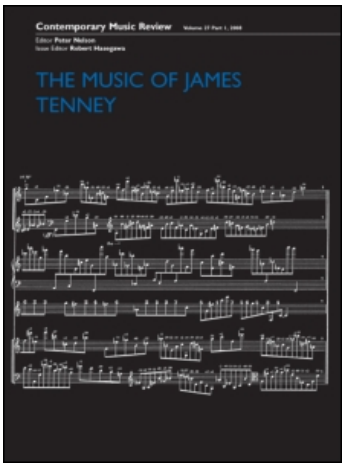

Contemporary Music Review
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713455393

Artists' Statements

Online Publication Date: 01 February 2009

To cite this Article (2009)'Artists' Statements',Contemporary Music Review,28:1,115 — 128
To link to this Article: DOI: 10.1080/07494460802664080
URL: http://dx.doi.org/10.1080/07494460802664080

Contemporary Music Review
Vol. 28, No. 1, February 2009, pp. 115–128

Artists' Statements

Heinrich Taube

There are a number of motivations that lead me to work with generative musical processes. On a philosophical level, I believe artists should always work at the limits of what is generally accepted to be 'artistic expression'. Everyone agrees that the music a composer creates can be called art, but in generative composition composers do not create music, they create processes that create music. Since algorithms allow the composition of compositions, generative composition represents a deeper, more radical degree of separation of the composer from the performance than the traditional compositional process. Are the algorithmic processes the composer creates art? Is music that an algorithm generates art? This strikes me as a rich frontier for artists to explore and one that is only beginning to realize its potential significance.

There are also aspects of generative composition that make it an interesting way to work on a more practical, methodological level for me as well. All composers have specific techniques and formalisms that they work with in their music. In generative composition, composers express their methodology in a literal, reusable way (as 'code') that can be altered and reused in many different compositional projects. Even within a single piece the identical algorithmic process can result in completely different musical consequences simply by altering its initial data set or conditions. If the composer can construct a coded representation, then so can a program. This means the algorithmic level (the composition of the composition) can, in fact, be an artifact of other, higher order abstractions in the machine (the composition of the composition of the composition, and so on).

Algorithmic generation can provide other profound leverages for the composer as well. Very small changes at the algorithmic level can have an enormous impact on the acoustic, performance level that results. Changes to the initial data set can also lead to unintended or unanticipated consequences that trigger new compositional ideas or take the compositional process in a totally new direction. Any composer who has integrated algorithms into compositional process in a meaningful way can cite examples of this phenomenon in their own music.

Relationships that are simple to express at the algorithmic level can also generate extremely complex performances and probability and chance procedures can be

ISSN 0749-4467 (print)/ISSN 1477-2256 (online)
DOI: 10.1080/07494460802664080

deeply expressed in the compositional decision-making when implemented by algorithms. Algorithms allow composers to explore domains that are too complex or too large to imagine doing without digital processing. Algorithmic composition is an essentially empirical activity, one in which the composer experiments with ideas and rapidly tests them out as they develop from an initial curiosity into their final compositional form. Whether or not a composer writes experimental music, generative techniques practically ensure that the composer writes 'experimenting music'. Of course, since the computer is deterministic, strictly speaking there is nothing that cannot also be accomplished by using pen and paper. But the quantitative difference in speed and processing power between computer-based and pen-and-paper-based composition is of such a magnitude that for all intents and purposes it is a qualitatively different activity.

Generative Music: From Koan to Noatikl, Desktop to Browser to Mobile

Tim Cole and Pete Cole

http://www.intermorphic.com

Back in 1986 we became excited by the idea of making a small, spherical 'musical performance device', a kind of musical hyper-instrument. We pulled together a small group of people, and in 1990 SSEYO was formed.

We quickly realised, however, that software was the key and so commenced work instead on the SSEYO Koan Music Engine (SKME), releasing our first software product in 2004. By 1996 Brian Eno had used the SKME in 'Generative Music 1', and in 1997 Timothy Didymus used it in 'Float'.

The SKME grew through trial, error and user feedback. We kept approaches that to us sounded good, and discarded others. We saw the engine as an 'integrator' of a number of interdependent composing voices, each listening to each other's output. For sufficient compositional interest we settled on four key rules at voice and piece level—Scale, Harmony, Next Note and Rhythm—and a number of Voice Types—Fixed Pattern, Ambient, Rhythmic, Follows, Listening and Repeat. For an additional 'organic dimension' we added a range of micro-controllers, envelopes and range guides to limit variation. When it came to sounds, general desktop computers in 1996 had insufficient processing power to generate them through software, so we settled on the best option there was at the time: the Creative Labs AWE soundcard.

Koan generative music had an organic feel, and composers felt they could stamp their own identity on tracks—but there was no integral sound generator, which

meant the pieces we not truly *portable*. To solve that we bolted on, in the ensuing years, an integrated modular software synthesiser and FX network and added support for DLS wavetables. This was included in the Koan player and free Koan plug-ins for Netscape Navigator and Internet Explorer.

Koan development came to an end after SSEYO merged with Tao Group in 2002, but we went on to develop various audio technologies for mobile devices, and a 2004-released mobile music mixer called SSEYO miniMIXA. Just as miniMIXA V3 was being finalised in 2007, Tao folded.

Happily, Intermorphic, the company we founded in 2007, now has rights to all the previous work we did, so we are back on course with a raft of software under our belt.

Intermorphic, like SSEYO, is a toolmaker. Our brand new, clean room trans-generative music engine, Noatikl, can play old Koan material but it has already moved beyond Koan as composition engine. Noatikl's support for Lua scripting means pieces can develop in more dimensions with more structure through programmatic control; hence the descriptor 'trans-generative'.

There are VST and AU sequencer plug-in variants of Noatikl allowing easy coupling to external software sound generators, and we are now including a Noatikl runtime in Mixtikl, a multi-utility mobile-to-desktop music tool. Our Mixtikl xml file formats are open and we are encouraging community sharing of content, paks and scripts.

With Mixtikl and its integrated Partikl soft synth/MIDI synth, FX network and Noatikl runtime, we have realised for mobile (e.g. Windows Mobile, Symbian, etc.) through to desktop (Windows, Mac) a trans-platform solution for music applications and generative music piece portability. Mixtikl apps like 'Performer' will be further developed and extended for generative music performance and interaction.

Our elliptical path may at last be taking us full circle.

John Eacott

The majority of music audiences are probably unaware of the use of generative processes despite the significant numbers of musicians and technologists investigating them. Brian Eno famously argued that generative music is a new 'kind' of music but I prefer to see generative processes as simply one of the tools musicians now have at their disposal. So prevalent in fact are generative processes within the current generation of musicians that for some people they are the norm, rather than an exceptional way of producing and performing. Making algorithms for some musicians is synonymous with composing. One inference that may be drawn from this statement is that the term 'generative' process may eventually fade from our vocabulary, coexisting as it does with the act of creating.

There are many kinds of generative process that may be used in the composition, performance or production of music. So varied are approaches that it is now difficult to argue what is and what isn't a generative process, and some latitude should be accepted. A wide range of musical practices employ one or more of these kinds of process either as a conscious choice by the musician or because they are becoming built into mainstream digital tools.

My own work uses generative processes in various forms. Most of these processes are very basic and consist of what I term 'constrained selection' (i.e. limiting the choices of outputs to those that you want and which work together with other elements). I gradually developed this work by adding 'motif amplification'—building different processes which all read from the same initial data in order to ensure that a piece is unified and coherent.

'Morpheus' (2001) is a CD-ROM of generative electronica featuring sixteen tracks by six composers, Alex Marcou, Ben Milstein, Fredrik Olofsson, Fabrice Mogini, Nick Collins and myself. The idea was to make something that looked and behaved like an ordinary audio CD but used generative processes to ensure that each performance of the music was unique.

The installation 'The Street' (2000) in collaboration with Ross Clement offered a responsive sound environment in a public space. In the first version, ultrasound motion sensors were used to determine the amount of activity in the space and a selection of eight layers of a generative composition was unveiled to reflect this activity. A follow-up, 'Intelligent Street' (2003), created by members from University of Westminster in collaboration with the Interactive Institute in Sweden, allowed users to interact with the sonic environment by sending text messages from their mobile phones.

An example of a more frivolous kind of generative composition is the generative jingle I made for Resonance FM in 2002. The jingle was a simple programme ident following some of the usual conventions. The difference was that rather than being saved as a recording, the ident was run live from a SuperCollider patch and of course varied each time it was played.

My recent works 'Flood Tide—a sonification of the Thames' (2008) and 'Hour Angle' (2008) use generative processes with live performance. As the title suggests, 'Flood Tide' generates music from the flow of the Thames, using a physical impeller in the river connected to a SuperCollider patch that interprets the gradually changing speed of flow to produce live notation for instrumentalists. 'Hour Angle' is based on similar principles but derives data from astronomical calculations of the movement of the sun.

In conclusion, it interesting to note that in the twelve years since Brian Eno's claim that generative music is a 'third kind of music', the notion of Generative Music itself is diminishing, while the use of generative processes by a large number of musicians doing a wide range of work has grown to be the norm rather than the exception.

Sarah Angliss

Over the last few years, I've coded various generative soundpieces, including *Dubiofossil,* a genetic algorithm that evolves hurdy-gurdy music, and *Music for Butterflies*, a delicate, ever-changing ostinato that's created on the fly in London Zoo. Using a camera connected to an optical flow analyser, *Music for Butterflies* responds instantaneously to the butterflies' movements as they flit between the trees.

Despite this track record, in all honesty, I'm unconvinced that my generative pieces are examples of 'good' music. Moreover, I'm unconvinced that the generative processes I've used—including genetic algorithms, random walks and the sonification of live events—are ever going to create music that would earn the adjective 'good'. I'm aware that the word 'good' should be applied with caution to something as subjective as music, an artefact which is created to fulfil so many different functions (whether it's communicating profound emotions, or simply passing time in a lift). But very briefly, these are the problems I've encountered:

Whether I'm creating an artificial musician for the stage or a gallery piece that plays endless variations throughout the day, I want to use generative techniques to create something more enticing than a pre-recorded track. On stage, for example, a generative drummer could have a degree of autonomy and scope for delightful surprise, unlike a simple drum loop. I've spent many hours creating generative soundpieces for such settings. However, despite my attempts to infuse the code with fitness functions and other programming elements that somehow reflect my tastes, when the music emerges, the autonomy I perspired over invariably sounds like randomness. But if I tweak the system to make it more predictable, it sounds boring because it's too unadventurous—arguably, less interesting than a pre-composed piece. The 'good' music seems to exist somewhere between these two extremes—but is always out of reach. Actually, it might only be a few tweaks of the code away, but herein lies the problem: I can't navigate my way to the sounds I want to hear, due to the indirect coupling between my intentions and the resulting music. This makes generative composition a frustrating, intractable process—like controlling a pen in the dark, using a curious pulley system.

In this regard, I feel my most promising generative piece is *Music for Butterflies*, as it incorporates the real-time sonification of the intentional movements of animals. This certainly makes it easier to predict the character of the resulting piece (and shape it by tweaking the camera set-up). I'm pleased with the music that results, although I don't know if listeners can tell that the music is controlled by the butterflies (something it would be interesting to test).

Does it matter if my generative music isn't 'good'? I think so, as I'm interested in creating music for listeners. I realise this isn't every composer's focus. In critiquing evolutionary music, for example, some have cited Langton's description of artificial life (1987), arguing genetic algorithms make 'music-as-it-could-be', rather than

'music-as-we-know-it'. I'm wary of this argument as it may put evolutionary music beyond human scrutiny. Surely 'music-as-it-could-be' is simply music that's unfamiliar—and unfamiliarity and dislike are closely correlated (Scmidhuber, 1997). If my generative composition creates music that's disliked, then it would be disingenuous to describe it as a successful project.

Lovely Algorithms, Hot Weather and Uninspiring Solfeggio

Eduardo Reck Miranda

I have been preaching the practice of algorithmic composition on many occasions. However, I must admit that less than 20% of my music was composed using algorithms. I do find beauty in algorithmic processes, but I often find their musical rendering somewhat frustrating. It is often the case that algorithmic music processes are more appealing than their actual outcome. At the end of the day my compositional methods often boil down to GOFEM (Good Old Fashioned Electroacoustic Music) practices. Careful listening, honing, and combination of sounds in the studio are central to my creative process. But make no mistake: I am not mad about hiding the source of my sounds, acousmatics, and so on.

I probably started using algorithms in composition because I already knew how to program computers when I decided to become a composer. I was about to graduate in computing when I signed up for a music degree in Brazil. I thoroughly enjoyed spending time browsing the shelves in the music library. One day, I accidentally came across a volume of the *La Revue Musicale* entirely devoted to the work of Xenakis. I did not know any French at the time, but I gathered that something interesting was going on—all very familiar to a computing graduate: Formal Logics, Venn diagrams, and so on. I grabbed a French dictionary and spent months dissecting the volume. Only then did I realise that the music degree was not going to teach me how to compose the kind of music I was after; I suspended it. I got hold of an Apple II computer and an engineer in the university built a MIDI interface for me. After various months developing basic software for the interface I managed to get it to control a Roland Juno 106. The astonishment of hearing a Gaussian-constrained random note generator for the first time in my life had a profound impact on me. But it soon became evident that MIDI had its limitations and I moved on to sound synthesis. I went to continue my studies in the UK, where I was introduced to the CDP system running on the Atari 1040. Excellent stuff: I had the power of synthesising and manipulating sounds at my fingertips. But things got really exciting when I had the opportunity to work with a Cray parallel computer at the Edinburgh Parallel Computing Centre, where I finally combined algorithmic practices with

sound synthesis. I developed Chaosynth, a granular synthesiser controlled by cellular automata. While Chaosynth was not designed to generate entire pieces of music, it generated interesting sounds that I have been using until recently to compose a few GOFEM-like pieces.

I do use algorithms as a matter of fact, but my pieces can seldom be referred to as algorithmic. Why do I use algorithms? I think I would blame the hot humid weather and uninspiring solfeggio lessons. Had that sunny Brazilian afternoon been less excruciatingly hot, I would probably not have gone to the library to seek the breeze from the fans.

A Practical Approach to Generative Music

Aaron McLeran

A common motivation for many an aspiring generative composer is the pursuit of a universal formalism, the result of which will always produce beautiful, perfect music as a logical consequence. Other motivations are based on the desire to distill and reproduce the musical essence from great composers of the past. As a result of our work on Electronic Art's Spore, a mainstream game which uses note-based generative music, we grew to abandon these lofty ideals in favor of a more practical and, for our domain, even more effective approach. We developed a modular system of music generators which connects common-practice tonal compositional methods to seeded random numbers.

Initially, we attempted to create a single, unified generative music system, one based on abstract musical and rhythmic concepts. This approach resulted in an unwieldy and overly complex structure with numerous weaknesses: it was brittle, it took a long time to develop, and it enforced a limiting uniformity across development phases and all areas of the game.

Eventually, due to the practical constraints of a game development environment and because of our audience, we grew to prefer a modular approach. This direction was also motivated by the fact that we were riding a perpetual learning curve. Studying the literature, we investigated various generative music techniques and prototyped different approaches as small, self-contained programs, or 'generators'. We found that generative techniques that focused on analysis were impractical because we needed flexibility in terms of both real-time implementation as well as aesthetics.

Ultimately, a direct application of music perception and of functional harmony was most successful, in terms of both simplicity and efficiency. Furthermore, we found it surprising how much we could accomplish simply by developing what we

called 'seed structure' systems, whereby random numbers in music generators are periodically seeded according to various algorithms.

Initially our successful prototypes were integrated into the larger, singular system. However, the generators eventually began to replace the system as we found greater success by simply plugging them together. They turned out to be easily adaptable and extendable, with individual generators designed for different tasks. For example, a particular situation in the game might employ a network of different types of melody generators, chord generators, rhythm generators or bass generators.

Most importantly, by breaking the system into unitary, tonal-music generators, the process of creating generative music began to resemble traditional, non-generative composition. The only difference: instead of having to decide on discrete and singular musical choices, we were defining ranges of musical possibility within a heuristic and interactive system.

Mobile Autonomous Distributed Music

Masayuki Akamatsu

The computer died. Or at least, desktop-style and laptop-style computing died. As we find it difficult to believe that pioneers used a room-size mainframe, programmed in machine code and created the music so slowly outside of real time (thank you! Max Mathews, etc.), some day we will wonder that composers and performers ever used desktop or laptop computers.

In 1999, I accomplished the 'incubator' project (http://www.iamas.ac.jp/~aka/incubator/). In this work, 50 iMacs (with CRT displays!) were placed on the floor of the venue to form a grid, and connected using the Ethernet to form a local area network. The iMacs' built-in microphones were used as input and the built-in display and built-in stereo speakers were used as output. In total, this system had 50 audio inputs, 50 video outputs and 100 audio outputs. Since the iMac was equipped with a CPU, each of them could process anything and communicate with each other through the Ethernet.

One of the main aims of 'incubator' was to compose network-based music and present it as a real formation with audible and visible presence. Most computer music at the time promoted a centralization of power, assuming a single musician, such as an isolated composer, a virtuoso performer or an orchestra conductor. Thus I wanted to investigate the possibilities of a new musical form with an autonomous distributed processing system and interaction with the audience.

Today, nearly 10 years since 'incubator', the computer is evolving and appears on the palm of my hand. It's not called a computer but is presented as an iPhone, a mobile phone. So I'm producing 'incubator'-like networked music pieces with iPhones. The first one is named 'Snowflakes' and was performed with Mr Fredrik

Olofsson and Mr Itaru Yasuda, in Ogaki (Japan) in September 2008, and in Shanghai (China) in October 2008 (http://akamatsu.org/snowflakes/).

'Snowflakes', compared with 'incubator', simply replaces iMacs with iPhones and changes from Ethernet to Wi-Fi. However, a big difference between them is that the audience with the iPhone can participate, move and leave freely. In addition, I don't provide iPhones but exploit iPhones in the pockets of the audience. In other words, 'Snowflakes' is a flexible open system even though 'incubator' was a fixed closed system. What will happen in this new system?

Is music changed by the form of computer or network? The difference from iMac to iPhone is clear, and the difference will affect the music, or at least that's my opinion. Please remember how music changed when the PA system and the recording system appeared. Eventually, ring-like or necklace-like wearable computers will appear, and then a computer will be implanted and blended into our body. I wish to plan works assuming such a world. Don't use old computers anymore.

Programming by Ear (or How to Give up Control and Still be a Control Freak)

Amy Alexander

I'll start with a conclusion: when I write software as art, I understand everything and nothing about what I'm doing.

People often assume that business and scientific programming paradigms are necessary for writing software-based art. Setting clearly defined goals, implementing them cleanly and sticking to spec are considered to be of central importance to the project—but also implicitly reflect on the worthiness of the programmer. Clean, well-organized code denotes technical skill and, by implication, some sort of puritanical godliness. Working this way no doubt prevents a lot of headaches, and it also allows artists to, in some cases, hire others to program for them. But it doesn't fit the way art is often made: painting and sculpture, for example, are a feedback loop between the work and the artist. The artist starts with an idea, but continually adjusts as she sees the project come to fruition. Responses are often visceral and non-verbal; impulse is as important as conscious thought. Programming, on the other hand, is a highly structured and verbal activity. The 'computers vs. art' dichotomy is by now something of a cliché, disproven by more than a generation of programming artists. Yet artist-programmers must still individually reconcile this inherent contradiction.

Circa 2000, when digital media artists began delving into programming en masse, a number of artists acknowledged that their projects were the result of accidents

produced while coding. These acknowledgments gave rise to criticisms that intentionality was lost—which created further pressure upon coding artists to be technically proficient, well-organized, properly planning programmers. But such criticisms ignore the intentionality of editing: an artist in any medium may create thousands of accidents, but chooses only a few to put in a piece. How is a programming artist who chooses from among her algorithmic accidents different from a photographer who snaps a thousand rapid-fire photos but exhibits only a few?

Before I got started with computers, in some of my past lives I performed as a musician and as an analog video performance artist. Probably as a result, my artistic coding practice is largely improvisatory, and, while not based on accidents, necessarily includes them. I start out with a general idea, and I start coding something in that direction. Then I see what this rough process is doing, and I add, delete and revise code as I go along—largely instinctively and obsessive compulsively. I spend a lot of time tweaking code until it's 'just right' (which of course it never is). Particularly in my live visual projects, interactions take place between the various algorithms I've written, such that the result is something I could not have planned at the start. Similarly, in performance, I have a sense of what is about to happen that guides me as I improvise through—but I don't know exactly what will happen until it happens. Every now and then I'm completely surprised. To put it another way—in both coding and performing, I play it by ear.

David Cope

I read a passage in my youth that resonated with my view of composing. This passage—'artists must struggle with the problems generated by their medium and from these struggles produce elegant solutions'—has remained with me throughout my life, providing me with many important insights. Believing as well in the adage 'divide and conquer' to solve such problems, I have used—and continue to use—what today we call *algorithms* (sets of rules for solving problems in finite numbers of steps) as a model for composing, often called *generative composition*. Thus, over the years, I have employed mathematics, slide rulers, and computers to help me produce what I hope are 'elegant solutions' to the problems generated by my composing mediums. Not surprisingly, using algorithms in creating music is pervasive throughout music history, with the *isorhythms* of the fourteenth century, fugues of the seventeenth century, *Musikalisches Würfelspiele* of the eighteenth century, and serialism of the twentieth century being obvious examples.

Differentiating *composers who use algorithms* from *algorithmic composers* represents an important distinction. 'Composers who use algorithms' incorporate algorithms to achieve momentary effects in their music. In contrast, 'algorithmic composers'

compose works whole cloth using algorithms, thus dealing algorithmically with issues of structure and coherence as well as pitches and rhythms. I define myself as an algorithmic composer.

The music of algorithmic composers presents problems for many listeners that being a composer who uses algorithms does not. Recently, for example, two pianists performed *From Darkness, Light*, a work by one of my computer programs called Emily Howell. The accompanying notes for this performance gave no hint of this work's algorithmic—computer—origins. Following the performance, a member of the audience—a professor at a nearby university and a trained musician—complimented the performers and told them that this work moved him more deeply than any music he had recently heard, including the other works on the program by Bach, Beethoven, Mozart, and Chopin, among others. Several months later I gave a presentation on my algorithmic music that included a taped performance of this same piece by Emily Howell. Of course, I revealed the algorithmic origins of this music during my lecture. This same man spoke to me afterwards and noted—apparently unaware that this piece was identical to the one he had heard previously—that he could tell immediately that this music was computer composed and that it moved him not at all.

One might now ask how the same work can be both good and bad for the same individual? While differing contexts provide a fairly straightforward answer, can it then be said that the quality of music cannot be a priori good or bad, but relies entirely on situational aesthetics? I do not know the answer to this question, but I do know that I will continue being an algorithmic composer, using computers to both compose my music and attempt to understand the nature of music itself however best I can, leaving the rest to philosophers and music cognitionists.

To Beep or Not to Beep

Marcus Pearce

Developments in hardware and software have yielded exciting new opportunities for automatically generating music. However, it is important to clarify the different motivations behind computer programs that compose music. Each implies different goals and, therefore, different methodological requirements for demonstrably achieving those goals. We distinguish four activities involving different motivations.

In *algorithmic composition*, the motivations are artistic: a composer writes computer programs to generate novel musical structures or compositional techniques. Since the computer program is an integral part of the compositional process, its construction is unconstrained methodologically and there is no need to define any rigorous criteria for success.

When the motivation is to develop compositional systems for use by other composers, we have a very different activity, *design of compositional tools*. It is important to distinguish this from algorithmic composition because particular methods (derived from software engineering and HCI) are required to design the tools and evaluate whether they satisfy their design objectives.

Other compositional systems are developed with theoretical concerns in mind. A theory of musical composition can be implemented as a computer program and its success assessed through comparison of music generated by the program with existing music that the theory is intended to cover. This approach affords two advantages: first, all assumptions necessary for the theory to compose music must be included; and second, the theory can be intersubjectively evaluated.

In *computational modelling of musical styles* (CMMS), the motivations are musicological and the theory concerns a musical style. CMMS differs from algorithmic composition in two ways. First, the program should express the theory clearly (allowing it to be modified easily). Second, the theory should be tested empirically through objective statistical analysis either of existing music using the program (e.g. to address relevant musicological questions) or of unaltered representative samples of music generated by the program (e.g. whether it generates all and only compositions in the style as judged by experts). David Cope, for example, not distinguishing his musicological and artistic motivations, doesn't do either of these things, severely limiting the significance of his work on style modelling.

In *computational modelling of music cognition*, the motivations are cognitive-scientific and the theory concerns cognitive processes involved in human composition. In contrast to CMMS, the hypotheses embodied in the program must be derived from psychological experiments and related to a level of description in the mind/brain. Furthermore, the hypotheses are evaluated through systematic, empirical attempts to refute them based on generated compositions. This provides a principled method for testing and improving the theory (Pearce, 2005, ch. 9).

Whilst it is necessary for researchers to embrace developments in related disciplines, it is also important to be clear about motivations, goals and methods used in computer generation of music.

References

Pearce, M. (2005). The construction and evaluation of statistical models of melodic structure in music perception and composition (Doctoral dissertation, Department of Computing, City University, London).

Pearce, M., Meredith, D. & Wiggins, G. (2002). Motivations and methodologies for automation of the compositional process. *Musicae Scientiae, 6*(2), 119–147.

Laurie Spiegel

The distinction between 'generative music' (logic-based, algorithmically specified) and music composed by other means is vague. Rule-based composition was commonplace for centuries prior to computers. We all studied species counterpoint, and many older 'forms' are actually process descriptions (canon, fugue) rather than abstract structures to instantiate.

My own use of logically defined processes are often simulations, of self (a new form of self-expression) or of something external.

By self-simulation I mean that I discovered early in my use of computers that some of my own sonic decision-making was predictable enough to be described by rule. I noticed in my music biases regarding continuity and repetition and shapes and rates of change, and other consistencies in handling of balance, distribution and density. Because I most often use algorithmic logic in real-time interaction with my own spontaneous control, by delegating whatever decision-making I can to a set of computerized rules, I free myself to focus more completely on the aspects of music that I cannot reduce to logic, which I reserve for real-time control. I found that the process of coding the software instructions in which my live input has meaning gave me unprecedented opportunity for introspection and increased self-awareness of how I compose and of how my mind works.

My incorporation of processes external to myself comprises several intersecting subsets, including:

1. Allusion: to very roughly approximate or simulate a natural occurrence that appears to me somehow inherently musical, capturing more like a painting than a photo, a perception of process or shape rather than an exact scientific replication (e.g. 'The Expanding Universe').
2. Inverse Analysis: simply rendering into computerized form rules based on analysis of successful music of the past (e.g. 'A Harmonic Algorithm', which resulted from analyses of Bach Chorales).
3. Scientific modeling: implementing, in a set of software-coded rules, scientific research that describes cognitively meaningful data such as music (e.g. Shannon and Pierce's information theory that formulates how to optimally encode content for intelligible reception, used in several pieces I made at Bell Labs).
4. Mimicry of Process: coding into a computer program the rules by which some natural phenomenon unfolds (e.g. my realization of Kepler's 'Harmonia Mundi').
5. Mimicry of Process Result: literal translation of specific non-musical data to musical data (e.g. my little piece 'Viroid', in which I mapped the genetic content of a simple organism to a set of pitches).
6. Mixed: the specification of one realm of a piece by one generative method while another realm of the same piece is determined by an unrelated method

(e.g. the Knowlton-Spiegel algorithm for an illusion of perpetual acceleration in the rhythmic domain coupled with real-time control of corruption rate on a fixed-data source, aka informational entropy, in the pitch domain, in my 'Orient Express').

I choose among these approaches on the basis of the aesthetic qualities of both creative process and output, on what I can learn by doing them and their inherent fascination.