software studies\ a lexicon

edited by **matthew fuller**

SOFTWARE STUDIES

Software Studies

LEONARDO

Roger F. Malina, Executive Editor
Sean Cubitt, Editor-in-Chief

A complete list of books published in the Leonardo series appears at the back of this book.

Software Studies

A Lexicon

edited by Matthew Fuller

The MIT Press
Cambridge, Massachusetts
London, England

For information about special quantity discounts, please email special_sales@mitpress.mit.edu

This book was set in Garamond 3 and Bell Gothic by Graphic Composition, Inc.

Printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data
Software studies : a lexicon / edited by Matthew Fuller.
p. cm.—(Leonardo books)
Includes bibliographical references and index.
ISBN 978-0-262-06274-9 (hbk. : alk. paper) 1. Computer software. 2. Computers and civilization—Encyclopedias. 3. Programming languages (Electronic computers)—Lexicography. 4. Technology and the arts. I. Fuller, Matthew.
QA76.754.S64723 2008
005.1—dc22

2007039724

10 9 8 7 6 5 4 3 2 1

15. For more on this see, N. Katherine Hayles, *My Mother was a Computer*; Julian Dibbell, "Viruses Are Good For You"; Jussi Parikka, "The Universal Viral Machine."

16. See Wendy Hui Kyong Chun, *Control and Freedom*.

Sonic Algorithm

Steve Goodman

Contemporary sound art has come under the influence of digital simulations. These simulations are based on artificial life models, producing generative compositional systems derived from rules abstracted from actual processes occurring in nature. Yet taking these intersections of algorithms and art, divorced from a wider sonic field can be misleading. With their often arbitrary, metaphorical transcodings of processes in nature into musical notation, uncritical transpositions of artificial life into the artistic domain often neglect the qualitative, affective transformations that drive sonic culture. With care, however, we can learn much about the evolution of musical cultures from conceptions (both digital and memetic) of sonic algorithms—on the condition that we remember that software is never simply an internally closed system, but a catalytic network of relays connecting one analog domain to another. Here, the computing concept of the abstract machine attains a wider meaning, corresponding to the immanent forms that also pattern non-computational culture. For this reason, an analysis of the abstract culture of music requires the contextualization of digital forms within the contagious sonic field of memetic algorithms as they animate musicians, dancers and listeners.

An algorithm is a sequence of instructions performed in order to attain a finite solution to a problem or complete a task. Algorithms predate digital culture and are traceable in their origins to ancient mathematics. Whereas a computer program is the concretization or implementation of an assemblage of algorithms, the algorithm itself can be termed an abstract machine, a diagrammatic method that is programming language independent. Abstract machines are "mechanical contraptions {that} reach the level of abstract machines when they become mechanism-independent, that is, as soon as they can be thought of independently of their specific physical embodiments"[1] thereby intensifying the powers of transmission, replication and proliferation. This quality of algorithms is crucial to software-based music, with key processes distilled to

formalized equations that are generalizable, transferable, reversable, and applied. "Coupled with software (or mechanism or score or programme or diagram) that efficiently exploits these ideas, the abstract machine begins to propagate across the technological field, affecting the way people think about machines and eventually the world."[2] The affective power of the sonic algorithm is not limited to the morphology of music form. Leaking out of the sterile domain of the digital sound lab and across the audio-social field, these abstract machines traverse the host bodies of listeners, users, and dancers, producing movements and sensations, before migrating back to the vibratory substrate.

If, as Gottfried Leibniz proposed, all music is "unconscious counting,"[3] then clearly, despite its recent popularity, algorithmic music composition cannot be considered the exclusive domain of computing. It should instead be placed in an historical context of experiments with, for example, out of phase tape recorders, where tape loops already constituted "social software organized to maximize the emergence of unanticipated musical matter."[4] As Michael Nyman has outlined, bottom-up approaches to musical composition take into account the context of composition and production as a system, and are "concerned with actions dependent on unpredictable conditions and on variables which arise from within the musical continuity."[5] Examples from the history of experimental music can be found in the oft-cited investigations of rule-centered sonic composition processes in the exploration of randomness and chance, such as John Cage's use of the I Ching, Terry Riley's "In C," Steve Reich's "It's Gonna Rain" and "Come Out," Cornelius Cardew's "The Great Learning," Christian Wolff's "Burdocks," Frederic Rzewski's "Spacecraft," and Alvin Lucier's "Vespers."[6] In this sense, as Kodwo Eshun argues, the "ideas of additive synthesis, loop structure, iteration and duplication are pre-digital. Far from new, the loop as sonic process predates the computer by decades. Synthesis precedes digitality by centuries."[7]

Recent developments in software music have extended this earlier research into bottom-up compositional practice. Examples centering around the digital domain include software programs such as Supercollider, MaxMsp, Pure Data, Reactor and Camus,[8] which deploy mathematical algorithms to simulate the conditions and dynamics of growth, complexity, emergence, and mutation of evolutionary algorithms and transcode them to musical parameters. The analysis of digital algorithms within the cultural domain of music is not limited to composition and creation. Recent Darwinian evolutionary musicology has at-

tempted to simulate the conditions for the emergence and evolution of music styles as shifting ecologies of rules or conventions for music-making. These ecologies, it is claimed, while sustaining their organization, are also subject to change and constant adaption to the dynamic cultural environment. The suggestion in such studies is that the simulation of complexity usually found within biological systems may illuminate some of the more cryptic dynamics of musical systems.[9] Here, music is understood as an adaptive system of sounds used by distributed agents (the members of some kind of collective; in this type of model, typically, none of the agents would have access to the others' knowledge except what they hear) engaged in a sonic group encounter, whether as producers or listeners. Such a system would have no global supervision. Typical applications within this musicological context attempt to map the conditions of emergence for the origin and evolution[10] of music cultures modeled as "artificially created worlds inhabited by virtual communities of musicians and listeners. Origins and evolution are studied here in the context of the cultural conventions that may emerge under a number of constraints, for example psychological, physiological and ecological."[11] Eduardo Miranda, despite issuing a cautionary note on the limitations of using biological models for the study of cultural phenomena,[12] suggests that the results of such simulations may be of interest to composers keen to unearth new creation techniques. He asserts that artificial life should join acoustics, psychoacoustics, and artificial intelligence in the armory of the scientifically upgraded musician. According to Miranda, software models for evolutionary sound generation tend to be based on engines constructed around cellular automata or genetic algorithms.

Cellular automata were invented in the 1960s by von Neumann and Stanislaw Ulam as simulations of biological self-reproduction.[13] Such models attempted to explain how an abstract machine could construct a copy of itself automatically. Cellular automata are commonly implemented as an ordered array or grid of variables termed cells. Each component cell of this matrix can be assigned values from a limited set of integers, and each value usually corresponds with a color. On screen, the functioning cellular automata is a mutating matrix of cells that edges forward in time at variable speed. The mutation of the pattern, while displaying some kind of global organization, is generated only through the implementation of a very limited system of rules that govern locally. Heavily influential to generative musicians such as Brian Eno, the most famous instantiation of cellular automata is John Conway's Game

of Life (1967). Game of Life has recently been implemented in the software system CAMUS, whereby the emergent behaviors of cellular automata are developed into a system that transposes the simple algorithmic processes into musical notation. The rules of the Game of Life are very simple. In the cellular grid, a square can be either dead or alive. With each generation, or step of the clock, the squares change status. A square with one or zero neighbors will die. A square with two neighbors will survive. A square with three neighbors becomes alive if not already, and a square with four or more neighbors will die from overcrowding. The focus of such generative music revolves around the emergent behavior of sonic lifeforms from their local neighborhood interactions, where no global tendencies are preprogrammed into the system.

As in the case of cellular automata and artificial neural networks, models based around genetic algorithms transpose a number of abstract models from biology, in particular the basic evolutionary biological processes identified in particular by Darwin[14] and updated by Dawkins.[15] These algorithms are often used to obtain and test optimal design or engineering results out of a wide range of combinatorial possibilities. Simulations so derived allow evolutionary systems to be iteratively modeled in the digital domain without the inefficiency and impracticality of more concrete trial and error methods. But, as Miranda points out, by abstracting from Darwinian processes such as natural selection based on fitness, crossover of genes, and mutation, "genetic algorithms go beyond standard combinatorial processing as they embody powerful mechanisms for targeting only potentially fruitful combinations."[16] In practice, genetic algorithms will usually be deployed iteratively (repeated until fitness tests are satisfied) on a set of binary codes that constitute the individuals in the population. Often this population of code will be randomly generated and can stand in for anything, such as musical notes. This presupposes some kind of codification schema involved in transposing the evolutionary dynamic into some kind of sonic notation, which, as Miranda points, out will usually seek to adopt the smallest possible "coding alphabet." Typically each digit or cluster of digits will be cross-linked to a sonic quality such as pitch, or specific preset instruments as is typical in MIDI. This deployment consists of three fundamental algorithmic operations, which, in evolutionary terms, are known as recombination (trading in information between a pair of codes spawning offspring codes through combining the "parental" codes), mutation (adjusting the numerical values of bits in the code, thereby adding diversity to the

population) and selection (choosing the optimal code based on predetermined pre-coded fitness criteria or subjective/aesthetic criteria). One example of the application of genetic algorithms in music composition is Gary Lee Nelson's 1995 project *Sonomorphs*, which used

> genetic algorithms to evolve rhythmic patterns. In this case, the binary-string method is used to represent a series of equally spaced pulses whereby a note is articulated if the bit is switched on . . . and rests are made if the bit is switched off. The fitness test is based on a simple summing test; if the number of bits that are on is higher than a certain threshold, then the string meets the fitness test. High threshold values lead to rhythms with very high density up to the point where nearly all the pulses are switched on. Conversely, lower threshold settings tend to produce thinner textures, leading to complete silence.[17]

In summary, then, the development of artificial life techniques within music software culture aims to open the precoded possibilities of most applications to creative contingency.[18] The scientific paradigm of artificial life marks a shift from a preoccupation with the composition of matter to a focus on the systemic interactions between the components out of which nature is under constant construction. Artificial life uses computers to simulate the functions of these actual interactions as patterns of information, investigating the global behaviors that arise from a multitude of local conjunctions and interactions. Instead of messy biochemical labs deployed to probe the makeup of chemicals, cells, etc., these evolutionary sonic algorithms instantiated in digital software take place in the artificial worlds of the CPU, hard disk, the computer screen, and speakers. However, with an extended definition of an abstract machine, sonic algorithms beyond artificial life must also describe the ways in which software-based music must always exceed the sterile and often aesthetically impoverished closed circuit of digital sound design. With non-software musics, such abstract machines leak out in analog sound waves, sometimes laying dormant in recorded media awaiting activation, sometimes mobilizing eardrums and bodies subject to coded numerical rules in the guise of rhythms and melodies. The broader notion of the abstract machine rewrites the connection between developments in software and a wider sonic culture via the zone of transduction between an abstract sonic pattern and its catalytic affects on a population. By exploring these noncomputational effects and the propagation

of these sonic algorithms outside of digital space, software culture opens to the outside that was always within.

Notes

1. Manuel De Landa, *War in the Age of Intelligent Machines*, 142.

2. Kodwo Eshun, "An Unidentified Audio Event Arrives from the Post-Computer Age," in Jem Finer, ed., *Longplayer*, 11.

3. Gottfried W. Leibniz, *Epistolae ad diversos*, 240.

4. Eshun, "An Unidentified Audio Event Arrives from the Post-Computer Age," 11.

5. Michael Nyman, *Experimental Music: Cage & Beyond.*

6. David Toop, "Growth and Complexity," in *Haunted Weather.*

7. Eshun, "An Unidentified Audio Event Arrives from the Post-Computer Age," 11.

8. Supercollider (http://www.audiosynth.com/), MaxMsp (http://www.cycling74.com/), Pure Data (http://puredata.info/), Reactor (http://www.native-instruments.com), Camus (http://website.lineone.net/~edandalex/camus.htm).

9. See, for example, Peter Todd, "Simulating the Evolution of Musical Behavior," 361–389.

10. Eduardo Miranda, *Composing Music with Computers*, 139–143, points to four mechanism in origins and evolution useful for modeling musical systems:

a. transformation and selection (should preserve the information of the entity): improve components of ecosystem evolution of music subject to psychophysiological constraints rather than biological needs (i.e., survival); exceptions are bird song and mate attraction
b. co-evolution: pushes the whole system (of transformations and selections) toward greater complexity in a coordinated manner, e.g., musical styles co-evolve with music instruments/ technologies.
c. self-organization: coherence ingredients include (1) a set of possible variations, (2) random fluctuations, and (3) a feedback mechanism.

d. level formation: formation of higher level compositional conventions, e.g., abstract rules of rhythm e.g. metre and a sense of hierarchical functionality.

11. Miranda 2001, 119.

12. Miranda is particularly cautious of linear, progressive models of evolution:

> Evolution is generally associated with the idea of the transition from an inferior species to an superior one and this alleged superiority can often be measured by means of fairly explicit and objective criteria: we believe, however, that this notion should be treated with caution . . . with reference to prominently cultural phenomena, such as music, the notion of evolution surely cannot have exactly the same connotations as it does in natural history: biological and cultural evolution are therefore quite different domains. Cultural evolution should be taken here as the transition from one state of affairs to another, not necessarily associated with the notion of improvement. Cultural transition is normally accompanied by an increase in the systems' complexity, but note that "complex" is not a synonym for "better." (140)

13. E. F. Cood, *Cellular Automata.*

14. Charles Darwin, *The Origin of Species*, 1859.

15. Richard Dawkins, *The Blind Watchmaker*, 1986.

16. Eduardo Miranda, *Composing Music with Computers*, 131.

17. Ibid., 136.

18. See Peter Todd, "Simulating the Evolution of Musical Behavior," and Eleonora Bilotta, Pietro Pantano, and Valerio Talarico, "Synthetic Harmonies: An Approach to Musical Semiosis by Means of Cellular Automata."