Dynamical Systems and Applications to Music Composition: A Research Report

Kenny B. McAlpine[1,2], Eduardo R. Miranda[2], Stuart G. Hoggar[1]

University of Glasgow

[1] Dept. of Mathematics and [2] Dept. of Music

km@maths.gla.ac.uk, eduardo@music.gla.ac.uk, sgh@maths.gla.ac.uk

Abstract: *In this paper we report the current status of our research work involving the application of dynamical systems to music composition. We introduce the fundamentals of our work and present CAMUS, a system for music composition which uses cellular automata. We then suggest a new engine for CAMUS, which uses fractal techniques and genetic algorithms to extend the capabilities of the system.*

1. Introduction

Fractals are, without a doubt, one of the most exciting developments in recent mathematical knowledge. Not only do they have many practical uses, ranging from image compression through earthquake prediction to the animation of simulated worlds, but they are one of the few pure mathematical forms which have successfully captured the public imagination. One can see fractal images on posters, in advertisements, and at musical gatherings. However, it is not only fractal images which have become associated with certain musical genres, fractal techniques have also been employed in the composition of music. It is with this in mind that we wish to introduce the present paper.

Fractals are just one class of system with which our research is concerned. We wish to consider the more general class of dynamical systems - those which change some feature with time. This includes fractal zooms, cellular automata and neural networks, each of which hold some potential as the basis for both musical and sound generating algorithms.

The paper consists of four sections. The first details work which has previously been carried out in this area; the second is a case study, and will present an outline of the techniques behind the CAMUS system developed by Eduardo Miranda; the third introduces some ideas for extending these results, along with plans for further research, and the final section presents methods for analysis of the techniques described in the first three sections.

2. Background

The use of fractal techniques in music composition began when it was discovered that almost all musical melodies mimic $1/f$ noise ([Clarke & Voss 1975] [Clarke & Voss 1978] [Peitgen & Saupe 1988]). $1/f$ noise is a term which is used to describe any fluctuating quantity in which the 'amount' of frequency, f, varies as $1/f$. Much of the early work with fractal music concentrated on this relationship, but more recently, fractal techniques for both composition and synthesis have focused on self-similarity ([Ames 1982] [Barnsley 1988] [Bolognesi 1983] [Hoggar 1992] [Little 1993] [Peitgen, Jurgens & Saupe 1992] [Peitgen & Saupe 1988]) - a characteristic feature of many fractal forms which describes how they appear similar when viewed under different levels of magnification. There are three different kinds of self-similarity: *exact self-similarity*, in which magnified small parts of the object in question are identical to the whole; *statistical self-similarity*, in which the magnified portion of the image has the same statistical properties as the whole object, and generalised self-similarity, in which scaled copies of the whole figure undergo some transformation. Figure 1 shows a sample of $1/f$ noise, which demonstrates statistical self-similarity. A good example of an exactly self-similar figure is the Sierpinski Gasket (see figure 2), which is made up of identical scaled copies of a triangle. Figure 3 shows a fractal spiral, and a skewed and rotated detail .

Figure 1: 1/f noise and detail

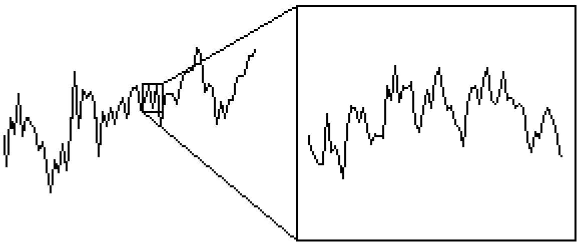


Figure 2: Sierpinski Gasket and detail

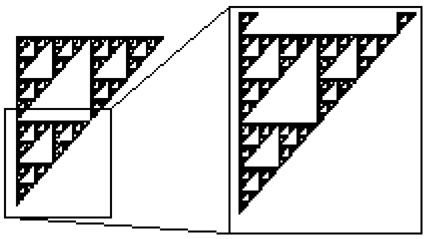


Figure 3: Fractal spiral and detail

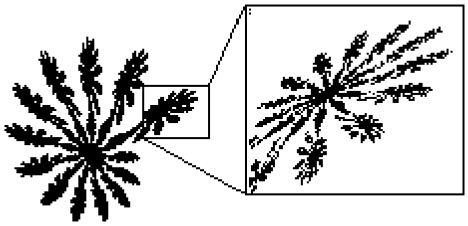
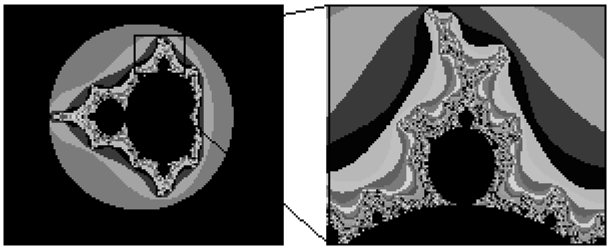


Figure 4: Mandelbrot Set and detail

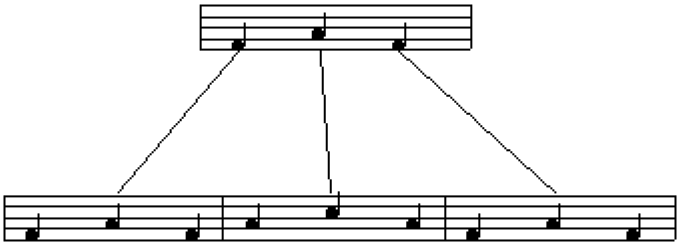


It should be noted that a fractal figure need not exhibit self-similarity, the Mandelbrot set (see figure 4) does not, but it does, however, have detail on every scale. This will be discussed further in a later section.

As a musical device, self-similarity has a great deal to offer - traditional composers have long used devices such as repetition, rondo and variations in their works. Each of these devices can be considered as a form of self-similarity, since they introduce a theme or a specific motif at some point in a work, before re-introducing it, either exactly, or after some transformation, to the listener.

Musical self-similarity can be realised in several ways. For example, three bars of music, each with three beats in a bar would constitute a degree of self-similarity. Alternatively, the self-similarity can be applied to the musical phrases themselves, as is shown in figure 5. Here, a simple three note phrase is repeated three times, with each repetition beginning on a note of the original melody.

Figure 5 - A simple melodic phrase which exhibits self-similarity.



The hierarchical view presented in figure 5 is a useful way to visualise self-similarity. There is a 'master' figure, whose parts give rise to a number of (possibly transformed) copies of the original figure, each of whose parts, in turn, give rise to a number of copies of the copies, and so on. For example, one could readily visualise the next step in the hierarchy for figure 5: we would obtain a 9 bar phrase, with the original three notes mirrored in each bar.

Some caution, however, must be exercised in the degree of self-similarity imposed on the music. If the degree is too high, the piece becomes wholly predictable and ultimately very dull. On the other hand, too little self-similarity leads to randomness and lack of any coherent structure, which can lead to frustration as the human brain tries and fails to find patterns in the music.

Some work has also been carried out on mappings of fractals onto music. However, these mappings have not been particularly interesting, either musically or mathematically, as they have consisted mainly of techniques which associate, say, a certain colour band of the Mandelbrot set with a certain pitch. There are three main problems with this approach. The first is the fact that the Mandelbrot set does not exhibit self-similarity, which leads to the generation of pseudo-random pitches. The second is that the mapping itself is very simplistic, and so the infinitely complex structure contained within the set simply does not translate to the musical output. Finally, the technique essentially maps a static structure (the Mandelbrot set) onto a dynamic structure (music), and as a result, little consideration is given to the temporal development of the music. Possible strategies for countering these problems will be mentioned in a later section.

Another method of composition uses cellular automata as the driving mechanism for the music-generating algorithm ([Miranda 1993] [Miranda 1994]). A cellular automaton is a mathematical model of a dynamical system over which space and time are discrete and all quantities take on discrete values. A cellular automaton is often viewed as an array of elements, referred to as cells, to which we apply some evolution rule which will dictate how the automaton develops in time. Cellular automata have also been used to distribute grains in a granular synthesis model ([Miranda 1995]). This technique seems to have some potential, as the behaviour of a cellular automaton appears to follow much the same pattern as the behaviour of the partials of a sound - either settling down to a steady, cyclic state, or gradually dying out altogether.

In summary, the musical potential of dynamical systems, particularly that of fractal structures, has only been partly realised. Previous work in this area has hinted at the rich sonic material which lies within their confines, but it is only when both the complexity and dynamical behaviour of these systems are harnessed that this musical potential can be explored fully.

3. CAMUS - a Case Study

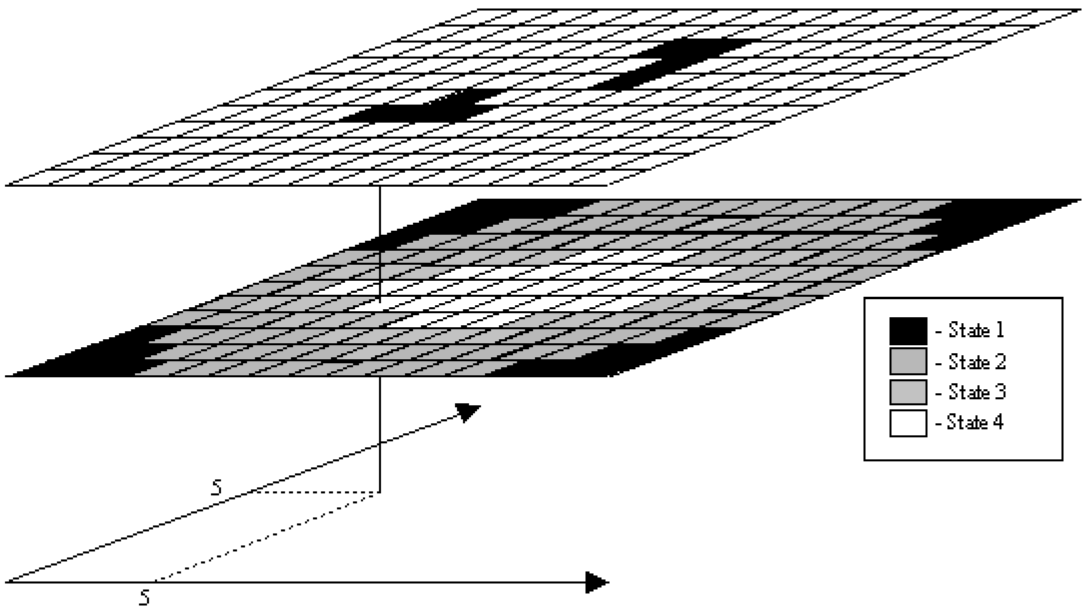
A system called CAMUS (**C**ellular **A**utomata **MUS**ic; [Miranda 1993] [Miranda 1994]) uses the 'Game of Life' and 'Demon Cyclic Space' automata to generate compositions.

The Game of Life automaton consists of an array of $(m \times n)$ cells, which can exist in two states, alive (i.e. 1) or dead (i.e. **0**). The rule which determines the development of the automaton is: *A cell will be alive at timestep* $t + 1$ *if and only if it has precisely* **3** *live neighbours at timestep* t. The Demon Cyclic Space automaton is an array of $(m \times n)$ cells which can exist in k states. The evolution of the automaton is determined by: *A cell which is in state* j *at timestep* t *will dominate any neighbouring cells which are in state* $j - 1$*, so that they increase their state to* j *at timestep* $t + 1$.

The CAMUS system also allows these rules to be tailored slightly, to allow for variations in the composition algorithm.

To begin the composition process, the Game of Life automaton is set up with a starting configuration, the Demon Cyclic Space automaton is intialised 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 *triad*[1] which will be played at the corresponding time in the composition. The state of the corresponding cell of the Demon Cyclic Space automaton is used to determine the *orchestration*[2] of the piece. This configuration is demonstrated in figure 6. In this case, the cell in the Game of Life at **(5, 5)** is alive, and thus constitutes a sonic event. 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 number 4). The co-ordinates **(5, 5)** describe the intervals in a triad: the fundamental pitch, the note five semitones above the fundamental, and the note ten semitones above the fundamental.

Figure 6 - Configuration for each timestep of the cellular automata used in CAMUS



This configuration, of the points in the euclidean plane being used to determine musical intervals is known as the *von Neumann Music Space* ([Miranda 1993] [Miranda 1994]). Once the triad for each cell has been

[1] In this paper, the term 'triad' is used to describe any set of three (not necessarily distinct) notes.

[2] Similarly, the term 'orchestration' is taken to mean which instrument 'plays' the cells.

determined, the states of the neighbouring cells in the Game of Life are used to calculate the temporal position and duration of each note as follows: Suppose we wish to determine the temporal positioning of the cell $(\boldsymbol{i}, \boldsymbol{j})$. We can construct a set of values from the states of the neighbouring cells - the value being **1** if the cell is alive and **0** if it is dead:

a = cell $(i, j - 1)$	b = cell $(i, j + 1)$	c = cell $(i + 1, j)$	d = cell $(i - 1, j)$
m = cell $(i - 1, j - 1)$	n = cell $(i+1, j + 1)$	o = cell $(i + 1, j - 1)$	p = cell $(i - 1, j + 1)$

We then form the four four-bit words, ***abcd***, ***dcba***, ***mnop*** and ***ponm***. Next, we perform the bitwise inclusive OR operation, '|' , to form two four-bit words, ***Tgg*** and ***Dur***:

$$Tgg = abcd \mid dcba$$
$$Dur = mnop \mid ponm$$

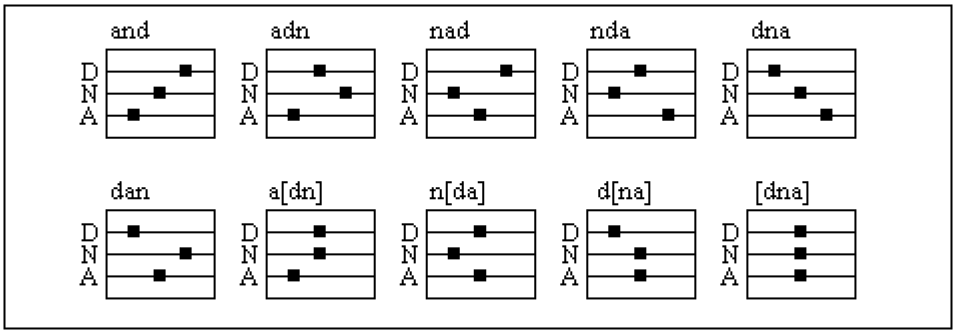
From ***Tgg***, we derive the note trigger information, and from ***Dur***, the note duration information. For each relevant four-bit word, we associate a code, known as an **AND** (cellul**A**r ge**N**etic co**D**e). Note that the **AND** code is distinct from the bitwise AND operator, and is assigned as follows:

0000 : a[dn]	**0001: [dna]**	**0010 : adn**	**0011 : dna**	**0101 : and**
0110 : dan	**0111 : nad**	**1001 : d[na]**	**1011 : nda**	**1111 : n[da]**

Each of the three letters in the **AND** codeword is used to represent a note in the triad, with **a** denoting the lower pitch, **n**, the middle pitch, and **d** the upper. The square brackets are used to indicate that the note events contained within that bracket occur simultaneously.

We associate each codeword with a particular temporal configuration, as shown in figure 7.

Figure 7 - Ten different temporal codes



These codes determine the *temporal shape* of each triad, the actual values for the trigger and duration parameters are calculated according to a user-defined equation.

The final stage in the CAMUS compositional process is the application of articulations. These are simply user-defined control parameters, which specify a speed, dynamic level, and 12 pitch sequence which is performed at each timestep. Up to 22 such articulations can be defined, giving a fairly wide scope for compositional development. CAMUS has been successfully used to compose a number of compositions, including a prize-winning piece, *Entre l'Absurde et le Mystère*, by Eduardo Miranda.

4. Extending the capabilities of CAMUS.

We now describe a new compositional algorithm, which, although drawing on certain ideas behind the CAMUS system, differs substantially in the control system. The system utilises two properties of the fractal classes known as the Mandelbrot set, and Julia sets.

The Mandelbrot set can be considered as a graphical representation of the behaviour of the complex plane according to the iterative equation $z_{i+1} = z_i^2 + \boldsymbol{c}$ where $\boldsymbol{c}$ is a complex constant. For each $\boldsymbol{c}$ in the plane, the system is initialised with $z_0 = \mathbf{0}$, and iterated. This results in the following possible outcomes:

(*i*) $\quad |z_n| \to \infty$ as $\boldsymbol{n} \to \infty$.

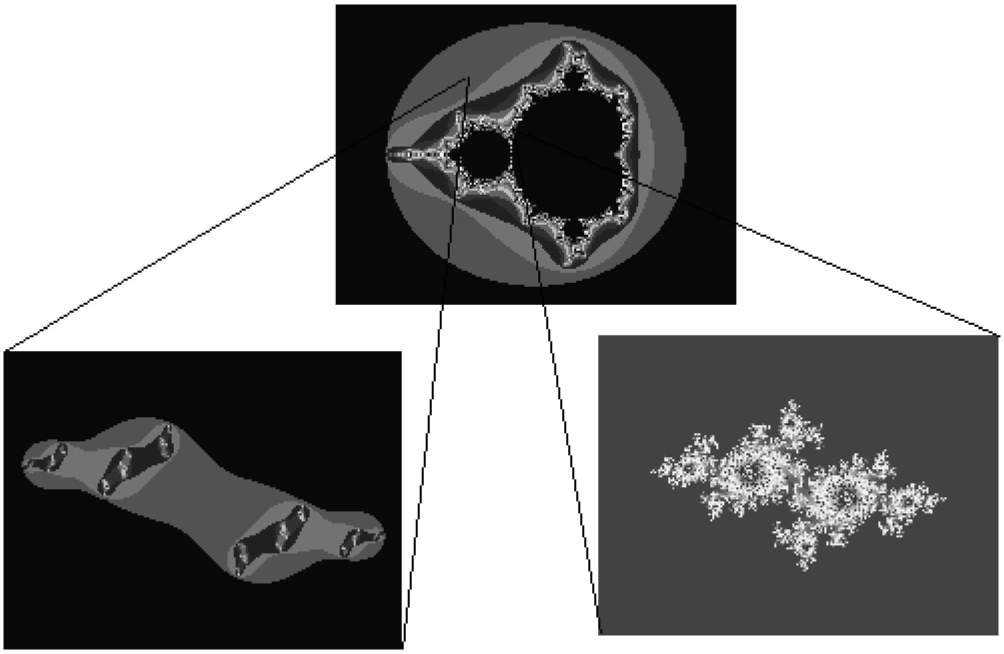
(*ii*) $\quad |z_n|$ remains bounded as $\boldsymbol{n} \to \infty$.

Each $\boldsymbol{c}$ is assigned a colour according to the number of iterations required to send the point to infinity, or a default colour (usually black) if the point does not increase without bound (in practice, we assign a threshold value, $\boldsymbol{T}$, and a maximum number of iterations, ***MI***, and stop the iterative process when either $|z_n| \geq \boldsymbol{T}$, or when $\boldsymbol{n} > \boldsymbol{MI}$). The Mandelbrot set is then the set of all values of c which are coloured black. The area of interest occurs within the region bounded by the lines $x = -2.5$ and $x = 1.5$ on the real axis, and $y = -1.5i$ and $y = 1.5i$ on the imaginary axis. At, and near the boundary of the Mandelbrot set, there is described a structure of inconceivable complexity, with curves which spiral onwards for eternity, and detail at every scale.

Julia sets are determined in much the same way as the Mandelbrot set, using the same iterative formula, but in this case, the complex parameter, c, remains fixed. After each iteration, the initial value, z_0, is altered. This gives rise to the Julia set, J_c, which is defined to be the boundary formed between the set of all initial values, z_0, for which $|z_n| \to \infty$ as $n \to \infty$, and the set of all initial values for which $|z_n|$ remains bounded.

When c is controlled from within the confines of the Mandelbrot set, say by the user clicking on a point, which is then passed to the Julia algorithm, one can see a striking visual correlation between the region which was clicked, and the type of Julia set which is obtained (see figure 8). Parameters taken from the exterior of the Mandelbrot set result in non-connected sets, whilst parameters taken from within the set itself give rise to connected Julia sets.

Figure 8 - The Mandelbrot set and two Julia sets with respective regions



Like the Mandelbrot set, Julia sets exhibit detail at every scale, and it is this feature along with the relationship between the two on which the composition system is based.

To begin with, the user is presented with a window showing the Mandelbrot set. The user then draws a path on the screen, specifying a series of Julia sets, which will be calculated in sequence. Next, a further path, which may involve zooming in or out of the Mandelbrot set, is drawn. This defines a sequence of regions of the Mandelbrot set, to be calculated in parallel with the Julia sets. The user specifies the number of timesteps required to complete the composition, along with other composition specific parameters, such as the number of instruments and their timbres. The sets are then mapped onto the von Neumann Music Space, with each Julia colour band corresponding to a different instrument, and points on the boundary[3] of the Mandelbrot set contributing to the notes of the composition. A set of articulations, similar to those of the CAMUS system are then used to determine the actual notes and dynamics of the piece. Alternatively, some stochastic process could be used, with user-defined probability tables controlling the evolution of the composition.

4.1 Using a genetic algorithm to control composition parameters

An alternative method for controlling the development of the composition parameters can be derived from a computation technique known as a genetic algorithm. This is an evolutionary algorithm, that is one which is based on the various biological phenomena associated with evolution, and operates on a set of binary codewords. There are four basic types of operation which can be performed on the codewords: selection;

---

[3] Here, the 'boundary' of the Mandelbrot set is taken to mean the quantised graphical boundary.

crossover; mutation and inversion. We shall consider only the first three, since inversion can be derived from these. The crossover operation exchanges information between a pair of codewords. Mutation alters the value of a single bit in a codeword. Typically, the genetic algorithm is used as an optimisation technique, and so the selection operation is used to find the 'best' possible codewords for some predetermined criterion. However, for compositional purposes, no such optimisation is necessary, since the operations are used solely for the development of the compositional parameters. A typical parameter evolution step is shown in figure 9.

Figure 9 - A typical parameter development step

Explanations	Codewords
Consider a set of ***n*** codewords which represent the values of some musical parameters.	$\boldsymbol{C}_1$: 1 1 0 1 0 1 1 0 $\boldsymbol{C}_2$: 1 0 0 1 0 1 1 1 ... $\boldsymbol{C}_n$: 0 1 0 0 1 0 0 1
Selection: Codewords are chosen to undergo evolution according to some stochastic mechanism (*c.f.* Darwin's theory of evolution - chance mutation leading to population development (Darwin, 1859)).	$\boldsymbol{C}_2$: 1 0 0 1 0 1 1 1 $\boldsymbol{C}_7$: 1 1 0 0 0 1 0 1 $\boldsymbol{C}_{11}$: 0 1 1 1 1 0 0 1
Crossover: Some portion of a pair of codewords (in this case, $\boldsymbol{C}_7$ and $\boldsymbol{C}_{11}$) is exchanged at the highlighted position randomly specified, producing the two offspring $\boldsymbol{C}_7'$ and $\boldsymbol{C}_{11}'$, whose values are then assigned to $\boldsymbol{C}_7$ and $\boldsymbol{C}_{11}$. This is applied to some predetermined section of the population, specified by the crossover rate.	$\boldsymbol{C}_2$: 1 0 0 1 0 1 1 1 $\boldsymbol{C}_7$: 1 1 0 0 0 *1 0 1* $\boldsymbol{C}_{11}$: 0 1 1 1 1 *0 0 1* $\boldsymbol{C}_7'$: 1 1 0 0 0 *0 0 1* $\boldsymbol{C}_{11}'$: 0 1 1 1 1 *1 0 1*
Mutation: The bit values of some codewords are inverted at a mutation rate of 1 - 5%. The value that has been changed as and example is highlighted by an underline.	$\boldsymbol{C}_2$: 1 0 0 1 0 1 1 1 $\boldsymbol{C}_7$: 1 1 **1** 0 0 0 0 1 $\boldsymbol{C}_{11}$: 0 1 1 1 1 1 0 1

The genetic algorithm can be utilised as a parameter development technique as follows: firstly, each parameter is assigned a binary codeword according to its initial value. At each timestep, the genetic algorithm is performed on the set of codewords as illustrated in figure 9 above, leading to population development.

4.2 Alternative mappings from dynamical systems to music

Finally, we consider the effect of many different mappings from both cellular automata and fractals onto music. The technique of mapping coordinates to musical triads through the von Neumann Music Space has already been described, but some work must be undertaken to evaluate its effectiveness both musically and computationally with respect to other mappings. Some examples of mappings which have yet to be evaluated with these systems are: mapping cells onto a discrete two-dimensional space defined by pitch vs. a musical parameter (see figure 10), and using the states of cells to determine probability tables from a set held in memory. Once a probability table has been established for an individual cell, a random selection routine can then be used to select the actual note values and musical parameters with the cell (see figure 11).

Figure 10 - An alternative two-dimensional space for use in mapping dynamical systems to music

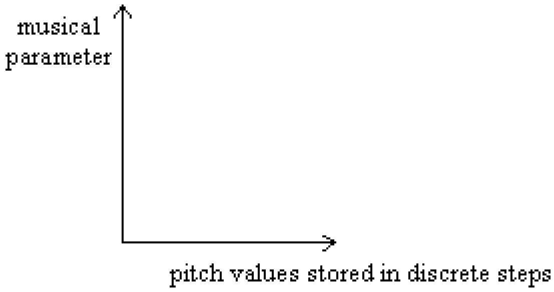
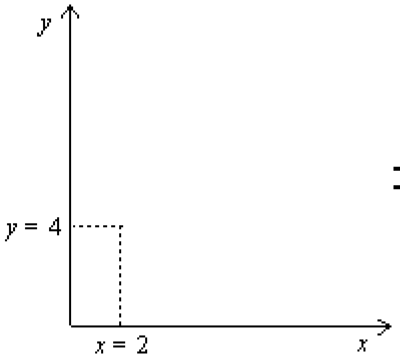


Figure 11 - A cell at position (2 , 4) determines a probability table associated with that note. The actual note values and parameters can then be calculated using a random selection technique



⇒

Probability table stored at array position (2 , 4)

Note	*Dynamic level*	*Probability*
C#	*p*	0.2
E	*mf*	0.35
Ab	*pp*	0.4
Rest	-	0.05

5. Evaluation Techniques

Evaluating compositional methods can present almost as many problems as defining and implementing the techniques themselves. There are two levels on which we can approach this. The first is to perform an analysis of how well the technique in question performs. In particular, we should be concerned with the following:

How efficiently does the technique in question utilise its resources? This point is very important if the user is to hear real-time feedback of compositions, particularly if the system is being used in a live performance, or with other composition systems. Inefficient use of resources could lead to glitches in the output, possibly throwing other players out of step.

How intuitive is the system? Is it possible for a new user to pick up the basics of the system and produce compositions straight away, or is there a steep learning curve which must first be surmounted? How effective is the user interface? These questions are all fundamental to the usefulness of any system. An interface which is too complex may alienate many users before they become familiar enough with it to obtain pleasing results. On the other hand, a system which is too simplistic may mean that experienced users become frustrated at being 'pampered' through many stages of introduction and setup instead of getting on with the job in hand. There is a fine balance to be struck here, and care must be taken to get it right. Algorithmic composition systems are, after all, tools to aid creativity and few things inhibit the creative process more than struggling with a poorly designed interface.

How robust is the system? How well behaved is it? A good system will be both *well behaved*, that is it will produce output which is within the human hearing range and files which store only the information needed to reproduce the music, and *robust*, that is impervious to erroneous input by the user and not prone to crashing or causing other programs to crash. No program can be guaranteed to be completely error-free. However, with extensive testing under many different operating environments and with many types of input, the system can at least be declared reasonably stable. Tidy programming techniques and a well-designed system should ensure good-behaviour.

How versatile is the system? Do all the compositions it helps create sound the same, or can the system produce music in a number of different styles? There is nothing wrong with designing a system to produce one particular style of music, nor is there anything wrong with a system which can produce many different styles of music. The key issue here is that the system can produce a variety of compositions. System versatility is also closely related to the mapping techniques used in the composition algorithm. Versatility can often be improved by offering the user a variety of techniques which can be applied to the problem in hand.

The second approach to evaluation of a composition system is to consider the artistic merits of the musical output generated by the technique, and it is here that difficulties arise. The problem is that music is, by its very nature, subjective, whereas rigorous analysis demands a high degree of objectivity. Clearly, a balance must be struck, so that new techniques are not dismissed out of hand due to personal tastes, but nor are subtle artistic details omitted from consideration because of a very clinical approach to analysis

6. Conclusion

Dynamical systems hold great potential as a means for driving musical composition. It is hoped that future research into this particular area of music technology will unleash some of this potential, producing not only interesting composition techniques, but also interesting and musically pleasing compositions.

Acknowledgements

Many thanks to the Carnegie Trust for the Universities in Scotland, who generously provided the funding which has enabled this research. Thanks also to the University of Glasgow for providing both a research position and the facilities to undertake research in this field.

References

[Ames 1982] C. Ames, *Crystals: Recursive Structures in Automated Composition*, Computer Music Journal 6(3): 46 - 64, 1982.

[Barnsley 1988] M. Barnsley, *Fractals Everywhere*, Academic Press, 1988.

[Bolognesi 1983] T. Bolognesi, *Automatic Composition: Experiments with Self-Similar Music*, Computer Music Journal 7(1): 25 - 36, 1983.

[Chapman, Clarke, Smith & Archbold 1996] D. Chapman, M. Clarke, M. Smith & P. Archbold, *Self-Similar Grain Distribution: A Fractal Approach to Granular Synthesis*, ICMC Proceedings: 212 - 213, 1996.

[Clarke & Voss 1975] J. Clarke & R. F. Voss, *1 /f Noise in Music and Speech*, Nature 258: 317 - 318, 1975.

[Clarke & Voss 1978] J. Clarke & R. F. Voss, *1/f Noise in Music: Music from 1/f Noise*, Journal of the Acoustical Society of America 63: 258 - 263, 1978.

[Hoggar 1992] S. G. Hoggar, *Mathematics for Computer Graphics*, Cambridge University Press, 1992.

[Little 1993] D. Little, *Composing With Chaos: Applications of a New Science for Music*, Interface 22: 23 - 51, 1993.

[Miranda 1993] E. R. Miranda, *Cellular Automata Music: An Interdisciplinary Project*, Interface 22: 3 - 21, 1993.

[Miranda 1994] E. R. Miranda, *Music Composition Using Cellular Automata*, Languages of Design 2: 105 - 117, 1994.

[Miranda 1995] E. R. Miranda, *Granular Synthesis by Means of a Cellular Automaton*, Leonardo 28(4): 297 - 300, 1995.

[Peitgen & Saupe 1988] H. Peitgen & D. Saupe, *The Science of Fractal Images*, Springer-Verlag New York, Inc, 1988.

[Peitgen, Jurgens & Saupe 1992] H. Peitgen, H. Jurgens & D. Saupe, *Fractals for the Classroom: Part Two - Complex Systems and Mandelbrot Set*, Springer-Verlag New York, Inc., 1992.

[Wolfram 1984] S. Wolfram, *Universality and Complexity in Cellular Automata*, Physics 10: 1 - 35, 1984.