

Introduction to Cellular Automata Music Research

Introduction

Music has always been an interesting domain for the application of new scientific discoveries inviting composers to combine artistic creativity with scientific methods. Today, it is becoming increasingly common for the composer to turn to the sciences to supplement his or her compositional models. By the same token, scientists also seem to show interest in the organisational principles to be found in music.

Scientific models carry an important component of human thought, namely formal abstraction, which can be very inspiring for music composition. My research work attempts to identify a correlation among different disciplines such as biology, crystallography and computing in order to investigate the possibility of synthesising sounds and composing music inspired by this interdisciplinary framework.

A class of mathematical models known as cellular automata plays a central role in my research. Cellular automata have been used to model a wide range of scientific phenomena. They have been studied and developed for over three decades. Although very simple, they can provide models for a wide variety of complex phenomena in a range of disciplines including physics (e.g., dynamic and chaotic systems), genetics and chemistry (e.g., chemical reactions and crystal growth).

Cellular Automata

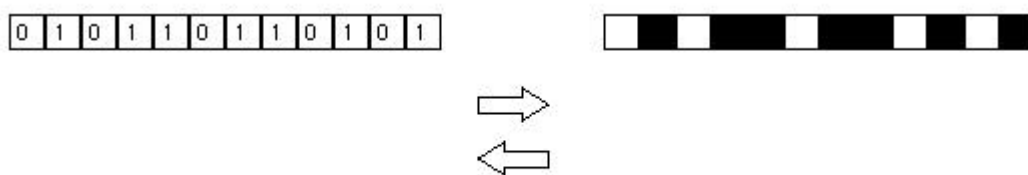
Cellular automata (CA) are computer modelling techniques widely used to model systems in which space and time are discrete, and quantities take on a finite set of discrete values.

Cellular automata were originally introduced in the sixties by von Neumann and Ulan as a model of a biological self-reproduction. They wanted to know if it would be possible for an abstract machine to reproduce; that is, to automatically construct a copy of itself. Their model consisted of a two-dimensional grid of cells, each cell of which had a number of states, representing the components out of which they built the self-reproducing machine. Controlled completely by a set of rules designed by its creators, the machine would extend an arm into a virgin portion of the grid, then slowly scan it back and forth, creating a copy of itself - reproducing the patterns of cells at another location in the grid.

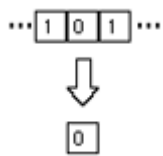
Since then cellular automata have been repeatedly reintroduced and applied to a considerable variety of purposes. Many interesting algorithms have been developed during the past thirty years. In general, CA are implemented as a regular array or matrix of variables called cells. Each cell may assume values from a finite set of integers and each value is normally associated with a colour. The functioning of a cellular automaton is displayed on the computer screen as a sequence of changing patterns of tiny coloured cells, according to the tick of an imaginary clock, like an animated film. At each tick of the clock, the values of all cells change simultaneously, according to a set of transition rules that takes into account the values of their neighbourhood.

The following figure illustrates a very simple CA. It consists of an array of 12 cells and each cell can value either 0 or 1, represented by the colours white or black, respectively.

A very simple CA. The right hand figure displays the colours associated to cell values; in this case 0=white and 1=black.



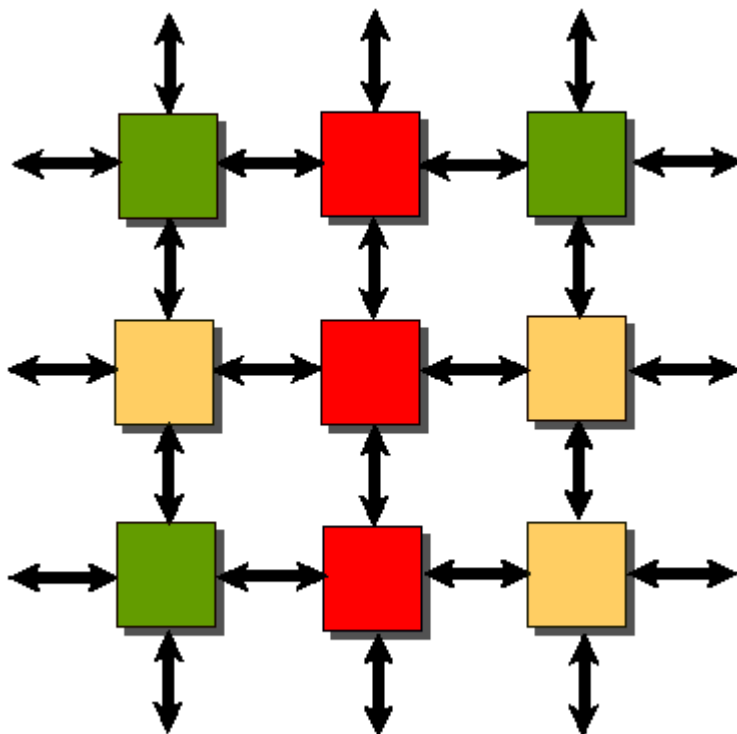
At each tick of the clock, the values of all 12 cells change simultaneously according to a set of rules that determines a new value for each cell. In this case, the rules are based upon the values of



its two neighbours. For example, if a cell is equal to 0 and if both neighbours are equal to 1, then this cell continues equal to zero in the next stage (see figure below).

An example of a CA rule in action.

More sophisticated CA configurations use a matrix of cells that can assume values other than 0 and 1 (and consequently, colours other than black and white). In these cases, the transition rules normally computes four or eight neighbours.



An example of a matrix of cells where the transition rule computes 4 neighbours.

A wide variety of CA and transition rules have been invented and adapted for a variety of modelling purposes in many scientific areas, including physics, computing, biology and meteorology. CA have also attracted the interest of musicians because of their organisational principles. Many composers and researchers have used CA to aid the control of both higher-level musical structures, or musical form (e.g., to generate melodies, phrases and entire pieces of music), and lower-level structures, such as the spectra of individual sound events (e.g., sound synthesis research).

CAMUS A Cellular Automata Music Generator

Since cellular automata produce large amounts of patterned data and if we assume that music composition can be thought of as being based on pattern propagation and formal manipulation of its parameters, it comes as no surprise that researchers started to suspect that cellular automata could be mapped into a music representation in order to generate compositional material.

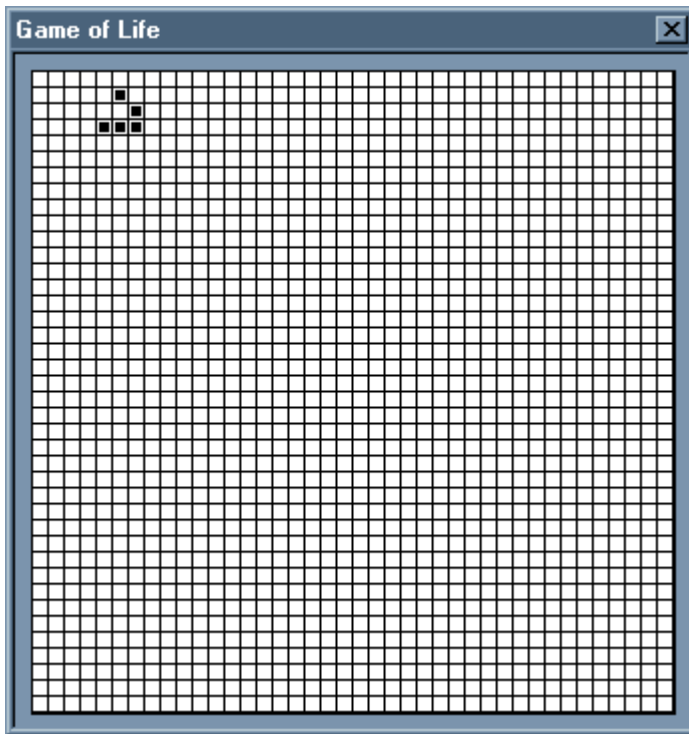
CAMUS is a cellular automata-based music generator. From many different cellular automata algorithms available today, two have been selected for CAMUS, namely Game of Life (invented by John Horton Conway) and Demon Cyclic Space (designed by David Griffeth).

Game of Life

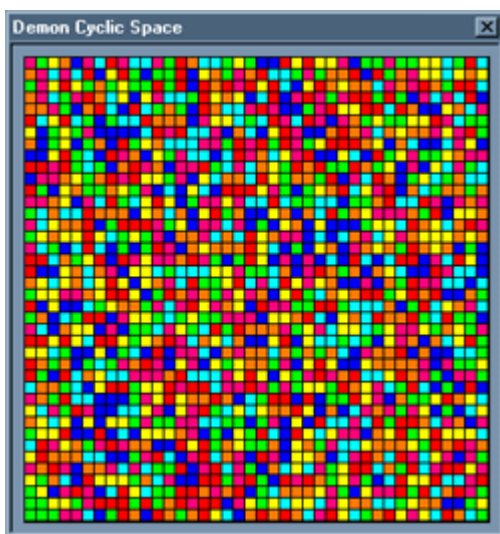
From one tick of the clock to the next, the cells of the Game of Life cellular automaton can be either alive (i.e., black) or dead (i.e., white), according to the following rules devised by Conway:

- if a cell is dead at time t , it comes alive at time $t+1$ if it has exactly 3 neighbours alive;
- if a cell is alive at time t , it comes dead at time $t+1$ if it has fewer than 2 or more than 3 neighbours alive.

These rules are applied simultaneously to all cells of the lattice. An initial configuration of live cells may either grow interminably, fall into cyclic patterns, or eventually die off. The CAMUS implementation of this algorithm enables the musician to design his or her own rules, beyond Conway's original rule.



Demon Cyclic Space



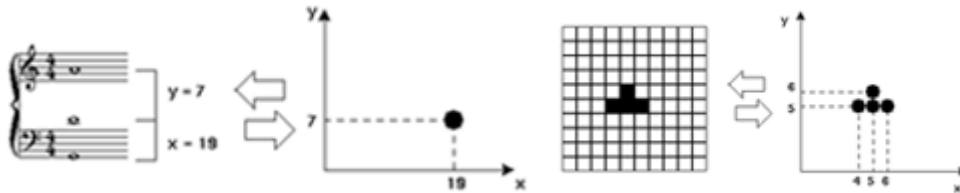
Click on the image

The rules of the Demon Cyclic Space cellular automaton generate miniature worlds of incredible complexity. Initialised as a random distribution of coloured cells, it always end up with stable, angular spirals reminiscent of crystalline growths.

Each of the n possible states for a cell is represented by a different colour and numbered from 0 to $n-1$. A cell that happens to be in state k at one tick of the clock dominates any adjacent cells that are in state $k-1$ meaning that these adjacent cells change from $k-1$ to k . This rule resembles a natural chain in which a cell in state 2 can dominate a cell in state 1 even if the later is dominating a cell in state 0. But, since the automaton is cyclic, the chain has no end and a cell in state 0 dominates its neighbouring cells that are in state $n-1$.

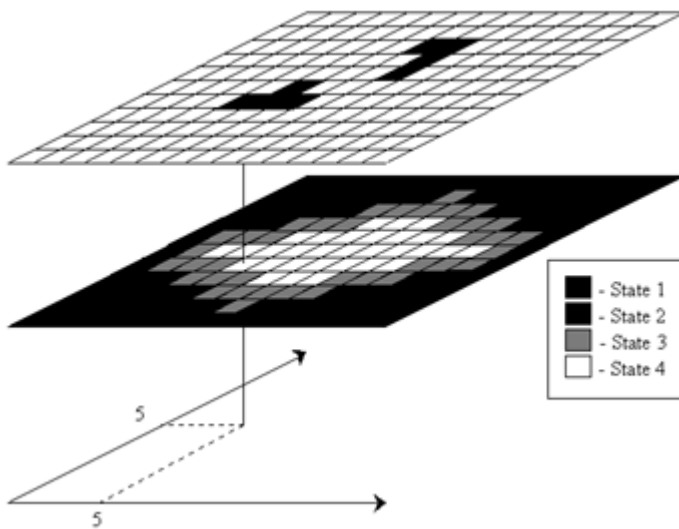
The musical engine of CAMUS

CAMUS uses a Cartesian model in order to represent a triple; that is, a set of three notes. The model has two dimensions, where the horizontal coordinate represents the first interval of the triple and the vertical coordinate represents its second interval.

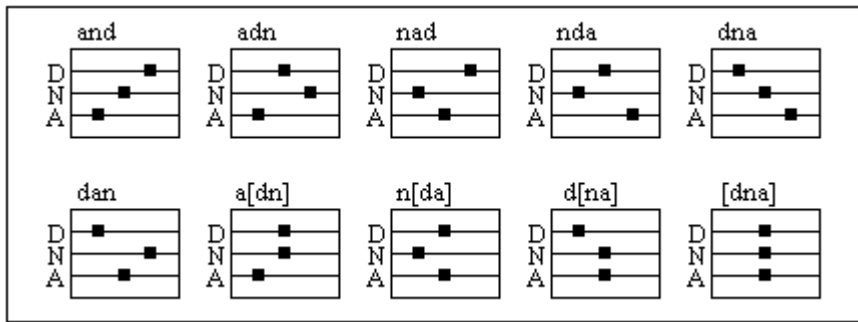


The system uses both automata in parallel to produce music. The Game of Life automaton produces triples and the Demon Cyclic Space automaton determines the "orchestration" of the composition. In this case, each colour corresponds to an instrument (MIDI) designated to perform the notes generated by a specific cell. Each musical cell has its own timing, but the notes within a cell can assume different durations and can be triggered at different times.

To begin the music process, the Game of Life automaton is set up with a starting configuration, the Demon Cyclic Space automaton is initialised with random states, and both are set to run. At each time step, the co-ordinates of each live cell are analysed and used to determine a triple which will be played at the corresponding time in the composition. The state of the corresponding cell of the Demon Cyclic Space is used to determine the orchestration of the piece. This configuration is illustrated below. In this case, the cell in the Game of Life at (5, 5) is alive, and thus constitutes a sonic event (that is, a set of three notes). The corresponding cell in the Demon Cyclic Space is in state 4, which means that the sonic event would be played by instrument number four (e.g., using MIDI channel 4). The co-ordinates (5, 5) describe the intervals in a triple: the fundamental pitch, the note five semitones above the fundamental, and the note ten semitones above the fundamental.





Once the triple for each cell has been determined, the states of the neighbouring cells in the Game of Life are used to calculate the temporal position and duration of each note, according to a set of temporal codes. These codes determine the temporal shape of each triple; the actual values for the trigger and duration parameters are calculated according to a user-defined equation.




Acknowledgments and Further Information

CAMUS started in 1990 and the first version of the program was for the Atari platform. I used this version to compose a number of pieces, including *Entre l'Absurde et le Mystère* (for chamber orchestra) and the second movement of *Wee Batucada Scotica* (string quartet).

 [Entre l'Absurde et le Mystère](#)
 [Wee Batucada Scotica](#)

In 1997, Kenny McAlpine (one of my PhD student at Glasgow University) implemented two slightly different versions of CAMUS for Windows.

 [Click here](#) to hear a short CAMUS-aided tune (MIDI file). For this example, the outcome of CAMUS was subjected to very few editings, mostly involving rhythm and instrumentation; the pitch material remains almost untouched. Another musical example can be found [\[here\]](#).

[Click here](#) for more information about the Atari version of CAMUS.

Kenny's versions of the system, [CAMUS](#) and [CAMUS 3D](#), are available for Windows (tested on Win 95 and 98). A substantial tutorial about CAMUS is available [here](#).

2003©Copyright

All the materials on this Web site are subject to copyright protection. Any form of copying without the express permission of the copyright holder, except such copying as may be permitted under the applicable copyright laws, is strictly prohibited. Redistribution of these materials without the permission of the copyright holder is forbidden, including but not limited to, posting, e-mailing, faxing, archiving in a public database, redistribution via a computer network, or in printed form.