Granular Synthesis of Sounds by Means of a Cellular Automaton

Eduardo Reck Miranda

ABSTRACT

Chaosynth is a new sound synthesis system being developed by the author and others working at Edinburgh University. Chaosynth functions by generating a large amount of short sonic events, or particles, in order to form larger, complex sound events. This synthesis technique is inspired by granular synthesis. Most granular synthesis techniques, however, use stochastic methods to control the formation of sound events, while Chaosynth uses a cellular automaton. Following an introduction to the basics of granular synthesis, the author explains how Chaosynth's technique works. He then introduces the basics of cellular automata and presents ChaOs, the cellular automaton used in Chaosynth. The article concludes with some final remarks and suggestions for further work.

GRANULAR SYNTHESIS AND CHAOSYNTH

The granular synthesis of sounds involves the production of thousands of short sonic particles—for example, 30-millisecond-long sounds—that are combined to form larger sound events. This synthesis technique was inspired by Denis Gabor's proposition that large, complex sound events are composed of simple acoustic particles, or sonic grains [1]; he suggested that a granular representation can be used to describe sounds with complex morphology. Norbert Wiener [2] also adopted a "granular" representation of sounds to measure the information content of a sonic message. It was the composer Iannis Xenakis [3], however, who suggested the first theory of granular synthesis for musical purposes. Since then, a few others (Barry Truax [4] and Curtis Roads [5], for example) have proposed granular synthesis systems.

So far, most of these systems use stochastic methods to control the production of sonic particles (for example, density and duration of particles). Chaosynth proposes a different method: the use of cellular automata (CA) [6–8].

THE BASICS OF CELLULAR AUTOMATA

Cellular automata are mathematical models of dynamic systems in which space and time are discrete and quantities take on a finite set of discrete values. A cellular automaton is often represented as a regular array with a discrete variable at each site, referred to as a cell. The state of a cellular automaton is specified by the values of the variables at each cell. The automaton evolves in synchronization with the tick of an imaginary clock according to an algorithm that determines the value of a cell based on the value of its neighborhood [9,10]. This algorithm is called global transition function, or simply F. As implemented on a computer, the cells are represented as a grid of tiny rectangles whose states are indicated by different colors.

Cellular automata were originally introduced in the 1960s by John von Neumann [11] as a model of biological self-reproduction. He wanted to know if it is possible for a machine to reproduce—that is, to automatically construct a copy of itself. His model consisted of a two-dimensional (2D) grid of cells, each of which was in one of a number of states; each state represented the components of the self-reproducing machine. Controlled completely by the global transition function designed by von Neumann, the machine (a pattern of cells in the grid) would extend an arm into a virgin portion of the universe (that is, the grid), then slowly scan it back and forth, creating a copy of itself.

A wide variety of cellular automata and transition functions have been invented and adapted for many modelling purposes. Cellular automata have also attracted the interest of musicians because of their organizational principles. Various composers and researchers have used cellular automata to aid the control of both higher-level musical structures (musical forms) and lower-level sound structures (the spectra of individual sound events) [12–18]. Chaosynth uses cellular automata to control the inner structure of sounds.

THE CHAOS CELLULAR AUTOMATON

The Metaphor

ChaOs (an abbreviation of Chemical Oscillator) is a metaphorical model of a neurophysiological phenomenon known as a neural reverberatory circuit [19,20]. ChaOs can be thought of as an array of identical electronic circuits called nerve cells. At any moment, a nerve cell can be in a quiescent, depolarized or burned state. The array tends to evolve from an initial random distribution of these three states in the grid to an oscillatory cycle of wave patterns. The behavior of ChaOs resembles the way in which most of the natural sounds produced by acoustic instruments evolve: sounds tend to converge from a wide distribution of their partials (for example, noise) to oscillatory patterns (for example, a sustained tone).

A nerve cell and its neighborhood interact through the flow of electric current between them. Minimum (Vmin) and maximum (Vmax) threshold values characterize the state of a nerve cell. If the nerve cell's voltage (Vi) is under Vmin, then the nerve cell is quiescent (or polarized). If it is between Vmin (inclusive) and Vmax values, then it is being depolarized. Each nerve cell has a potential divider that is aimed at maintaining Vi below Vmin. When the divider fails (that is, if Vi reaches Vmin) the nerve cell becomes depolarized. There is also an electric capacitor that regulates the rate of depolarization. Cells have a tendency, however, to become increasingly depolarized with time. When Vi reaches Vmax, the nerve cell fires and is burned. A burned nerve cell at time t is auto-

Eduardo Reck Miranda (researcher), Faculty of Music and Edinburgh Parallel Computing Centre (EPCC), University of Edinburgh, 12, Nicolson Square, Edinburgh, EH8 9DF, Scotland, United Kingdom. E-mail: <miranda@music.ed.ac.uk>.

This article is an extension of a paper presented at the Fourth International Symposium on Electronic Art (FISEA 93), Minneapolis, Minnesota, U.S.A., 5–7 November 1993.

matically replaced by a new, quiescent nerve cell at time t+1.

The behavior of ChaOs is specified by setting up a number of parameters:

- the number n of states, such that n≥3
- the resistances r1 and r2 of the potential divider
- the capacitance k of the rate of depolarization
- the speed t of the imaginary clock
- the dimension of the grid.

The Algorithm

The states of nerve cells are represented by a number between 0 and n–1 (n = amount of different states). A nerve cell in state 0 corresponds to a quiescent state, while a nerve cell in state n–1 corresponds to a burned state. All states between the two exhibit a degree of depolarization corresponding to their state number. The closer a nerve cell's state number gets to n–1, the more depolarized it becomes.

The global transition function F is defined by three rules simultaneously applied to each nerve cell, and selected according to its current state: quiescent, burned or depolarized. The rules are as follows:

1. If the cell is quiescent, it may or may not become depolarized at the next tick of the clock (t+1). This depends upon the number of polarized nerve cells (Pcells) in its neighborhood (eight neighbors), the number of burned nerve cells (Bcells) in its neighborhood and the resistance to fire (r1 and r2) of the nerve cell:

$$\text{if } cell(n)_t = 0,$$
$$\text{then } cell(n)_{t+t} = int(Pcells(n)/r1)_t + int(Bcells(n)/r2)_t$$

2. If the cell is depolarized, its tendency is to become more depolarized as the clock t evolves. Its state at the next tick of the clock (t+1) depends on two factors: the capacitance k of the nerve cell and the degree of polarization of its neighborhood. The degree of polarization of the neighborhood is the sum of the numbers that correspond to the states of the eight neighbors (Pdegree) divided by the number of polarized neighbor (Pcells):

$$\text{if } 0 < cell(n)_t < n-1$$
$$\text{then } cell(n)_{t+1} = k + int(Pdegree(n)/Pcells(n))_t$$

3. If a cell is burned at time t, it generates a new, quiescent nerve cell at time t+1:

$$\text{if } cell(n)_t = n-1$$
$$\text{then } cell(n)_{t+1} = 0$$

The Mapping Technique

The organization principles of ChaOs intuitively suggest that it could be applied to control the production of large numbers of sonic particles that together form a complex sound event. Finding an effective way to map the behavior of ChaOs onto the parameters of a synthesi algorithm is not, however, a straightforward task. I have devised and te ted several techniques, but only a few produced interesting sounds. Following i an introduction to the technique currently implemented in Chaosynth.

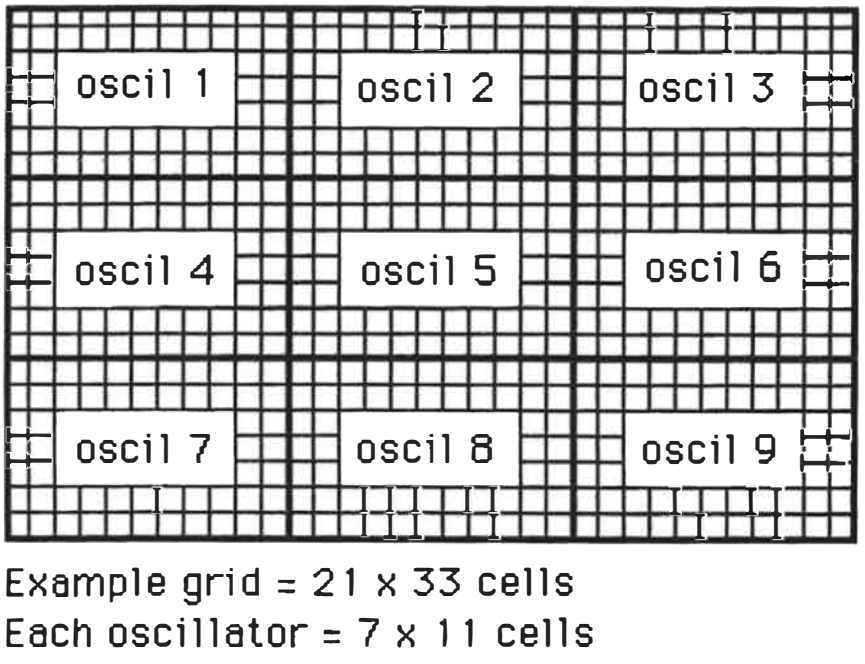
Each sonic particle produced by Chaosynth is composed of several partials. Each partial is a sine wave produced by an oscillator. An oscillator needs three parameters to function: frequency, amplitude and duration (in millisecond) of the sine wave. ChaOs controls the frequency values and the duration of a particle, but the amplitude values are set up by the user beforehand. The states of nerve cells are associated with a frequency value and oscillators are associated with a certain number of nerve cells. The frequency values of partials at time t are therefore established by the arithmetic mean of the frequencies associated with the states of the nerve cells of the oscillators. The user also specifies the dimension of the grid, the amount of oscillators, the allocation of nerve cells to oscillators and the parameters of ChaOs (that is, the number of states, the resistances of the potential divider, the capacitance of cells and the number of iterations).

Each particle is, in fact, the product of the additive ynthesis [21] of sine waves: at each iteration of ChaOs, all oscillators simultaneously produce sine waves whose frequencies are determined by the arithmetic mean over the frequency values of their corresponding nerve cells. The duration of a whole sound event is determined by the number of iterations and the duration of the particles. For example, 100 iterations of 30-millisecond particles result in a sound event of 10 seconds in duration. An example of a grid of 693 nerve cells allocated to 9 oscillators i shown in Fig. 1.

This mapping method is interesting because it explores the behavior of ChaOs in order to produce sounds in a way that resembles the functioning of some acoustic instruments (for example, the violin and the human voice). The random initialization of states in the grid produces an initial wide distribution of frequency values, which tend to settle into an oscillatory cycle. This behavior resembles the way in which the sounds produced by most acou tic instruments evolve during their production: their harmonics converge from a wide distribution (as in the noise of the attack part of a vocal sound, for example) to oscillatory patterns (the characteristic of a su tained tone).

We have synthesized sounds u ing up to 40 different ChaOs states (that is, 40

Fig. 1. An example of a grid of 693 nerve cells distributed to 9 oscillators; each oscillator, in this case, holds 77 nerve cells. The oscillators produce sine waves whose frequency values are determined by the arithmetic mean over the values of their corresponding nerve cells, according to the state of the cellular automaton.



different frequency values) and up to 25 oscillators on a 1,000,000 nerve-cell grid (1,000 × 1,000). The results resemble the sounds of flowing water. One can produce a wide range of gurgling sounds in various flow speeds with Chaosynth by varying the speed of the cellular automata's internal clock. Variations in tone color are achieved by varying the frequency values and the amount of nerve cells per oscillator. Different rates of transition from noise to oscillatory patterns are obtained by changing the values of r1, r2 and k.

The Chaosynth Program and the Parallelization Technique

The architecture of the program is shown in Fig. 2. The sounds are synthesized using Csound [22]. Csound is software for sound synthesis in which one specifies a synthesis algorithm in an orchestra file and a list of synthesis parameters in a score file. When the Csound compiler is activated, it reads these two files and produces a sound file for playback.

Chaosynth's user interface triggers the cellular automaton, which produces a Csound score file. The score file activates the Csound compiler and plays the sound.

The current version of Chaosynth (1.0) uses parallelization in order to speed up the cellular automata algorithm. To parallelize the cellular automata algorithm, we took advantage of the Parallel Utility Library–Regular Decomposition (PUL-RD) utility at the Edinburgh Parallel Computing Centre (EPCC) [23]. PUL-RD is a utility for the parallelization of grid-based problems using the regular domain-decomposition technique. The regular domain-decomposition technique involves arranging a very large set of data in a grid, then computing this data in parallel. The PUL-RD utility splits up a large grid of data elements into regular subgrids, then distributes them for concurrent processing. This allows much larger computations to be solved in shorter periods of time. Cellular automata are a typical case of regular domain decomposition.

In Chaosynth, PUL-RD is used to split the grid of nerve cells into subgrids; each subgrid corresponds to an oscillator. In this case, Chaosynth computes all oscillators in parallel.

We are currently parallelizing the sound-synthesis process of Chaosynth.

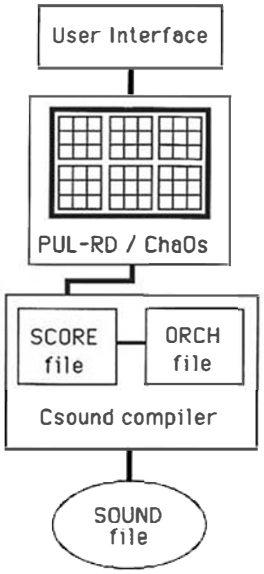


Fig. 2. The system architecture of Chaosynth. The system is divided into three main modules: the user interface, the cellular automata engine and the sound synthesis processor. The cellular automata engine benefits from parallel computing techniques; it provides the parameter values for the sound synthesis processor.

The Csound program takes too long to compile the sound files. Moreover, Csound is not suitable for synthesis in real time; it takes a few moments to produce a sound file for playback. Our aim is to provide Chaosynth with means for real-time sound synthesis. To accomplish this, we are devising ways to provide Chaosynth with its own sound synthesis module. This synthesis module will feature parallelization in order to speed up the synthesis process. We expect to be able to provide real-time changing of the parameters of Chaosynth, so that the user can actually "play" it as a musical instrument by using appropriate controllers (for example, joysticks, MIDI devices [24], the Dataglove and Biomuse [25]).

Conclusion and Further Work

In this article I introduced Chaosynth, a cellular automata–controlled additive synthesizer that generates a large amount of short sonic particles in order to form larger, complex sound events. Chaosynth can produce a wide variety of sounds; however, more research is needed to gain a better understanding of the role of Chaosynth's parameters.

The cellular automaton used by Chaosynth to control the production of the sonic particles is called ChaOs; it mimics a neurophysiological phenomenon. Of the many possible ways to map the behavior of ChaOs onto the parameters of the synthesis algorithm, we have implemented only one, which is capable of producing interesting sounds. We are, however, aware that instead of providing a system that uses only one mapping possibility, we should provide the means for user-specification of other mapping possibilities. We are currently studying how to provide this facility.

Although the parallelization of the cellular automata algorithm provides high performance computing to Chaosynth, the system still does not produce sounds in real time; the user has to wait a few seconds until the sound can actually be heard. We are currently parallelizing the synthesis module of Chaosynth (Fig. 2) in order to be able to produce sounds in real time. We intend to allow the user to change its parameter values during the production of the sound, as if he or she were playing a musical instrument, by using appropriate controllers.

We have produced an electroacoustic music composition, "Olivine Trees," using Chaosynth's sounds. "Olivine Trees" was inspired by a Van Gogh painting, "Olive Trees" (National Gallery of Scotland, Edinburgh). The varied and individually identifiable brush strokes of this painting inspired the composition of the sounds of this piece; in direct correlation, color relates to timbre and length of brush stroke relates to the duration of individual sounds. Olivine is the name of the EPCC workstation where we worked with Chaosynth to produce the sounds of the piece. Olivine Trees is perhaps the first piece of music ever composed using a parallel computer.

Chaosynth is available to composers as part of the EPCC's Computer Music Workstation set of programs.

References

1. D. Gabor, "Acoustical Quanta and the Theory of Hearing," *Nature* 159, No. 1044, 591–594 (1947).

2. N. Wiener, "Spatial-temporal Continuity, Quantum Theory, and Music," in M. Capeck, ed., *The Concepts of Space and Time* (London: Reidel, 1964).

3. I. Xenakis, *Formalized Music* (Indiana: Indiana Univ. Press, 1971).

4. B. Truax, "Real Time Granular Synthesis with a DSP Computer," *Computer Music Journal* 12, No. 2, 14–26 (1988).

5. C. Roads, "Asynchronous Granular Synthesis," in G. De Poli et al., eds., *Representation of Music Signals* (Boston, MA: MIT Press, 1991).

6. G. Wilson, "The Life and Times of Cellular Automata," *New Scientist* (8 October 1988) pp. 44–47.

7. E.R. Miranda, P. Nelson and A. Smaill, "ChaOs: a Model for Granular Synthesis by Means of Cellular Automata," *Annual Report and Project Directory 1991–92* (Edinburgh Parallel Computing Centre, 1992) pp. 153–156.

8. M.D. Westhead, "Chaosynth: A System for Granular Synthesis Using a Cellular Automaton," EPCC Summer Course Project Report (Edinburgh University, 1993).

9. A.K. Dewdney, "A Cellular Universe of Debris, Droplets, Defects and Demons," *Scientific American* (August 1989) pp. 88–91. See also Wilson [6].

10. S. Wolfram, "Statistical Mechanics of Cellular Automata," *Reviews of Modern Physics* 55, No. 3, 601–623 (1983).

11. E.F. Cood, *Cellular Automata* (London: Academic Press, 1968).

12. E.R. Miranda, "Cellular Automata Music: An Interdisciplinary Project," *Interface* 22, No. 1, 3–21 (1963).

13. E.R. Miranda, "Cellular Automata Music Composition: A Bio-Logical Inspiration," *Proceedings of the Fourth International Symposium on Electronic Art* (Minneapolis, MN: ISEA, 1993) pp. 95–109.

14. E.R. Miranda, "Cellular Automata Music Composition," *Languages of Design* 2, 105–107 (1994).

15. R. Orton, A. Hunt and R. Kirk, "Graphical Control of Granular Synthesis Using a Cellular Automata and the Freehand Program," *Montreal ICMC 1991 Proceedings* (San Francisco, CA: ICMA, 1991) pp. 416–418.

16. A. Hunt, R. Kirk, and R. Orton, "Musical Applications of a Cellular Automata Workstation," unpublished Music Technology Group discussion paper (University of York, 1991).

17. M. Hamman, "Mapping Complex Systems Using Granular Synthesis," *Montreal ICMC 1991 Proceedings* [15] pp. 475–478.

18. D. Millen, "Cellular Automata Music," ICMC Glasgow 1990 Proceedings (San Francisco, CA: ICMA, 1990) pp. 314–316.

19. J.C. Eccles, "The Physiology of Imagination," reprint from *Scientific American* (New York: W.H. Freeman, 1958).

20. R.H.S. Carpenter, *Neurophysiology* (London: Edward Arnold, 1990).

21. C. Doge and T. Jerse, *Computer Music—Synthesis, Composition, and Performance* (London: Schirmer Books, 1985).

22. B. Vercoe, *Csound Manual*, available via the Internet through ftp from the music directory of the host machine <media-lab.mit.edu>.

23. S. Brown, "Parallelising Grid-Based Applications with PUL-RD" (Edinburgh Parallel Computing Centre, 1994).

24. J. Pressing, *Synthesizer Performance and Real-Time Techniques* (Oxford, UK: Oxford Univ. Press, 1992).

25. A. Mulder, "Virtual Musical Instruments: Accessing the Sound Synthesis Universe as a Performer," 1 *Simpósio Brasileiro de Computaçao e Música, Proceedings* (Brazil: SBC, 1994) pp. 243–250.