensive array of functions available, most of which users will never use or, when they do, are 'one trick ponies' that soon become kitsch. Selection of such a set of foundational functions is likely to be media-specific and will have aesthetic implications. An example of such a selection is from the live coding duo *aa-cell* who suggest the following list of algorithms; probability, linear and higher order polynomials, periodic functions, modular arithmetic, set theory, and recursion [12]. They comment that "what has been a surprise to us, however, is just how much utility a small set of processes have provided" [12]. However, they also warn that correlations between mathematical functions and musical patterns are limited.

In addition to the choice of algorithm, algorithmic musicians often focus on "the way in which generative algorithms are represented in code to best afford interaction and modification during performance" [13]. They are aware that algorithm selection and design can significantly influence the flexibility and responsiveness of generative processes. So, they are concerned to choose the best parametric variations and constraints that maximize novelty whilst avoiding system misbehavior in the form of inappropriate output.

The balance of stability and novelty is at the heart of music making and is often stressed as key to other creative arts and to creativity itself. Algorithm design needs to enable the tuning of a system that walks a line between output that is neither boring nor inappropriate. Stability is often provided by incorporating domain knowledge into generative systems and novelty is often provided by processes with unpredictable outcomes.

One risk in overloading a system with domain specifics and constraints, is the restriction of output variety. Methods, such as genetic programming, can avoid such constraints by automating algorithm construction, not simply algorithm execution. However, the challenges of reliably automating meta-structural choices are many.

## B. Compact code

It is important in live coding performances to manage the amount of code being typed on stage. Succinct expression of ideas and outcomes allows for efficient expression of ideas and responsiveness to contextual changes. Succinct expression relies on the language used, algorithm design, and interface for interaction.

Compactness of code is highlighted by Farnell as a desirable principle of procedural audio design, he argues that "compact code can be useful for purely software development reasons, being easy to understand, extend, and maintain" [14]. The advantage of code succinctness for creative expression is elsewhere emphasized: "The description length and complexity of an algorithm plays a large factor in its appropriateness for live coding... we consider algorithmic directness one of the most powerful aspects of live coding" [13].

However, compactness is not the algorithmic musician's only criteria. Many also seek 'descriptive transparency' defined as the "ability to further interact with the algorithm at a syntactic level" [13]. In practice this means that algorithms are designed to be modifiable through exposure of appropriate parameters. Descriptive transparency can be in tension with succinct expression as too concise a representation may obscure opportunities for interaction or variation by hiding key parameters or commitments.

## C. Structure and abstractions

The choice of software abstractions makes a substantive difference to the ability to express ideas, in both a positive and a negative way [15]. The algorithmic musician constructs mental images of predicted outcomes and devises strategies to achieve them. In computing we use predefined structures, probabilistic decisions or situationally responsive processes, just like mental models—to guide prediction and planning. So computational abstractions need to be conditioned by limits or rules that guide effective outcomes.

Algorithmic musicians emphasize the importance of hierarchy in managing complexity and affording significant change with minimal high-level adjustments. This is necessary in live algorithmic music because "there is no possible way for us to deal with the complexity of the underlying operating system and hardware without levels of abstraction" [13]. In generative content systems, abstractions too are more than a structural convenience, they instil constraints and affordances and help represent a model of the content domain [16]. This push toward domain-specific abstraction has resulted in mini-languages being developed by algorithmic musicians to minimize cognitive load and enhance coding efficiency when performing.

Along with abstractions, the use of familiar metaphors in software design is also emphasized for reducing cognitive load for users. Brown and Sorensen [13] mention the deliberate use of familiar names for functions in their live coding practice—for example, bass, melody, and ambient—so that audience members unfamiliar with computer languages can make more sense of the projected code. When designing algorithmic music systems, Hoeberechts and co-authors [19] emphasize the importance of using real-world metaphors for system components, including 'instrument', 'performer', 'mood' and the like. Metaphors are also used to manage system organization. For example, in describing musical processes in terms of well understood practices such as describing software functions as 'players' that execute 'scores' on 'instruments'. The power of such analogy has also been emphasized by other studies of creativity and computation [18].

Metaphors can be useful for describing high-level parameters for algorithmic control. This should allow generative media systems to be guided by human interaction to produce a wide range of suitable outcomes. If abstractions are well chosen, then expressiveness and diversity of content output should be enhanced.

 ISBN: 978-1-61208-707-8 

### D. Networked architecture

Multiple dimensions can exist within one generative outcome. For example, a musical melody contains pitch, rhythm, timbre, volume and so on. However, relationships also exist between elements, like musical parts in a score, or visual components in a scene. Algorithmic musicians have found it useful to organize their code and their collaborations such that these elements correspond to a network of relationships. Networked musical performance practices include those with a focus on distributed interaction over the internet or, perhaps more pertinent here, distributed multi-agent systems whose architectures model the interdependence of sub components of the media being generated.

In many algorithmic music environments, concurrency is a key coding strategy to achieve interdependent modularity. Multiple concurrent operations are often conceived as 'loops' or 'processes' that act independently even if they share data. More formally, they have been implemented as 'temporal recursions'—code functions (closures) that call themselves periodically and maintain their own state [19].

Formalized protocols have been developed by algorithmic musicians for networked generative system architectures. A recent example is the Musebot (musical robot) framework designed to explore "the affordances of a multi-agent decision-making process" [20]. The Musebot protocol establishes "a parsimonious set of parameters for effective musical interaction between musebots" [20]. The open source specification includes a state-driven communication system for coordinating activities between agents. Messages are not themselves defined but are decided upon by cooperating developers. The objective of this approach is to compartmentalise the generative processes into components that manage complexity and enable flexible modular design and reuse.

### E. The materiality of algorithms

Algorithms are made concrete using electromechanical means. When so constituted "an algorithm is a statement (in Foucault's meaning of the word), that operates, also indirectly, on the world around it" [21]. Algorithms that manifest as music machines have existed throughout history, as evident in the well-known player piano. The physicality of such machines conditions outcomes from them, for example through limits on speed of operation, resolution of output, and accuracy of calculation. In short, materiality matters.

According to Sorensen and Gardner, the "traditional view [in computer science] is to promote a strong separation between the *program*, *process* and *task* domains" [19]. Such separation may be counterproductive to effective media outcomes because it ignores the material implications of the world in which the computational processes are engaged. They suggest, instead, a model of 'cyber-physical programming' that acknowledges the temporal bounds of real-time computation and the interactions with physical media; such as sound playback systems, electronic circuitry, or 3D printing materials.

For algorithmic musicians, the affordances of computing machinery and software are particularly felt in relation to time. Music, as a temporal art form, relies on very precise timing for musical expression and sonic fidelity. Temporality is also pertinent for other interactive generative systems, such as computer games.

For performers or game players, feedback about the ongoing generative process is often expressed as real-time audio-visual output. The materiality of algorithmic media imposes limits on operations and provides feedback to human participants who can, in response to that feedback, become active agents in the generative process. Thus, materiality becomes the basis for interaction with generative process.

## III. THE HUMAN IN THE LOOP

Almost by definition, algorithmic musicians are continually interacting with real-time generative processes. While such intimate co-creation may not always be true for all content generation systems, none are free from the influence of designers and programmers. Therefore, methods of co-creation with algorithms need to be taken into account.

### A. Embodiment

In musical performance on acoustic instruments, sound strongly implies causality and agency [21]. In algorithmic music performance this connection can seem less direct, however, as Farnell suggests, "above all, it is important that we remain mindful of sound as a process of transforming energy" [14]. Even though Emmerson suggests that "electricity and electronic technology have allowed (even encouraged) the rupture of these relationships of body to object to sound" [22] the impact of gesture and (implied) action remain important in algorithmic media. When developing computational systems for generative media content, we should not lose sight of how human agency is implicated in the outcome.

Generative models of music often focus on emulating musical theories or musical cognition. Algorithms based on these theories need to take into account the performative aspects of music. When algorithmic musicians are producing music, they pay attention to sonic expression alongside compositional structure. Techniques that can be applied to both are discussed in [12] whilst techniques that focus on musical expression in particular are explored elsewhere [23].

Outside of music, the modelling of human creative gesture is well established. For example, in systems for digital drawing and character animation, or in the use of style transfer by machine learning systems for artistic practice. In music studies, the role of gesture is well explored [24], as is expressive gesture as a musical 'force' that guides expectations [25]. The implications of for generative content creation systems include consideration of a role for direct human motion in algorithm control, or for motion capture or physics simulation to animate parametric movements in an organic way.

 ISBN: 978-1-61208-707-8 

### B. Interaction

When addressing the role of visual feedback for audiences, live coders included in their TOPLAP manifesto (available online) a fundamental principle; "show us your screens". Projecting code during performances, it is hoped, will make visible the actions (typing) of the performer. For live coders themselves, visual feedback is also provided by the text editor which acts as their user interface to code acting as a musical score. In live coding, interaction is mediated by reading and writing code. Relatedly, recent explorations with the Musebot protocol have included human integration into multi-agent music systems using "a 'code-wrapper' around the human player—whimsically termed an *algoskin*" [26] that enables a human performer to appear to the network like another musebot. Other algorithmic musicians' employ various interfaces with algorithms, often via controllers employing combinations of buttons, dials and sliders that trigger functions and manipulate parameters. This field of interaction design for music is so active it has its own conference, i.e., New Interfaces for Musical Expression (NIME).

At issue here is the expression of 'liveness' [27], "a sense that the person playing is contributing to that emotive energy through the performance decisions being made" [28]. More generally our interest is in the contribution of performative interaction on the outcome of the generated digital media. This is particularly important for interactions between people and machines in co-creative algorithmic systems.

So, how can a person be an active co-creator? Dahlstedt suggests the following categorization; "You can play *on*, *in* or *with* an algorithm" [21]. Performing 'on' an algorithm means to control its parameters. Performing 'with' an algorithm means to undertake your own activities in parallel to the algorithm's without influencing it. Performing 'in' an algorithm means that actions of the algorithm and human are socially coupled [29] such that each interaction has an effect on other parts of the cybernetic system.

A fourth category of co-creation is the ability for the human to redefine the generative process as it executes. As is the case in virtuosic live coding performances. According to Magnusson this "seems to be a logical and necessary step in the evolution of human–machine communication" [11].

## IV. CONCLUSION

The production of generative media requires creators to design the behaviors of algorithmic systems. In this way content outcomes are managed by the specification of creative behaviors rather than only by direct manipulation of materials. Behaviors are performative, and so we can learn from the performing arts how algorithmic behaviours lead to creative outcomes. Computational performing arts, such as live algorithmic music, have a special role to play in revealing pertinent practices applicable to generative content.

This article summarized live algorithmic music practices to assemble, for the first time, a consolidated set of lessons that may be helpful for co-creative content production. Methods that were identified include algorithm selection, algorithmic system architecture, the effects of materiality on algorithmic behaviour, and the influence of humans as co-creative agents. Future research will look at implementing these methods in prototype generative media systems for evaluation.

Lewis philosophically suggests that the impact of live algorithmic music may even reach beyond these lessons, that "perhaps our improvising computers can teach us how to live in a world marked by agency, indeterminacy, analysis of conditions, and the apparent ineffability of choice" [10].

## REFERENCES


[1] G. Wiggins, "A preliminary framework for description, analysis and comparison of creative systems," *Journal of Knowledge Based Systems*, vol. 19, no. 7, pp. 449–458, 2006.

[2] M. Csikszentmihalyi, "The Domain of Creativity," in *Changing the World: A framework for the study of creativity*, London: Praeger, 1994.

[3] S. Colton, A. Pease, J. Corneli, M. Cook, and T. Llano, "Assessing Progress in Building Autonomously Creative Systems.," in *ICCC*, pp. 137–145, 2014.

[4] T. Lubart, "How can computers be partners in the creative process: classification and commentary on the special issue," *International Journal of Human-Computer Studies*, vol. 63, no. 4–5, pp. 365–369, 2005.

[5] A. Kantosalo, J. M. Toivanen, P. Xiao, and H. Toivonen, "From Isolation to Involvement: Adapting Machine Creativity Software to Support Human-Computer Co-Creation.," in *Proceedings of the International Conference on Computational Creativity*, Ljubljana, Slovenia, pp. 1–7, 2014.

[6] R. Rowe, *Interactive Music Systems: Machine listening and composing*. Cambridge, MA: The MIT Press, 1993.

[7] T. Blackwell and M. Young, "Live Algorithms," *Artificial Intelligence and Simulation of Behaviour Quarterly*, vol. 122, pp. 7–9, 2005.

[8] E. M. Schooler and J. Touch, "Distributed music: A foray into networked performance," in *Proceedings of the International Network Music Festival*, Santa Monica, CA, 1993.

[9] N. Collins, A. McLean, J. Rohrhuber, and A. Ward, "Live Coding in Laptop Performance," *Organised Sound*, vol. 8, no. 3, pp. 321–330, 2003.

[10] G. E. Lewis, "Why do we want our computers to improvise?," in *The Oxford Handbook of Algorithmic Music*, A. McLean and R. T. Dean, Eds. New York: Oxford University Press, pp. 123–130, 2018.

[11] T. Magnusson, "Herding cats: Observing live coding in the wild," *Comp. Music Journal*, vol. 38, no. 1, pp. 8–16, 2014.

[12] A. Sorensen and A. R. Brown, "aa-cell in practice: an approach to musical live coding," in *Proceedings of the International Computer Music Conference*, Copenhagen, pp. 292–299, 2007.

[13] A. R. Brown and A. Sorensen, "Interacting with Generative Music through Live Coding," *Contemporary Music Review*, vol. 28, no. 1, pp. 17–29, 2009.

[14] A. Farnell, "Procedural Audio Theory and Practice," in *The Oxford Handbook of Interactive Audio*, K. Collins, B. Kapralos, and H. Tessler, Eds. Oxford: Oxford University Press, pp. 531–540, 2014.

[15] H. Abelson and G. J. Sussman, *Structure and Interpretation of Computer Programs*, 2nd Edition. Cambridge, MA: The MIT Press, 1996.

[16] J. Rohrhuber, A. de Campo, and R. Wieser, "Algorithms Today: Notes on language design for just in time programming," in *Proceedings of the International Computer Music Conference*, Barcelona, 2005.


 ISBN: 978-1-61208-707-8 

[17] N. Hoeberechts, J. Shamtz, and M. Katchabaw, "Delivering Interactive Experiences through the Emotional Adaptation of Automatically Composed Music," in *The Oxford Handbook of Interactive Audio*, K. Collins, B. Kapralos, and H. Tessler, Eds. Oxford: Oxford University Press, pp. 419–442, 2014.

[18] D. Hofstadter and E. Sanders, *Surfaces and Essences: Analogy as the fuel and fire of thinking*. New York: Basic Books, 2013.

[19] A. Sorensen and H. Gardner, "Cyber-physical programming with Impromptu," *ACM Sigplan Notices*, vol. 45, no. 10, pp. 822–834, 2010.

[20] A. Eigenfeldt, A. R. Brown, O. Bown, and T. Gifford, "Distributed Musical Decision-making in an Ensemble of Musebots: Dramatic Changes and Endings," in *Proceedings of the International Conference on Computational Creativity*, Atlanta, GA, pp. 88–95, 2017.

[21] P. Dahlstedt, "Action and Perception: Embodying Algorithms adn the Extended Mind," in *The Oxford Handbook of Algorithmic Music*, A. McLean and R. T. Dean, Eds. New York: Oxford Univerity Press, pp. 41–65, 2018.

[22] S. Emmerson, "'Losing Touch?': The human performer and electronics," in *Music, Electronic Media and Culture*, Hampshire, UK: Ashgate, pp. 194–216, 2000.

[23] P. Todd, "Simulating the Evolution of Musical Behaviour," in *The Origins of Music*, Cambridge, MA: The MIT Press, pp. 361–388, 2000.

[24] M. Leman, "Music, Gesture, and the Formation of Embodied Meaning," in *Musical Gestures: Sound, movement, and meaning*, R. I. Godøy and M. Leman, Eds. New York: Routledge, pp. 126–153, 2010.

[25] S. Larson, *Musical Forces: Motion, Metaphor and Meaning in Music*. Bloomington: Indiana University Press, 2012.

[26] A. R. Brown, et al., "Interacting with Musebots," in *Proceedings of New Interfaces for Musical Expression*, Blacksburg, VA, pp. 19–24, 2018.

[27] P. Auslander, *Liveness: Performance in a mediatized culture*. Oxon, UK: Routledge, 1999.

[28] M. Frengel, "Interactivity and Liveness in Electroacoustic Concert Music," in *The Oxford Handbook of Interactive Audio*, K. Collins, B. Kapralos, and H. Tessler, Eds. Oxford: Oxford University Press, 2014.

[29] H. R. Maturana and F. J. Varela, *The Tree of Knowledge: The biological roots of human understanding*. Boston: Shambhala, 1988.

 ISBN: 978-1-61208-707-8 