

Creative Computing and the Generative Artist

Bret Battey
Music Technology and Innovation Research Centre
De Montfort University
Clephan Building
Leicester, LE1 9BH, UK
E-mail: bbattey@dmu.ac.uk

Abstract: This article addresses the research agenda of creative computing from the perspective of a generative artist/composer, someone for whom processes of software creation and art creation inseparably intertwine. Using his audiovisual composition *Clonal Colonies* (2011) as a case study, the author addresses the dynamic of generative artistic creation when it is a process of discovery and dialog with artist-created, often unpredictable software systems. He provides technical specifics regarding his use of his Variable-Coupled Map Networks approach for music and his Brownian Doughnut Warper visual algorithm. Finally, he proposes a set of principles applicable to the creation of generative artwork, considers how tools and systems could better support such work, and proposes that creative computing research also focus on helping creatives surmount the fundamental personal challenges encountered in creative work of all types.

Keywords: creative computing; generative art; generative music; algorithmic art; algorithmic music; audiovisual art; visual music; creative processes; iterated maps; complex systems; emergent behaviour.

Biographical notes: Dr Bret Battey (b. 1967) creates electronic, acoustic, and multimedia concert works and installations. He has been a Fulbright Fellow to India and a MacDowell Colony Fellow, and he has received recognitions and prizes from Austria's *Prix Ars Electronica*, France's *Bourges Concours International de Musique Electroacoustique*, Spain's *Punto y Raya Festival* and *MuVi4*, *Abstracta Cinema* of Rome, *Amsterdam Film eXperience* and the *Texas Fresh Minds Festival* for his audiovisual compositions. He studied composition and electronic music at Oberlin Conservatory (BMus) and the University of Washington (MA, DMA Music) and is a Reader with the Music, Technology, and Innovation Research Centre at De Montfort University, Leicester, UK.

Error is drawing a straight line between anticipation of what should happen and what actually happens. (Cage 1961, pp.167-168)

We have come to value... responding to change over following a plan. [From the Agile Manifesto (Beck et al. 2001)]

Order and chaos, simplicity and complexity, the mechanical and the organic, aren't necessarily at opposite ends of a spectrum. They're symbiotic, intertwined. Any line we might walk between the two is a knife edge. Our very existence is poised between entropy and order... (Pearson 2011, p.xxviii)

1. The Artist Toolmaker

Despite the fact that history contains examples of artists spearheading technological innovations (Fishenden & Hugill 2014), a more common assumption is that engineers and technologists make tools that artists then use. A comment by composer John Adams reflects this view: 'Technology precedes artistic invention (as much as we artists would like to think it's the other way around!). First came the electric guitar and *then* came rock and roll' (cited in Cox & Warner 2005, p.111). This reflects an important partial truth that challenges some potentially misplaced artistic hubris, but even this particular example is not as clear as it might seem. Les Paul, one of the pioneers of the solid-body electric guitar, was himself both an electronics experimenter and a professional guitarist struggling with the challenge of making the guitar heard in ensemble performances (Lawrence 2008). The relationship between the electric guitar and rock and roll is not a linear one, either. It would be more accurate to say that they co-evolved, enabled by a feedback loop (sometimes quite literally!) in which artistic development and tool development enabled and guided each other. Certainly there were people involved who were primarily or purely technicians or artists, but hybrid artist-toolmakers like Les Paul arguably provided an essential locus for innovation, where one mind in active touch with both artistic and technical domains could envision, explore and make real new potentials.

This is an important consideration as we approach the question of creative computing for artists. First, we should recognize the distinction between creative computing that supports the work of individuals and that which supports the work of teams. One important focus for creative computing will be lowering the risks of creative impairment and blockages in cross-disciplinary group processes. But it will be equally important to recognise the crucial importance and distinct processes of individuals whose work crosses disciplinary boundaries, wielding a possibly unique capacity to synthesise new insights in way not possible for teams of specialists.

Second, habit may lead us to unconsciously approach the question of creative computing for artists with a broad assumption of a 'software as paintbrush' or 'software as instrument' model, where the active question will be on how software engineers should make better tools for artists, with an accordant temptation to focus on linear, waterfall models of development. The rise of approaches such as Agile Software Development or 'lean startups' (Ries 2011) reflects in part the reality that linear models are often not sufficient for creating successful software for even comparatively well-defined domains such as business processes or in the highly dynamic world of app and internet development. On the other hand, if co-evolution,

enabled by feedback, is an essential element of creative artistic and technical development — or even successful business software development — we might ask how we can strengthen and shorten the feedback loops involved.

Hugill and Yang, in their introductory essay for this journal (2013) go so far as to suggest ‘software applications that continuously rewrite themselves in real time in response to the creative needs of a particular problem or question will be the primary outputs of Creative Computing’. However, they also suggest that the creation of satisfactory creative tools in all spheres of human endeavour ‘requires... collaboration between creative people and software engineers.’ So their formulation to some degree takes this divide as fundamental. Such a divide, though it may prove necessary for the development of some facets of creative computing, may diminish and delay the co-evolution feedback loop in others.

So it is important to ask how creative computing can amplify the artistic potential of the single individual whose creative visions and activities bridge technological and artistic development. This includes artists who work with software development in particular. There is already a significant history of artists and composers for whom coding is an essential part of their creative processes (see for example Bohnacker et al. 2012, Chadabe 1997, Wilson 2003) — and who arguably provide a rich range of models of ‘achieving creativity through computation’, a phrase from Hugill and Yang’s core definition of creative computing. Given this history and rapidly expanding current practice, some salient focus questions for creative computing in the arts may be, ‘How can we better enable artists to make their own tools’, or even ‘How can we blur or even obliterate the line between tool making and art making?’ Two related questions are, ‘What, if anything, can only be achieved by an individual operating as both coder and artist?’ and ‘Are there forms of creative insight or artistic activity that will inevitably be blocked if creative-computing software is replacing low-level, or even mid-level, coding?’ Maybe, ultimately, we are *unavoidably* caught in the conflict between the flexibility — and corresponding barriers — provided by low-level coding and the constraints — and corresponding facilitating of pre-defined behaviours — provided by high-level software. Either way, it seems likely that for the foreseeable future there will be artists who see coding itself as an important creative pathway.

In that context, with this article I look at some of the technical specifics and artistic dynamics of my work as a generative artist¹ for my 2011 audiovisual composition *Clonal Colonies*². In part, my intent is to document some of my techniques for future elaboration by others. But I also seek to address creative computing by providing one perspective (of many possible) on how a creative process can unfold when coding is intrinsic to that process. I highlight gains and areas of dysfunctions of my own process and inquire how new directions in computing might amplify the gains and diminish the problems arising with this type of work.

2. Requirements / Motivation

Hugill and Yang propose initial *motivation* and *formulation* steps for the process of musical creation, comparing them the role of user requirements in traditional software development models. In that sense, the creation of *Clonal Colonies* was to a large degree motivated and circumscribed by a commission from New York’s Avian Orchestra ensemble for a botany-themed concert. This defined the core acoustic

instrumentation: flute, woodwind multi-instrumentalist, violin, cello, piano and percussion. This also meant that any techniques I used in the development of the piece needed to result in human-performable work, likely delivered via standard notation. Given the all-too-common realities of contemporary music making, I also needed to create work that could be successfully executed with quite limited rehearsal time (a constraint that, in the end, I arguably failed to meet). The final work eventually took form in two movements, “Fresh Runners” and “Soft Strata”. The acoustic instruments perform in sync with a fixed-media video containing a sound track comprised of computer-realised sound.

The botany theme encouraged me to use this as an opportunity to further develop my Variable-Coupled Map Networks (VCMN) approach for music composition (see below), since it was originally inspired in part by artificial life concepts and biological feedback and homeostasis. Since this decision would raise many technical and aesthetic challenges in itself, I constrained the scope of the work by engaging in only incremental expansion of my Brownian Doughnut Warper (BDW) visual filter (see below) for the visual component of the piece.

My work was also guided by broad (an engineer might say ‘ill defined’) interest in establishing ‘mysteriously coherent complexity’ — where the mind can perceive coherence and order amidst a dense and complex audiovisual texture, but isn’t necessarily sure why it seems coherent. Indeed, though I may be responsible for creating the software mechanisms that generate a visual or musical result, I may not fully understand how or why it leads to particular system behaviour, let alone the response it gives rise to *in perception*. I am thrilled — sometimes — when my software’s output exhibits an engaging complex order that I did not predict. For this reason, I often create software mechanisms that entail *complexity* and *emergence* — either in formal senses of those terms within complexity science, or in the practical sense that complex interactions between the elements generated by a system result in *effective complexity* in perception.

In discussing generative art, Philip Galantner, with reference to Gell-Mann and information theory, describes effective complexity as a balance point between high-redundancy (low information content) and randomness (high information content) (2003). He suggests that, ‘both highly ordered and highly disordered systems are simple. Complex systems exhibit a mix of order and disorder.’³

Emergence, though a common concept arising in discussions of generative art, is at least as problematic as the term complexity, ‘with a nexus of barely related meanings in different domains, making it a difficult term to clearly define, let alone understand’ (McCormack & Dorin 2001). But practically speaking, it points to the idea that we can attain relatively complex behavioural results from simple mechanisms, and be surprised by those results. This is a fundamental draw for many artists working directly with code. Surprise is an essential aspect of creativity (Boden 2003). For the generative artist, the surprises arising from emergence can inspire artistic outcomes that he or she may not have initially imagined. The technical ideas wielded or developed by the artist in the process of making the code need not necessarily be H-creative, or even fully competent from an engineering perspective, but the results and the emergent process have the potential to lead the artist in directions that may ultimately prove P-creative.⁴

When working with systems that give rise to emergent complexity, the artist often cannot work by imagining a specific outcome and commanding one’s tools to achieve it. Instead, the artist ‘dialogs’ with the system, finding what it can and cannot do. But this is precisely the domain where surprises arise: from what the artist-created

system does rather than elements arising directly out of the artist's own mind. The artist often imagines a technical approach, perhaps inspired by a sense it might provide certain types of results, and then discovers its aesthetic potential. Creativity is applied to finding ways to work with the system and being receptive to what arises even if the system's outputs do not fit one's original vision. The path is non-linear. Exploration and editing and continuous reassessment of goals, criteria and means — including making changes to the mechanisms themselves — will likely be part of the process.

To return to Hugill and Yang's proposed layers of activity in music creation, then, which also include *creation*, *dissemination*, and *revision*, it is crucial not to overlook their point that these layers *interact*. That is, there can be a great deal of feedback between all of these activities — very different from an ordered waterfall model from software engineering. Perhaps this is even more the case if coding emergent behaviours is part of the process, where possibilities and creative objectives likely only solidify through iterations of envisioning, attempting, surprise (and frustration), and re-envisioning of aesthetic and technical goals. The delight is that all of these aspects interact and can be sites for inspiration of new creative ideas in both the artistic and technological domains (see Figure 1).

<Insert Figure 1>

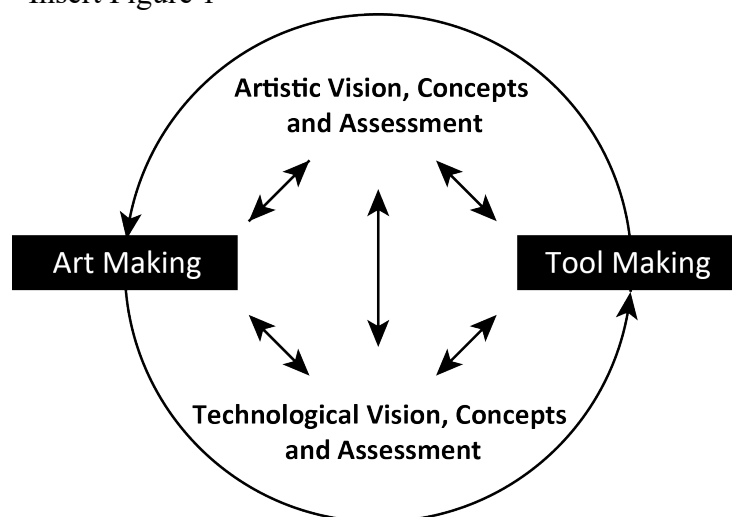


Figure 1: Potential process of the artist-coder. Artistic vision, concepts, assessment and acts interact with technological vision, concepts, assessment and acts, creating a dynamic system in which the final product(s) emerge from a process rather than a predefined set of 'specifications'.

With that in mind, we can now turn to the genesis, development and application of the two key generative systems I used in *Clonal Colonies*: VCMN and the Brownian Doughnut Warper.

3. Music with Variable-Coupled Map Networks

3.1. Overview

A Variable-Coupled Map Network (VCMN) consists of a set of interlinked nodes. These nodes are iterated maps — that is, mathematical functions that take their output and feed them back into an input. The output of any one of these nodes may set a function variable in itself or any of the other nodes. Each node has a wait-time between iterations, which can also be set or controlled by other nodes. I described the concept in detail in Battey (2004).

Most of my work with VCMNs has used Lehmer's Linear Congruence Formula (LLCF) as the iterated map:

$$x_{t+1} = (x_t a + b) \bmod m.$$

This formula is normally used as a pseudo-random number generator by optimising the variables for this purpose (Ames 1992). Professor Gary Nelson introduced LLCF to my computer music class when I was an undergraduate student at Oberlin College, demonstrating how deoptimising the variables caused LLCF to become an intriguing pattern generator. 'Breaking' a mathematical process or applying it to purposes for which it was not designed can be a useful creative strategy for the generative artist.

As a complex system, there is little that can be generalised about the behaviour of a deoptimised LLCF, and in most ranges small shifts of the variables will create a non-proportionate change in the behaviour of the output. This is all the more true when these maps are arranged in VCMNs, which may entail multiple levels of feedback.

I used LLCF and VCMNs to create sub-elements in a variety of compositions in the 1990s and 2000s. However, it was only with *Clonal Colonies* that I first set out to use VCMNs as a primary tool in developing a large-scale composition.

3.2. Goals/motivation

My goal was to use VCMNs to create a sense of continuous transformation of material. I had established such a sensibility in some previous pieces via manipulation of the parameters of one continuously running visual algorithm (see below) or a feedback-based audio-synthesis process (Battey 2011) for the duration of the piece. But this becomes a very different challenge when working with acoustic instruments and discrete notes.

I also wanted to ensure that there were clear, unifying behavioural or thematic elements audible to the listener. One principle of effective perceptual complexity is that change has most significance when it occurs to identifiable, coherent elements. I knew I would need to find ways to overcome the tendency of VCMNs to create *too much* novelty.

Finally, I also wanted to create a composition with clear dramatic shape and contrast. It is relatively easy to create dramatically flat or 'ambient' musical structures with algorithms; classic techniques tend to create patterns exhibiting continuous change, but not the coherent *change in the nature of change* necessary to create dramatic structures. I realised that I might find a way to configure a VCMN to do this, or I might need to intervene in a more direct fashion to shape the behaviour of the system. In this sense, I claim that a generative artist does not need to be a purist. The artist can do what is necessary to achieve results that he or she believes in, even this requires deviation from a conceptually elegant technical approach.

3.3. Implementation

For *Clonal Colonies*, I developed a more fully featured implementation of VCMNs in the Max programming language⁵ than had existed in my previous software sketches. This included a detailed and flexible user interface. I developed a time-quantization scheme, since I needed rhythmic results that could be notated and performed by musicians. I added dynamics (MIDI velocity) as a parameter for each node, as well as facilities for defining pitch modes/scales and refining the mapping of inputs to node controls. I also added randomisation of the network configuration as a means for (hopefully) discovering interesting behaviours.

Achieving consistent and reliable system behaviour required addressing numerous lower-level programming issues, which will not be discussed here. However, one example points to some risks for the artist-coder. I had recorded a great deal of VCMN output into a sequencer before discovering that code errors had generated numerous duplicate notes, which greatly complicated further editing. The artist coder may not need to engage in thorough unit testing, but he or she does have to exercise good judgement regarding when and how to test foundational code, particularly when later production steps have very specific input requirements.

3.4. Configuration

I chose to create a network with six nodes, one for each instrument. This was the simplest network conceptually with which to write for the ensemble. Even then, the range of potential configurations and control manipulations was vast — too vast. As I noted in my composition journal, ‘Lost again. Too many options, too many having only “OK” results’. I spent quite a bit of time trying various configurations of the network and nodes and listening to the results without feeling fully satisfied. This, too, points to a risk for the generative artist: since one does not directly do the hard work of generating the events themselves, it can be very easy to spend a lot of time sitting in fascination (or consternation) at the behaviour of the system rather than doing the hard work of making the artistic and technical explorations and decisions necessary to move the work forward. Sometimes such decisions feel essentially arbitrary, but that may be what is needed to narrow and focus the field of possibilities.

The VCMN configuration that I ultimately settled on for the first movement was a fully driven system, meaning that all node inputs were controlled by another node. I privileged the violin node, at least conceptually if not perceptibly, as the primary driver of the network behaviour. The output of the violin node controls all of the node inputs, including its own, except the violin *b*-variable and flute *b*-variable and velocity (controlled by the bass clarinet), the bass clarinet velocity (controlled by the flute), the piano *a*-variable and velocity (controlled by the gongs), and the gong *a*-variable and velocity (controlled by the piano). Notice the symmetry of control between the two percussion elements (piano and gongs). The scale was almost always a synthetic, symmetrical scale comprised of alternating one- and three-semitone leaps: C Db E F G# A. This creates a relatively ‘floating’ quality to the scale via the symmetry and the lack of a fifth scale step and leading tone to the tonic.

For the second movement the configuration was also fully driven, and the piano node was given emphasis in the control scheme. The two stringed instruments set each other’s *b*-variable, rhythm and duration. The two woodwinds were similarly paired. The piano node set the *a*-variable of all instruments. Its own *a*-variable was an average of the output of all of the other nodes, restraining the range of change it would undergo. The scale uses the notes of the Hindustani *raga* Yaman (major scale with sharp fourth step).

To help break through the paralysis of too many options, I used a common composer's strategy: improvisation. I wrote code to decode the MIDI data of a Roland PG300 synthesizer programmer interface, which provided a variety of continuous and discrete-value sliders and switches of different ranges. Thus it helpfully both suggested and limited what I could control with it. I connected the controllers to provide on/off controls for each instrument, *a*-variable and *b*-variable settings for each node (potentially overriding network connections setting those values), choice of musical mode/scale, minimum and maximum index into that mode, minimum and maximum rhythmic values, and minimum and maximum velocity ranges for each instrument. This made it far easier to explore the behavioural range of the system until I could gradually sketch a structure for improvisation that would provide a convincing dramatic shape. Once I had developed that structure, I started capturing takes to a MIDI sequencer.

As a creative constraint, I decided I would allow myself little or no editing of the timings or pitches after capturing a take. As I put it in my journal, 'Current inclination is to leave [time] space in the acoustic parts and add computer parts by hand to provide rhythmic cues, bass and canopy, gestural reinforcement, and spectral fusing.' In other words, I had to make the captured improvisation work by how I framed it. This would hopefully encourage aesthetically fresh results as I tried to find convincing solutions within the constraint.

3.5. Post-processing algorithms

One challenge of using VCMN to generate material for acoustic instruments is the fact that, in its base configuration, each node will continuously generate notes rather than break material into phrases. Besides the high potential for monotony, this can be a problem for wind instrument players, who need pauses in which to breathe. I used post-processing algorithms to address this issue and also generate additional unifying behaviours.

The *raga* system of Hindustani classical music includes the concept of particular notes in a scale that are given emphasis to achieve the emotional colour of a particular *raga*. These are referred to as the *vadi* (most significant) and *samvadi* (next most significant). This emphasis may be achieved with a variety of means, including relatively frequent restatement, common usage at the beginning and ending of phrases, longer durations, or special elaborations (Bagchee 1998, pp.44-45). For the first movement, each time the VCMN generated a note for an instrument, post-processing code tested to see if the note was an E. If so, a trill was generated for that note and that instrument's node was put on pause for a certain number of beats. I set the length of this pause with a slider on the control box. In this way the VCMN patterns were broken up into phrases, all of which ended with the same trilled pitch. This provided a strong behavioural identity that unites the entire movement even as the piece moves through a variety of strongly contrasting materials. The density of the whole musical texture was controllable in part by the pause-duration control.

In the second movement, pauses (but no trills) occurred upon arrival at a major third or seventh. Articulations were legato except when notes repeat, at which point the articulation turns to staccato. I achieved chords for the piano and gongs by storing the last three generated notes. An interface slider position determined which of those three stored notes (if any) were used to double the current note of the gong or form a chord with the current note of the piano.

3.6. Computer-rendered sound elements

I used the language SuperCollider⁶ to create most of the computer-rendered sound elements. This could be discussed at length itself, but the core techniques — convolution, granulation, pitch-shifting, time-stretching, and resonator models — are well established in the field. Exceptions include the use of non-quantized VCMN to create percussive gestures in movement I. These demonstrate the surprising capacity of VCMN to create coherent rhythmic gestures in a completely free rhythmic space.

3.7. Capture, edit and notate

A surprising amount of my composition journal records struggles to find and implement a viable technical path to integrate all of these elements. Gradually a solution evolved out of numerous trials and dead-ends. Ultimately, I routed MIDI data from the VCMN implementation in Max to the Digital Performer (DP) digital audio workstation software, with a separate track for each instrument. These tracks were routed to a Vienna Symphonic Library (VSL) plugin to provide acoustic-instrument emulation for each part. A Max hostsync~ object allowed the DP transport controls to drive the Max transport, ensuring timing sync between DP and the VCMN. I recorded multiple takes into DP. I then selected material from the best takes and edited the MIDI data. I added and edited the computer-realised sound elements within DP.

But transforming the final sequence data to notation raised another challenge, particularly given the large amount of detailed instrument-articulation data involved — triggered by special keyswitch MIDI notes sent to VSL — and often note-by-note changes of dynamics. I exported the instrumental tracks to Standard MIDI files. Using Rick Taube's GRACE⁷, a LISP/SCHEME-based algorithmic-music language, combined with David Psenicka's FOMUS⁸, a LISP-based library for music notation, I created a utility, VSLCONV⁹, to convert the MIDI files to MusicXML. This included appropriate indications of articulations and dynamics. The music-notation program Finale converted the MusicXML to music notation. Significant editing was still needed. From this point, any further edits to the music had to be manually executed in both DP and the notation.

Between both failed and successful attempts to find this technical path, it appears that I used at least three different commercial software packages, four different programming languages, four different data or communications protocols, and two different notation-oriented code libraries — including custom code to bridge to those libraries. Little of this effort provided rich potential for new creative insights. Too much of this kind of challenge risks bringing the artistic process to a standstill.

4. Image: Brownian Doughnut Warper Visual Algorithm

4.1. Origins: The Creative Value of Coding Errors

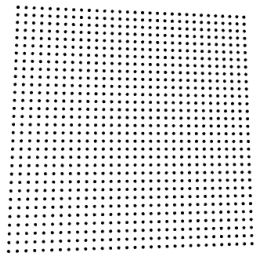

Creative artists think in a disciplined manner: they may be playful, but they aren't merely playing around. When something of potential interest turns up as a result of their playfulness, the focus on it — accepting, amending and developing it in disciplined ways. (Boden 2003, p.319)

Software engineering is focused in part on the avoidance and elimination of errors. But for the generative artist, technical errors are a source of the unexpected, and

therefore they can inspire new creative directions. Indeed, entire musical genres have developed around technological ‘glitches’ (Cascone 2000).

While teaching a class in generative design as a Research Fellow at the University of Washington, I developed a class example to demonstrate how to use the X and Y position of the mouse to control a 3D rotation of a grid around the X and Y axes, using the Java-based Processing language¹⁰. I established a nested loop to draw a grid of dots on the screen row-by-row and column-by-column. Having not fully aligned my thinking with fact that OpenGL works as a state machine, I then placed the 3D-rotation commands inside of the two nested loops rather than outside the loops. As a result, the rotation angles accumulated for each point in the grid, twisting each row into 3D spirals. Further, OpenGL points by default do not change their size based on distance, so the result appears as if it were on a flat plane despite the fact that displacements are actually in 3D space (see Figures 2a and 2b). Moving the mouse created a wide variety of patterns emerging from the interaction of the X and Y rotations. I was fascinated. This one error launched the trajectory of my next decade of algorithmic image making.

<Insert Figures 2a and 2b>

	
<p>Figure 2a: Intended rotation of a grid five degrees on both the x and y axis</p>	<p>Figure 2b: Result when the rotation command is incorrectly placed inside the nested loop that draws the grid</p>

4.2. cMatrix10 (2004)

I explored the potential of the algorithm through a series of Processing sketches. Ideas explored included slow, continuous increment of the angle by code rather than by mouse in order to provide slow transformation of the resulting patterns, an early decision to make the y-rotation angle always be twice the x-rotation angle (another case of an arbitrary decision moving the work forward), and providing colour variation by loading a digital image and displaying (and warping) it via the grid algorithm. Given that the points were not scaled by depth, I was free to scale them by some other basis to provide variation and interest. I gradually developed an approach in which point size s in pixels was determined by

$$s = 1 + \frac{w - x}{w} + \frac{y}{h}$$

where w is the width of the (non-warped) grid, h is its height, and x and y indicate the point’s width and height location within the grid. This meant that most of the slowest, inward points were larger than the outlying, fast-moving points. By sketch 10, I had conceived of a contemplative, looping installation artwork that fades in from a white screen, starting with tiny points in highly randomized motion, gradually solidifying

and congealing to reveal the rotational patterns. The process reversed at the end of the loop. Time-echo and blur processing in Adobe AfterEffects thickened the texture and provided more subtlety in colour gradations. The result, once sound was added, was the artwork *cMatrix10*¹¹.

The element of randomization here is in some sense the most readily controlled aspect of the system. When the same bounds or random behaviour apply to a vast mass of points, this is actually a consistency, rather than a deviance. It can be likened to the behaviour of a gas: we don't know the state of each individual molecule but can predict the behaviour of the whole with high precision. The gradual scaling of Brownian noise I implemented for the randomisation provided a fascinating transition band between randomization and perceptible order, and this in itself seemed worthy of further investigation.

4.3. Autarkeia Aggregatum (2005)

Another tool used by many artists to generate new ideas or release ideas from the unconscious is stream-of-consciousness writing, an approach I have used on and off again since the mid-90s. In a May 30, 2005 note I wrote: 'Realized today could be nice to have the *cMatrix10* also as a plugin. Dissolve an image into Brownian or *cMatrix10* rotations or both, then re-coalesce into a new image. Continuity of transformation. This is the magic.' Those ideas, which guided much subsequent work, unassumingly appear in the midst of pages and pages of seemingly mindless meanderings, dead ends, numerous other ideas never pursued, expressions of frustration, and mundane notes about how just get a certain idea to work technically at all. Sometimes giving mundane concerns an outlet, such as through stream of consciousness writing, provides room for deeper insights to arise. Even then, the significance of an insight is not always apparent at the time it occurs.

I began to conceive of a new work that would continue to have one thing continually transforming into the next, without cuts or edits of video material, but with greater variety and more intentional dramatic shaping than *cMatrix10*. This would require detailed control over the input image and of the parameters of the algorithm. While coding environments like Processing or Max invite exploratory approaches and iterative development of ideas, they often do not provide the refined control interfaces found in production-oriented software packages. To solve this, I shifted to writing custom plugins for Apple's Motion video effects software, using the FxPlug Objective-C API. Thus I could use Motion's interface to control algorithm parameters, stage and transform the input images, easily provide video (including time-stretching and filtering) as an input to the plugin, and render output directly to Quicktime video.

Still, production tools like Motion come with certain ground rules of how plugins are expected to behave. Typically, one should be able to request a render of any single arbitrary video frame at any time. But processes that involve feedback over time, such as a technique I was using of alpha blending a new video frame over the previous frame to get motion trails, violate this model. This would be a problem if one were creating software for others to use. But the generative artist can feel free to violate the host software's expectations — as long as he or she is willing to deal with the result (namely, in this case, that the only way to ultimately see what a given frame is like is to start rendering well prior to that frame).

Since the time cost of creating a Motion plugin for the first time was high, I constrained what I implemented within it. I focused only on refining the Brownian displacement and leaving out the 3D rotations. Displacement was now defined by a

polar magnitude and angle. I could set the magnitude and angle increment for each video frame, providing circular motion. Brownian motion could randomize magnitude and angle to varying degrees. I used this plugin to create *Autarkeia Aggregatum* (2005)¹². Artistic perspectives on this piece can be found in Battey and Fischmann (publication pending).

4.4. Luna Series (2007-9)

Over the course of three works entitled the *Luna Series* (*Mercurius* (2007), *Lacus Temporis* (2008), and *Sinus Aestum* (2009))¹³, I reintegrated and refined 3D-rotation schemes inspired by *cMatrix10*, making what I called the Brownian Doughnut Warper (BDW) filter. The ‘doughnut’ refers to an enhancement of the polar Brownian displacement that allowed me to give both an inner radius and outer radius to the magnitude displacement. Point sizes could be scaled by the brightness of the point or by the degree of magnitude displacement. I added Z-axis rotations, as well an overall scalar that could be applied to the angle displacement – allowing a smooth transition from a non-displaced grid to a full 3D-rotation displacement. Rotations of the three axes could be linked to each other, either by adding the angle of one of the other axes, with variable scaling applied, or the square of one of the other axes, again scalable. By the end of the process of creating the *Luna Series*, BDW had 30 parameters. By adding only a few elements with each composition and refining and editing them in the process of making that composition, I was able to explore the potential of BDW in depth, solidify the concepts, and demonstrate the core algorithm’s capacity to provide a wide variety of expressive characteristics. A video is available online that describes and demonstrates the Brownian Doughnut and Warper mechanisms as they functioned by 2009¹⁴.

4.5. Clonal Colonies (2011)

The plugin I developed in the process of making *Clonal Colonies*, BDWv4, included more incremental additions, each of which opened up significant new territory for exploration. By the completion of the piece, the plugin had 58 different parameters. The addition of two more plugins and a moveable camera (see below) provided even more options and points of control. The fact that I had gradually developed the tool to this state over years is what kept this range of options from being creatively overwhelming. Some of the more notable additions are described below.

4.5.1. Movable camera

For the first time, the camera became mobile in 3D space. This added tremendous artistic flexibility. But camera position, orientation, motion, and focal length are complex issues to address in terms of both code and user-interface design. Therefore, I used Motion’s own 3D-camera controls, passing the transformation matrix into the plugin — even though this isn’t really how Motion expects the 3D camera controls to be used (once again, breaking the normal usage paradigm for the host tool). This turned out to be a teeth-gnashingly frustrating and difficult thing to implement given my thin linear-algebra skills and limited understanding of OpenGL camera and view logic. The solution ultimately required significant help and custom code provided by Apple engineers who support FxPlug developers. Even an obstinately independent generative artist needs to know when to get help from others.

4.5.2. Bézier splines between points

With this version, BDW evolved to include lines. Let us describe the BDW point grid in the form

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

Recall that the actual position of each point is determined first by grid warping and then by polar displacement. For each point p ($1 \leq p < m$) in each column c ($1 \leq c \leq n$), a 3rd-order Bézier spline (Vince 2006) is cast between a_{pc} and $a_{(p+1)c}$ with a global user-definable thickness. The position of the Bézier control point b associated with any end point is determined by a tangent pointing from the polar-displaced point position towards the original, non-polar-displaced location. The length of the tangent is determined by multiplying the displacement distance d by a user-controllable scalar l ($-100 \leq l \leq 100$). Thus the final disposition of the spline is determined both by the polar displacement and the user's specification of the tangent length, providing a highly flexible and expressive mechanism (see Figure 3).

<<insert Figure 3>>

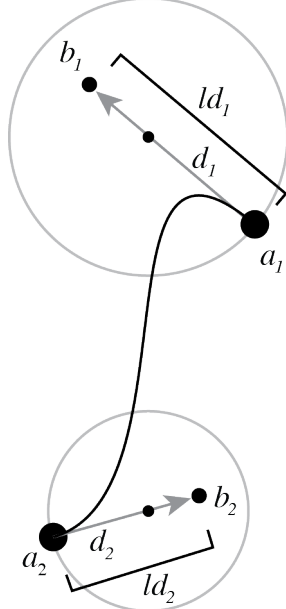


Figure 3: Determining control points b_1 and b_2 for a 3rd-order Bézier spline between two displaced points a_1 and a_2 . A vector d between displaced point a and its non-displaced origin is scaled by l (1.5 in this example) to determine the control point position.

Often in generative work, a seemingly simple idea proves much more complex to implement than the artist first assumes. In order to draw these lines between points, the position of each point in the world-coordinate system had to be identified. This was achieved by pushing the world coordinates onto the OpenGL stack prior to executing the needed world rotations for a given point, grabbing the resulting modelview matrix, multiplying the base and displaced point locations by that matrix, and then popping back to the world-coordinate system to begin drawing.

4.5.3. ComboSoft Filter

Wanting to achieve more subtle gradation and variety of colour, I re-integrated another idea from *cMatrix10*: using blurring and compositing to create subtle colour gradations and softer edges for objects. I created a new Motion plugin, ComboSoft, which takes the input OpenGL texture and uses Apple CoreImage to apply a Gaussian blur with user-definable radius. A parallel process applies user-definable saturation, brightness and contrast adjustments to the input image. This colour-adjusted image is then add-composited on the blurred image.

4.5.4. Trailz

A new Trailz plugin provided the motion blur effect built in to previous plugins — achieved with retention of the previous frame and alpha blending over it. Separating this effect into its own plugin allowed other post processing, such as the ComboSoft plugin, to operate after BDF and be included in the feedback loop. Again, this seemingly simple idea ended up requiring a time-consuming process: this time to learn how to generate and utilise OpenGL pixel buffers in an FxPlug. However, the conceptually simple combination of ComboSoft plus Trails provided, though the feedback mechanism, a highly expressive and flexible — and sometimes delightfully unpredictable — set of creative options.

5. Audiovisual Relationships: Fluid Audiovisual Counterpoint

Despite using algorithmic methods heavily in both the musical and visual elements of *Clonal Colonies*, I linked image to music almost entirely by hand through detailed keyframing of the parameters for all five Motion layers (input image, BDW, ComboSoft, Trailz and Motion Camera).

My choice in this regard is salient to the topic of creative computing. Music and moving image individually are often complex, multidimensional perceptual gestalts, where the gap between the quantitative (what we can measure) and the qualitative (what we experience) is vast. Western classical music, for example, can exhibit an extraordinarily complex interweaving of linear (melodic) and horizontal (harmonic) relationships, constraints and structures at multiple temporal levels with weighted stratifications (forms, modes and meters) — interacting with genetically and culturally defined psychoacoustic mechanisms and expectations. Instantaneous linear-reductionist measures applied to music, such as the commonly used amplitude measurement of a frequency band, will tell us very little about this type of perceptual gestalt. If an artist maps that reductionist measure to another isolated parameter in an image-production process, it should not be surprising if the result provides a shallow audiovisual relationship — at best.

In many situations, algorithmic linkage of music to image in a way that does reflect the full perceptual dimensionality of both mediums would likely be a question of artificial intelligence and/or artificial perception rather than creative computing. For now, deep audiovisual linkage usually requires manual crafting by an artist who is sensitive to the multidimensional complexity of both music and image gestalts — and to the new, truly audiovisual perceptions that arise when they are linked. In other words, I am manipulating the parameters — and sometimes the coding — of independent audio and visual processes, each of which alone gives rise to emergent

behaviour and perception, seeking through patient trial and error to achieve a satisfactory co-emergence as these two sensory streams combine.¹⁵

Given the complexity of the challenge, in my audiovisual works prior to *Clonal Colonies* I focused primarily on establishing long-term isomorphic relationships between sound and image gestalts. With *Clonal Colonies* I sought to push my practice in the direction of a ‘fluid audiovisual counterpoint’. This entails greater ebb and flow between alignment and non-alignment of tensions between articulation points in textures and gestures of each medium (Battey 2015).

So the two-year process of creating *Clonal Colonies* brought together a variety of techniques and aesthetic impulses, incrementally developed over several years and pieces, into what hopefully is finally perceived as a seamless audiovisual whole.

6. Conclusion: Creative Computing and the Generative Artist

If the domain of creative computing includes using computation creatively, then the work of generative artists can clearly be included under its aegis. As such, a legitimate task for creative-computing research is to seek principles underlying or facilitating the work of generative artists. We can also ask what types of software systems or characteristics might better facilitate the creative processes of these artists.

6.1. Principles

Reflecting on the above, I offer the following summary of some explicit or implied principles for the generative artist that arise from my own work:

- Errors in coding can be a catalyst for new creative directions.
- Artists can use code as a disciplined means to relinquish control to generate new ideas.
- Creating systems that entail some degree of unpredictability can be helpful, if one is willing to engage in the necessarily dialog with the system to discover its capabilities and limits.
- Perceptual complexity often lies between the extremes of redundancy and randomness.
- Deciding how to explore the behavioural potentials of a generative system is itself an art. Setting arbitrary constraints is often a necessary part of this art.
- When the generative system offers too many options, consider using improvisation.
- The refined mechanisms of production software can be very usefully combined with custom coded tools. The generative artist can choose to break the host software’s paradigms when doing so.
- The generative artist needs to acknowledge when he or she is in over her head and requires the help of others — or just needs to try a different path.
- Breaking, mistuning or misapplying algorithms can be a powerful creative approach.
- The generative artist must balance the love of creating new systems and pursuing new ideas with the drive and focus needed to create finished works (however one defined ‘finished’).

- Sometimes it is useful to focus on just the smallest incremental changes in a system that can open up new expressive potentials.
- The generative artist may be sorely tempted to justify his or her work in terms of the design of the code or its technical concept rather than the artistic result.
- Every programming language encourages certain kinds of behaviour and discourages others.
- The generative artist would do well to be aware that generalised code is more time-consuming to create than code ‘hardwired’ to achieve a specific purpose.
- Through-test complex production paths early.
- Technical ideas are almost always more difficult to implement than the eager generative artist thinks they will be.
- There is no replacement for the hard work of applying sensitive and patient human sensory perception and critique translated into iterative refinement.
- Overcoming the over-perfection of computation is an important consideration for the generative artist. It usually takes conscious work to create excellent imperfections that will optimise the interest of human perception.

6.2. Creative-Computing Support for Generative Production

Is it possible for a creative-computing meta-tool to enable some of the gains of generative computing while reducing some of its aspects that can inhibit creative flow?

My own work as a generative artist is characterised by attributes of creative computing such ‘endlessly fluctuating mix of divergent and convergent thinking’, and numerous moves back and forth between activities of ‘motivation, ideation, implementation and operation’ (Hugill & Yang 2013). In that case, any step in a production process that is a non-reversible transformation inhibits this flow. The capture of VCMN output in a sequencer, then editing, then transferring to notation provides an example, because I could no longer move back to the generative roots of the materials. Ideal tools and processes would create a high level of reversible transparency between such steps.

Certainly any coding process has some potential to give rise to a creative insight. But some coding activities are more likely to do so than others. My *Clonal Colonies* journal documents far, far more technical problem solving than artistic problem solving and visioning. When a generative artist has to expend extended effort and time on issues such as figuring out how to execute an idea using a poorly documented API, or coercing data types to pass information between disparate protocols and applications — they are distracted from the creative process and flow is interrupted. An ideal generative-arts meta-tool might be able to deal with such low- and mid-level issues that would otherwise add friction to the creative process, while still allowing artistically focused coding by the user.

On the other hand, we might imagine a generative-arts meta-tool that is an assistant to whom an artist can make requests in natural language, where coding is no longer necessary to work with generative means: ‘Let’s draw a grid of points starting from the upper left-hand corner, which can be rotated on the x and y axis using the mouse, and where I can control the grid size, spacing and rotation’. In theory, though, this could remove or reduce the possibility of fortunate errors. Fortunately, language parsing technologies and the ambiguities of mapping language to function might in

themselves result in interesting errors. But the meta-tool could also be designed to offer errors or alternative interpretations: ‘Here is your system. Would you like to see it with rotations applied for *each* point? With the nested loops reversed? With mouse motion inverse mapped? With the shape extended to a cube? Extended to a projected hypercube? Would you like to see other artists’ programs that involve similar systems? Would you like to search a cultural-symbol free-association space linked to the concept of rotating grids? Would you like access to a set of resources about constructivist art?’

6.3. Supporting the Human Dimension

<insert figure four>

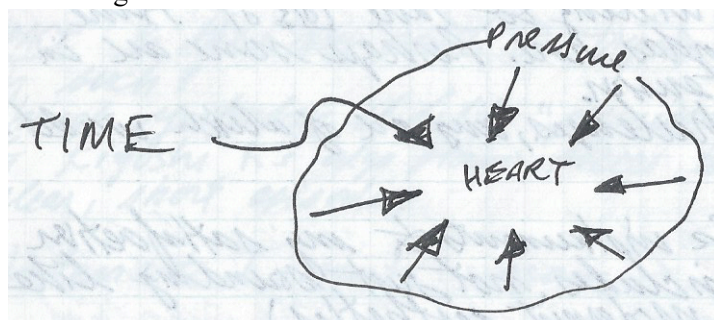


Figure 4: time pressure impinging on the heart from all directions (*Clonal Colonies* composition journal, November 2, 2009)

Over the last few days feeling uncomfortable with [the second movement]. It risks triviality and shallowness. Going back and listening to previous improvisations... (*Clonal Colonies* composition journal, July 21, 2011)

It is tempting to focus just on the technical needs of generative artists when considering the potentials of creative computing, or on specific mechanics of the creative process for creatives in general. However, for all types of creative workers, the greatest challenges are not technical, but personal.

To appropriately address the issue of creative computing, it is important to emphasise that the descriptions about my process and technique given above in sections 3-5 cannot sufficiently communicate the vast amount of nonlinearity, uncertainty, goal renegotiation, blind alleys, tiny increments of failure and success, discarded work, and overall groping exploration and problem solving in aesthetics and technology involved. At times, psychological stamina and self-belief, not new ideas, are the elements that seem at most risk of running dry.

Philosopher Robert Grudin proposes a set of common characteristics that comprise an ‘ethos of inspiration’ amongst people who exhibit high levels of innovative thinking. These include a passion for work (where the work/leisure dichotomy may disappear altogether), fidelity (on-going, long-term commitment to major challenges), love of the problematic (‘a chronic attraction to things that do not totally fit, agree, or make sense’), love of beauty, a sense of wholeness (equipping us for awareness of anomalies), boldness, consequence (‘seeing every major juncture in a given study as part of a process rather than a thing in itself’), innocence and playfulness, courtesy (suspension of ego), suffering (enduring the short-term pains and uncertainties inherent in the work), remembrance, a sense of continuity of perception, openness, and liberty. He also notes, ‘though creative insight may be delightful in itself, it normally is predicated on training, prolonged concentration, and exhausting practice that are not pleasant in the same sense’ (Grudin 1990, p.9).

In many ways, this is a significantly different list than we might find in an analysis of the mechanisms of creativity such as Boden's (2003). This is because Grudin is addressing the phenomenological reality of the creative process rather than looking for principles that might be used to guide the design of computational creativity. But perhaps it is precisely in this domain that creative computing could have its largest impact: helping human beings navigate with greater grace, efficacy and flow the lived dynamics of creative work. When we are being too safe, can a creative-computing tool encourage us to be bold? When we have been obstinately pushing against one wall for too long, could software encourage us to step back and reconsider? When we are being too self critical, can software point us to activities that will reconnect us with our sense of liberty? When we need to attain a deeply concentrated state in order to achieve creative insight, can software guide us? When we reach the limits of our stamina, can a tool help us 'gamify' our challenges or remind us of our past successes?

Such software would likely require computational creativity, but would not use it to replace or replicate human creativity. Instead, it would help human creators strengthen their individual 'ethos of inspiration' and sustain the fortitude needed to navigate the unknown — so that, as Rilke wrote in his *Book of Hours*, what we do may flow from us 'like a river / no forcing and no holding back / the way it is with children' (1996).

7. References

- Ames, C., 1992. A Catalog of Sequence Generators: Accounting for Proximity, Pattern, Exclusion, Balance and/or Randomness. *Leonardo Music Journal*, 2(1), pp.55–72.
- Bagchee, S., 1998. *Nād: Understanding Rāga Music*, Mumbai: Eshwar.
- Bathey, B., 2004. Musical pattern generation with variable-coupled iterated map networks. *Organised Sound*, 9(02), pp.137–150. Available at: http://www.journals.cambridge.org/abstract_S1355771804000226.
- Bathey, B., 2011. Sound Synthesis and Composition with Compression-Controlled Feedback. In *Proceedings of the International Computer Music Conference*. Huddersfield, UK.
- Bathey, B., 2015. Towards a Fluid Audiovisual Counterpoint. *Sonic Ideas*, 7(14), pp.26–32. Available at: <http://www.sonicideas.org>.
- Bathey, B. and Fischman, R. (publication pending) Convergence of Time and Space: Visual Music from an Electroacoustic Music Perspective. In Kaduri, Y., Ed., *The Oxford Handbook of Music, Sound, and Image in the Fine Arts*, Oxford: Oxford University Press.
- Beck, K. et al., 2001. The Agile Manifesto. Available at: <http://www.agilealliance.org/the-alliance/the-agile-manifesto/> [Accessed November 27, 2015].
- Boden, M., 2003. *The Creative Mind: Myths and Mechanisms*, London: Routledge.
- Bohnacker, H., Groß, B. & Laub, J., 2012. *Generative Design: Visualize, Program, and Create with Processing* C. Lazzeroni, ed., New York: Princeton Architectural Press.
- Cage, J., 1961. *Silence: Lectures and Writings by John Cage*, Middletown, CT: Wesleyan University Press.

- Cascone, K., 2000. The aesthetics of failure: 'Post-digital' tendencies in contemporary computer music. *Computer Music Journal*, 24(4), pp.12–18.
- Chadabe, J., 1997. *Electric Sound: The Past and Promise of Electronic Music*, Upper Saddle River, New Jersey: Prentice Hall.
- Cox, C. & Warner, D. eds., 2005. *Audio Culture: Readings in Modern Music*, London: Continuum International.
- Fishenden, J. & Hugill, A., 2014. *The Creativity-Innovation Elapse (CIE): Creative Innovation, Technological Acceleration and Economic Growth The Technology Adoption Lifecycle*, Available at: <https://www.bathspa.ac.uk/Media/Centre for Creative Computing/The Innovation Elapse - Creative Innovation and Acceleration. FishendenHugill-2.pdf> [Accessed November 27, 2015].
- Galanter, P., 2003. What is Generative Art? Complexity theory as a context for art theory. In *GA2003–6th Generative Art Conference*.
- Grudin, R., 1990. *The Grace of Great Things: Creativity and Innovation*, Boston, MA: Ticknor and Fields.
- Hugill, A. & Yang, H., 2013. The Creative Turn: New Challenges for Computing. *International Journal of Creative Computing*, 1(1), pp.4–19.
- Lawrence, R., 2008. *The Early Years of the Les Paul Legacy: 1915-1963*, Milwaukee, WI: Hal Leonard.
- McCormack, J. & Dorin, A., 2001. Art, Emergence, and the Computational Sublime. In *Second International Conference on Generative Systems in the Electronic Arts*. pp. 67–81.
- Pearson, M., 2011. *Generative Art: A Practical Guide Using Processing*, Shelter Island, NY: Manning.
- Ries, E., 2011. *The Lean Startup*, London: Penguin.
- Rilke, R., 1996. *Rilke's Book of Hours* Translated by A. Barrows & J. Macy. New York: Riverhead.
- Subyen, P., 2015. *Mapping, Meaning, and Motion: An Artistic Framework for Visualizing Movement Quality*. PhD Thesis, Simon Fraser University.
- Vince, J., 2006. *Mathematics for Computer Graphics* 2nd Edition, London: Springer.
- Wilson, S., 2003. *Information Arts: Intersections of Art, Science and Technology*, Cambridge, MA: MIT Press.

8. Notes

¹ The term 'generative art' is subject to numerous definitions. In this paper, it can simply be considered another term for art (visual, music or other) that involves an artist coding and manipulating algorithms as part of his or her process.

² Clonal Colonies is available online at <http://BatHatMedia.com/Gallery/clonal.html>

³ To be strictly true, this must be a statement regarding the behaviour/output of a system, not a statement about the system design itself. Indeed very simple systems can create statistical randomness; very elaborate processes can exhibit repetition or stasis.

⁴ Margaret Boden makes a distinction between P-creativity, which is novel to the individual mind, and H-creativity, which is novel with respect to human history. (Boden 2003)

⁵ <http://cycling74.com/>

⁶ <http://supercollider.github.io/>

⁷ <http://commonmusic.sourceforge.net/>

⁸ <http://fomus.sourceforge.net/>

⁹ <http://BatHatMedia.com/Software/index.html>

¹⁰ <http://processing.org>

¹¹ <http://BatHatMedia.com/Gallery/cmatrix10.html>

¹² <http://BatHatMedia.com/Gallery/autark.html>

¹³ These three works are available at <http://BatHatMedia.com/Gallery/>

¹⁴ <https://vimeo.com/14957896>

¹⁵ In this light, and challenging the absolutism of my statement, it is interesting to note the growing experimentation in recent years of using elements of Laban Movement Analysis as a qualitative intermediary between objective motion tracking and generative image and sound in interactive dance. See for example Subyen (2015). However, it is also notable that the phenomenological sense of the body is the foundation of the technique and most implementations require AI learning to link motion data to LMA movement qualities.