CHAPTER 2
On Unconventional Computing for Sound and Music

Eduardo R. Miranda, Alexis Kirke, Edward Braund and Aurélien Antoine

Abstract
Advances in technology have had a significant impact on the way in which we produce and consume music. The music industry is most likely to continue progressing in tandem with the evolution of electronics and computing technology. Despite the incredible power of today's computers, it is commonly acknowledged that computing technology is bound to progress beyond today's conventional models. Researchers working in the relatively new field of Unconventional Computing (UC) are investigating a number of alternative approaches to develop new types of computers, such as harnessing biological media to implement new kinds of processors. This chapter introduces the field of UC for sound and music, focusing on the work developed at Plymouth University's Interdisciplinary Centre for Computer Music Research (ICCMR) in the UK. From musical experiments with Cellular Automata modelling and *in vitro* neural networks, to quantum computing and bio-processing, this chapter introduces the substantial body of scientific and artistic work developed at ICCMR. Such work has paved the way for on-going research towards the development of robust general-purpose bio-processing components, referred to as biomemristors, and interactive musical biocomputers.

2.1 Introduction

Originally, the term 'computer' referred to a person or groups of people who followed sets of rules to solve mathematical or logic based problems. It was not until the beginning of the 20th century that it began to refer to a machine that performs such tasks. In the 1930s Alan Turing formalised the behaviour of these machines to create a theoretical model of a computer: the Turing Machine (Turing 1936). Shortly after this, in the 1940s John von Neumann developed a stored-program computing architecture (Aspray 1990). Whereas a Turing Machine is a theoretical machine invented to explore the domain of computable problems mathematically, von Neumann's architecture is a scheme for building actual computing devices. These two seminal works are considered the precursors of today's commercial computers, with their underlying concepts remaining relatively unchanged. However, we should note that the idea of developing programmable calculating machines had existed before Turing and von Neumann's works. Notable examples are Charles Babbage's various attempts at building mechanical calculating engines in the early 1800s (Swade 1991).

During the past 80 years or so, what we consider to be conventional computation has advanced at a rapid speed. Yet, despite the incredible power of today's computers, it is commonly acknowledged that computing technology is bound to progress beyond today's conventional models. For instance, D-Wave Systems, in Canada, has recently started to sell the world's first commercial quantum computer. This technology, however, is unaffordable for the time being and it is likely to remain so for a while.

Researchers working in the relatively new field of Unconventional Computing (UC) are developing a number of alternative approaches for implementing new processing devices; these include harnessing chemical and biological media, and understanding the immense parallelism and non-linearity of physical systems. Notable experiments have been developed to demonstrate the feasibility of building computers using reaction-diffusion chemical processors (Adamatzky et al. 2003) and biomolecular processors exploring the self-assembly properties of DNA (Shu et al. 2015). The rationale here is that natural agents (biological, chemical, etc.) would become components of the design rather than sources of inspiration to implement abstract models for software simulation. For instance, instead of modelling the functioning of neuronal networks for implementing machine learning algorithms, the UC approach is looking into harnessing networks of real brain cells to implement such algorithms. Please refer to Chapter 1 of this volume for a comprehensive introduction to the field of UC.

With respect to music, computers and music technology have developed almost in tandem. Back in the late 1940s, scientists of Australia's Council for Scientific and Industrial Research (CSIR) installed a loudspeaker on the CSIR Mk1 computer, which was one of the first four or five electronic computers built in the world at the time. Programmers would use the loudspeaker to play a sound at the end of their program to notify the operator that the machine had halted. Not surprisingly, a mathematician with a musical upbringing, Geoff Hill, had the brilliant idea of programming this computer to play back an Australian folk tune (Doornbusch 2004). This is allegedly the first ever piece of computer music. Since this early interdisciplinary endeavour, advances in Computer Science have had a significant impact on the way music and audio media is produced and consumed. Therefore, it is likely that future developments in Computer Science will continue to impact the music industry.

In Computer Music, there is a tradition of experimenting with emerging technologies, but until very recently, developments put forward by the field of UC have been left largely unexploited. This is most probably so due to a myriad of constraints, including the field's heavy theoretical nature, and the costly investment required to develop a laboratory and hire specially trained personnel to build prototypes and conduct experiments. Nevertheless, research into unconventional modes of computation has been building momentum and the accessibility of prototypes for the computer music community has been widening. This increased accessibility has enabled computer musicians to begin exploring the potential of emerging UC paradigms. For instance,

Miranda and Kirke have recently composed pieces of music using respectively a bespoke biocomputer (introduced below) and the D-Wave machine (see Chapter 5).

In the meantime, and given the abovementioned constraints, a realistic approach to initiate research into UC for sound and music is to work with modelling. The notion of simulating aspects of UC on conventional digital computers may sound preposterous, but as we will demonstrate below, it is a sensible and effective approach to get started. After all, as Susan Stepney discussed in Chapter 1, as well as developing hardware, UC research also involves the development of non-classical algorithms inspired by the way physical and biological processes work.

2.2 *Olivine Trees*: Musical Experiments with Cellular Automata

Cellular Automata (CA) modelling is a valuable tool to simulate aspects of biological, chemical and physical systems, which have been explored in UC. A typical example is reaction-diffusion chemical reactions (Adamatzky et al. 2003). “In the strict sense of the term, reaction-diffusion systems are systems involving constituents locally transformed into each other by chemical reactions and transported in space by diffusion. They arise, quite naturally, in chemistry and chemical engineering but also serve as a reference for the study of a wide range of phenomena encountered beyond the strict realm of chemical science such as environmental and life sciences.” (Nicolis and De Wit 2007).

Back in 1992, Miranda developed a CA model of a reaction-diffusion system to implement a sound synthesiser on a Connection CM-200 parallel computer at Edinburgh Parallel Computing Centre (Miranda et al. 1992; Miranda 1995).

A cellular automaton is normally implemented on a computer as an array (one-dimensional CA) or as grid (two-dimensional CA) of cells. Every cell can exist in a defined quantity of states, which are normally represented as integer numbers and displayed on the computer screen by colours. To enable the model to evolve, transition rules are applied to the cells informing them to change state according to state of their neighbourhood; the changes take place synchronously to all cells with the beat of an imaginary clock. Typically these rules remain the same throughout the model, but this is not necessarily the case. Initially, at time $t = 0$, each cell is assigned its starting state. The model can then produce a new generation ($t = 1$) of the grid by applying the defined rules. This process can continue for an infinite amount of generations.

Figure 1 illustrates a simple one-dimensional cellular automaton. It consists of a line of thirty cells, each of which can have a value of zero or one, which are represented by the colours white or black. In this example, there are eight transition rules that are shown above the grid. For example, rule number 6 (the sixth from the left) states that if a cell is equal to one (black) and both its neighbours are equal to zero (white) at row t, then this cell’s value will remain equal to one (black) at the next time-step ($t + 1$, one

row down). Note that in order to apply the rules to all cells simultaneously, the algorithm considers that the first and the last cells of the line are connected in a virtual loop: the left side neighbour of the first cell (counting from left to right) is the last cell of the row.

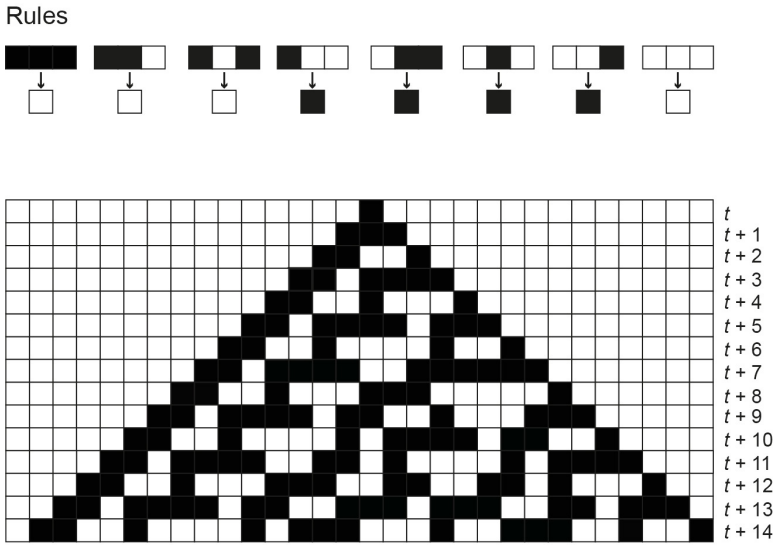

Figure 1: An example of a one-dimensional cellular automaton.

In the case of two-dimensional CA the transition rules take into account the eight nearest neighbours of each cell (Figure 2). In order to apply the transition rules one needs to consider that the grid of cells forms a doughnut-shaped object, where the right edge of the grid wraps around to join the left edge and the top edge wraps around to join the bottom edge. However, the grid is often displayed flat on a computer screen.

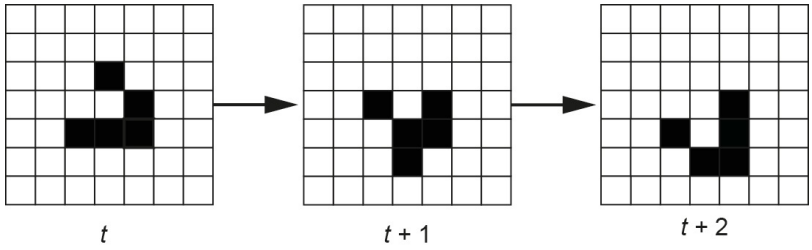

Figure 2: An example of a two-dimensional cellular automaton.

The examples above are of CA with cells that can exists as either zero (represented as white) or one (represented as black). The cells of Miranda's reaction-diffusion model can have values other than only zeroes and ones. By way of introduction, let us consider this automaton as a grid of cells, each of which represents a simple electronic circuit characterised by a varying internal voltage. At a given instant, each of these cells can be in any one of the following states: fired, quiescent and depolarized, depending

on the value of their internal voltage V at a certain time t. In addition to V, the automaton's transition rules take into consideration the following variables:

- The values of $R1$ and $R2$, which represent the resistance values of a potential divider
- The value of C, which represent an electronic capacitor
- The value of Max, which is a threshold value

A cell interacts with its neighbours through the flow of electric current between them. If a cell's internal voltage V is equal to zero, then this cell is in a quiescent state. As the transition rules (see below) are applied, the internal voltages of the cells tend to increase. Once the value of V for a certain cell reaches 1 then this cell becomes depolarized. However the cells have a potential divider, defined by the values of $R1$ and $R2$, which is aimed at creating a global resistance against depolarization over the whole network. Also, they have an electronic capacitor C, whose value regulates the rate of this increase. The values of $R1$, $R2$, C and Max are identical for all cells on the grid.

Depolarized cells go through a period of increasing depolarization gradients until their V remains below a pre-defined maximum threshold value Max. When the value V of a cell reaches the maximum threshold Max, then it fires and in the next time step it becomes quiescent again: that is, V becomes equal to zero. Before we look at the transition rules, let us establish that:

- The voltage value V of cell n at time t is notated as cell(n, t) = V
- F is the number of fired neighbouring cells (i.e., cells with $V = Max$)
- D is the number of depolarized neighbouring cells
- S is the sum of the values V of all neighbours

The transition rules are as follows:

Rule 1: if cell(n, t) = 0
then cell(n, t+1) = int(($F \div R1$) + ($D \div R2$))

Rule 2: if cell(n, t) > 0 and cell(n, t) < Max
then cell(n, t+1) = int(($S \div F$) + C)

Rule 3: if cell(n, t) = Max
then cell(n, t+1) = 0

As an example, let us consider the case of the cell n in co-ordinates (x, y) shown on grid on the left hand side of Figure 3. Let us assume that the values of Max, $R1$, $R2$ and C were pre-defined as follows: Max = 4, $R1$ = 8.5, $R2$ = 5.2 and C = 3, respectively. In this example, cell(n, t) = 0. Therefore the first rule applies. There are 3 fired neighbours

(i.e., with $V = Max$) and 4 depolarized ones, that is: $F = 3$ and $D = 4$, respectively. Therefore, the new value for this cell is calculated as follows:

cell(n, t+1) = int((3 ÷ 8.5) + (4 ÷ 5.2))
cell(n, t+1) = int(0.359 + 0.769)
cell(n, t+1) = int(1.128)
cell(n, i+1) = 1

The cell becomes depolarized: its new voltage at time t+1 is depicted on the grid on the right hand side of Figure 3.

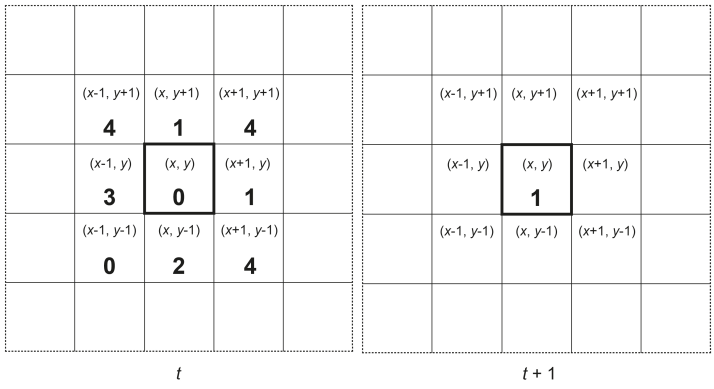

Figure 3: An illustration of the application of transitions rules to 1 cell on the cellular automaton.

Before running the automaton, one defines the values for $R1$, $R2$, C and Max. Then, the system initializes the cells with random values for V, ranging from 0 to Max. The automaton is displayed as a grid of coloured squares, with different colours corresponding to different states. To begin with we see a wide distribution of different colours on the grid, as shown on the grid on the top left hand side of Figure 4. As the automaton runs, the image tends to evolve towards oscillatory cycles of patterns, representing reaction and diffusion of cell states.

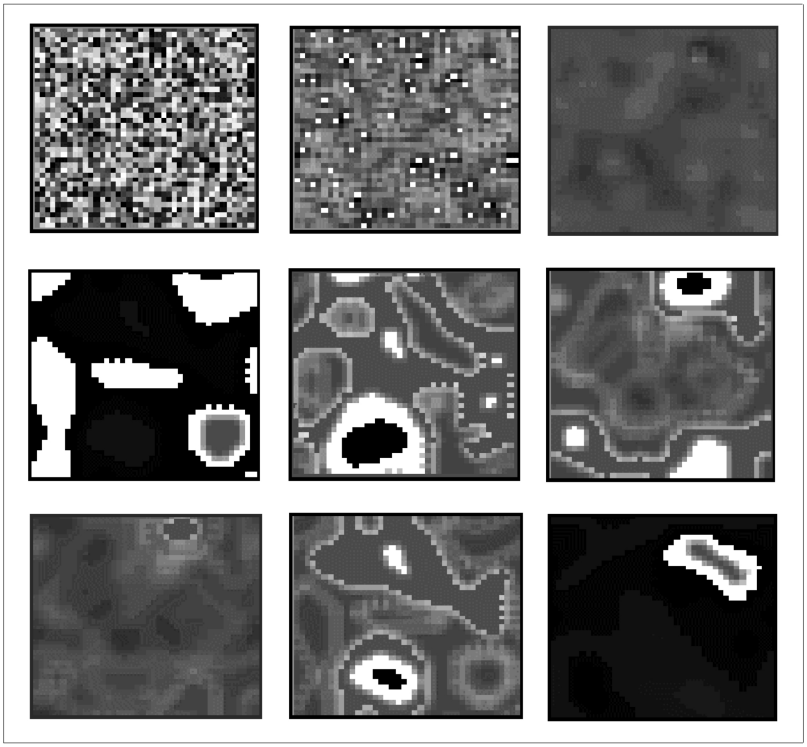

Figure 4: Various stages of the reaction-diffusion simulation.

The system implements a synthesis technique referred to as granular synthesis. Granular synthesis works by generating a rapid succession of very short sound bursts (e.g., 50 milliseconds long each), referred to as sound grains (Miranda 2002). Each of these grains represents the entire automaton's grid at the respective refresh point (Figure 7); in other words, each cycle of the automaton (*t*, *t* + 1, *t* + 2, …), produces a sound grain.

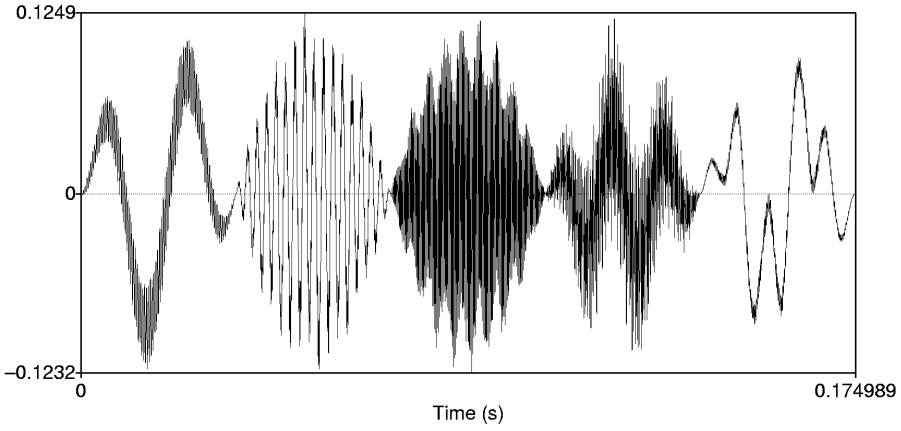

Figure 5: A granular sound comprising a succession of 5 grains.

A sound grain is a composite sound on its own right, comprising a number of partials, each of which is synthesized by a sine wave oscillator (Figure 5). The synthesiser is composed of a number of such oscillators. Each of them requires a frequency value in order to produce the respective partial. The system translates the voltage values *V* of the automaton's cells into frequency values for these oscillators.

The standard procedure to visualize the behaviour of a cellular automaton on a computer screen is to associate the cells' values with a colour, but in this system the values are also associated to different frequency values. For example, the voltage value corresponding to the colour yellow could be associated with 110Hz, the colour red with 220Hz, blue with 440Hz, and so on. These associations are arbitrarily defined and different associations will produce different sounds.

The automaton's grid is divided into smaller uniform sub-grids of cells and each of these sub-grids is associated to an oscillator (Figure 6).

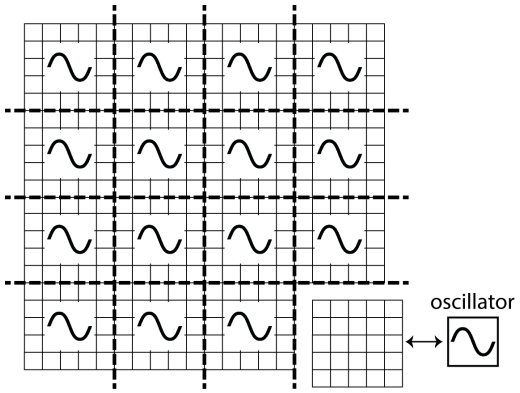

Figure 6: Sub-grids of cells are associated to different oscillators.

At each cycle of the automaton, the system calculates frequency values for the oscillators, which simultaneously synthesize partials that are added together in order to produce the respective sound grain (Figure 7). Figure 6 depicts an example of a grid of 400 cells divided into 16 sub-grids of 25 cells each; each sub-grid is associated to a different oscillator. In this case, the system would synthesize grains with spectra composed of 16 partials each.

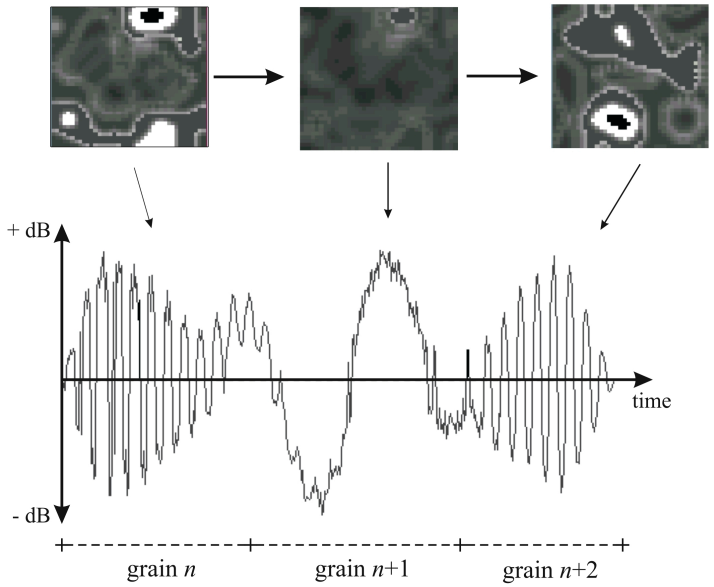

Figure 7: At each refresh point of the automaton a sound grain is synthesised.

The frequencies are calculated by taking the arithmetic mean over the frequencies associated to the values (or colours) of the cells of the respective sub-grids. For example, let us consider an hypothetical case where each oscillator is associated with a sub-grid of 16 cells, and that at a certain refresh point, 3 cells correspond to 55Hz, 2 to 110Hz, 7 to 220Hz and the remaining 4 to 880Hz. In this case, the mean frequency value for this oscillator will be 340.31Hz.

In 1993 Miranda composed, *Olivine Trees*, an electroacoustic piece of music using this system, which is believed to be the first ever piece of parallel computer music. The work effectively explores two veins of UC: a reaction-diffusion system and parallel processing. A recording of *Olivine Trees* is available on SoundCloud (Miranda 1994).

2.3 *Cloud Chamber*: Musical Experiments with Particle Physics

Research towards building a quantum machine that can fully embody computational models based on quantum properties of superposition and entanglement has been making steady progress. However, such machines are not generally available as we write this chapter. Nevertheless, it is possible to develop musical experiments in order to get started with particle physics, gain hands-on experience and prepare the ground for future work. For instance, Kirke championed the *Cloud Chamber* project, which was helpful to test musical ideas and prepare the ground to secure partnerships to develop a musical composition using a D-Wave computer located at the University of Southern California, in the USA (Chapter 5).

Cloud Chamber is also the title of Kirke's duet for violin and subatomic particles. The Interdisciplinary Centre for Computer Music Research (ICCMR) team developed a system for *Cloud Chamber* that renders the behaviour of atomic particles into sounds, which accompanies a solo violin live on stage. Here the violin controls an electromagnetic field system, which influences the way in which the particle tracks behave.

A diffusion cloud chamber was used to create a volume of supersaturated alcohol vapour that condenses on ions left in the wake of charged particles. This is accomplished by establishing a steep vertical temperature gradient with liquid nitrogen. Alcohol evaporates from the warm top region of the chamber and diffuses toward the cold bottom. The gravitationally stable temperature distribution permits a layer of supersaturation near the chamber bottom. Charged particles passing through the supersaturated air at close to the speed of light leave behind numerous ions along each centimetre traversed. In the absence of a radioactive source, most events observed in the cloud chamber are cosmic rays (Radtke 2001). About two-thirds of sea level cosmic rays are muons; one sixth are electrons, and most of the remaining one-sixth are neutrons. Neutrons cannot be directly observed, because they will not ionize air within the chamber. Low energy (<100keV) electrons can be identified from the convoluted character of the tracks. Higher energy electrons and muons form straighter tracks.

ICCMR developed a system referred to as *Cloud Catcher*. With a camera placed above the cloud chamber, *Cloud Catcher* is programmed to pick up the ions created by the radioactive particles and translate their trajectories into sound. It provides real-time audio input granulation (Truax 1988) driven by live video colour tracking. The system carries out video colour tracking by calculating bounding dimensions for a range of values. A frame of video is represented as a two-dimensional matrix, with each cell representing a pixel of the frame, and each cell containing four values representing alpha, red, green, and blue on a scale from 0 to 255 (RGB standard). The system scans the matrix for values in the range [min, max] and outputs the minimum and maximum points that contain values in the range [min, max] within the matrix. The bounding region is a rectangle, thus the software outputs the indices for the left-top and bottom-right cells of the region in which it finds the specified values.

Cloud Catcher provides the ability to define a colour range, which allows for targeting suitable colour ranges in the images. Every time a colour in the chosen range appears on the video, it will produce as output two coordinates related to the region boundaries; otherwise it will have no output. These two coordinates are then used to control the audio output, through real-time granular sampling.

Granular sampling is a variant of the granular synthesis techniques introduced in the previous session. Instead of synthesizing the sound grains from scratch, here the system employs a granulator mechanism to extract small portions of a given sound.

The granulator uses these portions to produce a new sound in a number of ways. The simplest method is to extract only a single grain and replicate it many times, as shown in Figure 8 (Miranda 2002). In *Cloud Chamber*, the audio input to the granulator is taken from the violin during performance and *Cloud Catcher* controls the way the grains are re-combined to produce new sounds (Kirke et al. 2011).

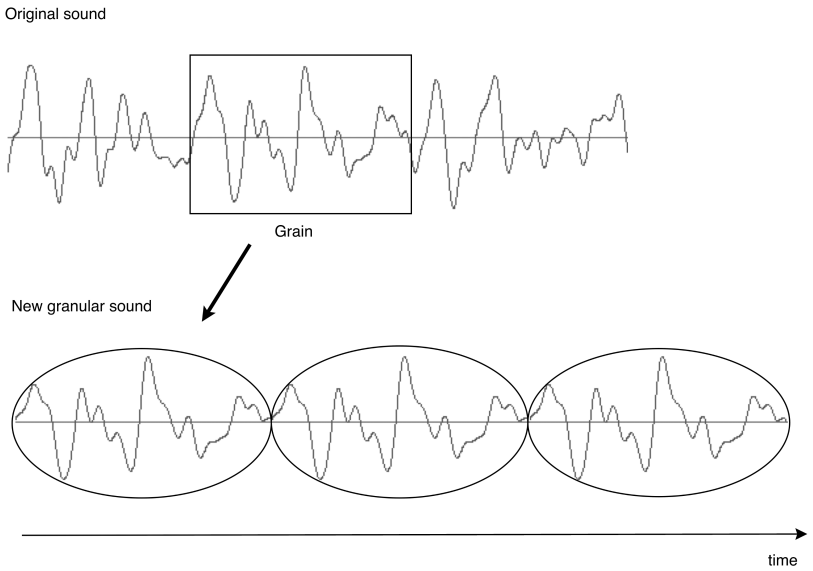

Figure 8: Granular sampling works by extracting grains from a given sound.

ICCMR devised an interface for *Cloud Catcher*, which enables a musician to use an acoustic instrument live to create a physical force field that directly affects the ions generated by radioactive particles. In short, we capture the sound of the violin with a clip-in microphone and convert it into an electrical signal, which is used to modulate a high-voltage power supply with an adjustable output between 1.5 and 3 kV. This connects to a projection field electrode installed in the cloud chamber. Thus the violin sounds modulate a positive potential in the chamber top. Applying a varying positive potential to the chamber top will directly change the particle tracks appearing in the chamber. Therefore the violinist can influence the behaviour of the subatomic particle tracks in the chamber during the performance and to a certain extent the violinist can be influenced by the sounds of the subatomic particles.

Musically speaking, *Cloud Chamber* is a semi-deterministic piece of music. Despite the fact that the violinist can be influenced by the sounds emanating from the *Cloud Catcher*, there is a musical score for the violinist to follow. The score draws on the Standard Model of Particle Physics for inspiration (Ellis 2006); it contains melodies generated algorithmically using the quark structure of observable Baryons and data kindly provided by ISIS Neutron and Muon Source, in Oxford, UK, from their experiment shining neutrons through liquid crystal (Newby et al. 2009). It should be noted that the use of this approach to generating musical material algorithmically for the score is not argued as being a meaningful expression of the Standard Model or of quarks. It was mainly used as a framework around which the composer could construct the piece.

Nevertheless, what is interesting here is that the method by which the particles generate sounds is based on non-deterministic quantum principles. Therefore, it is reasonable to believe that *Cloud Chamber* is the first piece of quantum music ever composed.

Figure 9: First performance of Cloud Chamber in 2011 in Plymouth, UK.

Cloud Chamber received its premiere in 2011 at Peninsula Arts Contemporary Music Festival (PACMF), Plymouth, UK. It was performed subsequently at ISIS Neutron and Muon Source, at the Rutherford-Appleton Laboratories, Oxford, UK and at California Academy of Sciences, in San Francisco, USA. A movie documenting the performance in San Francisco is available on-line (Kirke et al. 2013).

2.4 Making Sounds with *In Vitro* Neuronal Networks

Research into harnessing the complex dynamics of cultured brain cells to develop novel processing devices using neuronal networks cultured on circuit boards has been gaining momentum since DeMarse et al. (2001) reported the development of an artificial animal, or Animat, controlled by a processor built with using dissociated cortical neurones from rats. Distributed patterns of neural activity, also referred to as spike trains, controlled the behaviour of the Animat in a computer-simulated virtual environment. The Animat provided electrical feedback about its movement within its

environment to the neurones on the processing device. Changes in the Animat's behaviour were studied together with the neuronal processes that produced those changes in an attempt to understand how information was encoded and processed by the cultured neurones.

Increasingly sophisticated methods are being developed to culture brain tissue *in vitro* - neurones and glia - in a multi-electrode array (MEA) device, which is a mini Petri dish-like device with embedded electrodes (Figure 10). The electrodes can detect action potentials of aggregates of brain cells and stimulate them with electrical pulses (Figure 11). An MEA can record neuronal signals fast enough to detect the firing of thousands of nearby neurones as micro-voltage spikes. Neuronal network phenomena can be studied by supplying electrical stimulation through the multiple electrodes, which typically induces widespread neuronal activity (Potter et al. 2004, Bontorin et al. 2007, Novellino et al. 2007). Interestingly, Potter at al. (2004) introduced an art installation created with artists at SymbioticA in Australia. They connected an MEA device with cultured neurones in their lab in Atlanta to a robotic drawing arm in Perth. A video camera relayed the drawing process to Atlanta comparing the image in progress with a photograph of a person. The comparison generated a feedback signal for the cells on the MEA device.

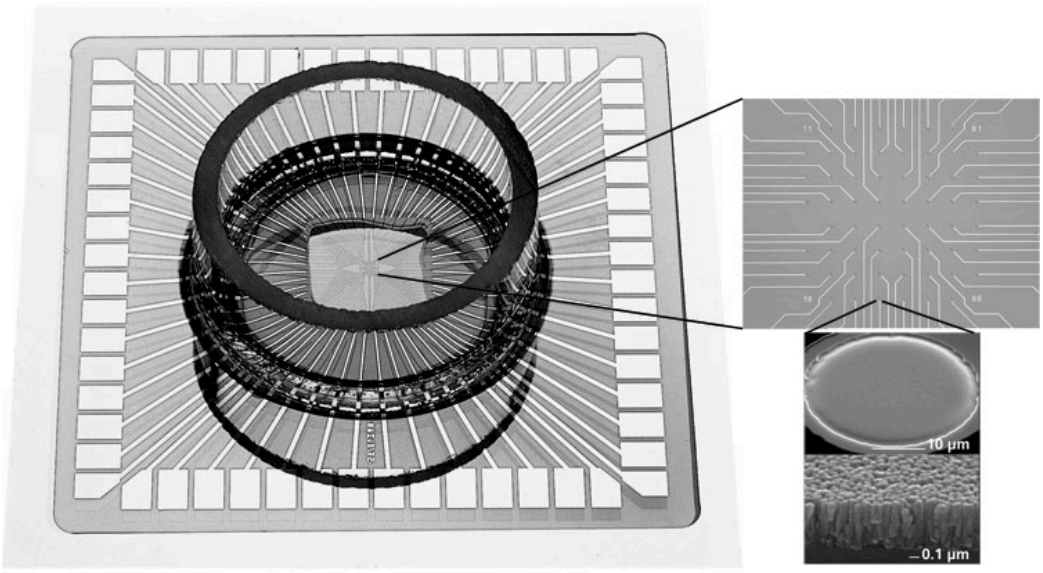

Figure 10: A typical MEA used to stimulate and record electrical activity of cultured brain cells on the surface of an array of electrodes. Reprinted from (Miranda et al. 2009) and with kind permission from Multichannel Systems http://www.multichannelsystems.com/.

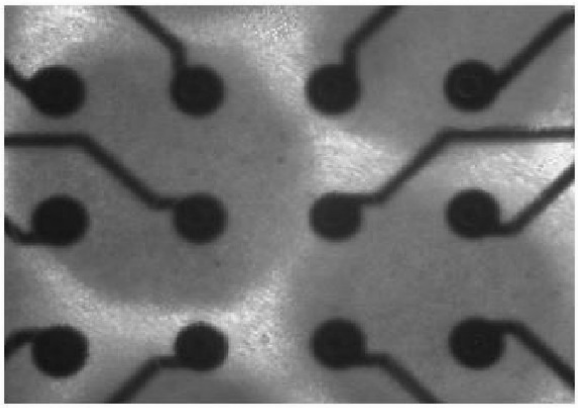

Figure 11: Phase contrast microscopy showing aggregates of cultured cells on a MEA device. Reprinted from (Miranda et al. 2009).

In vitro cultures of brain cells display a strong disposition to form synapses, especially so when subjected to electrical stimulation. The cells spontaneously branch out, even if left to themselves without external input other than nutrients in the dish. They establish connections with their neighbours within days, demonstrating an inherent bias to form communicating networks. In most cases, after a few weeks in culture, the development of these networks becomes relatively stable and is characterized by spontaneous bursts of activity (Kamioka et al. 1996). Furthermore, it has been possible to maintain functioning cultures of brain cells or a number of months, allowing for continuous long-term observations of their behaviour.

In addition to our far-reaching ambition of developing general-purpose bio-processors using living brain cells, ICCMR is particularly interested in exploring the potential of *in vitro* neuronal networks for developing bio-processors for sound and music because of their dynamic and rich temporal behaviour. To gain a better understanding of what it takes to develop our ambition, we conducted experiments with brain cells from seven-day-old chicken embryos, in collaboration with scientists at the University of the West of England, Bristol (Miranda et al. 2009). The objective was to harness the cells to build a musical instrument. We wanted to find out if it would it be possible to listen to the electrical activity of the cells, and if so, whether or not it would be possible to control this activity in order to make different sounds.

Figure 12 shows a typical chicken embryo aggregate neuronal culture, also referred to as a spheroid. These spheroids were grown in culture in an incubator for 21 days. Subsequently, they were placed into a MEA device in such a way that at least two electrodes made connections into the neurones of the spheroid. One electrode was arbitrarily designated as the input by which to apply electrical stimulation and another as the output from which to record the effects of the stimulation on the spheroid's spiking behaviour. Please refer to (Uroukov et al. 2006) for more information on the protocols for culturing cells and placement into an MEA device.

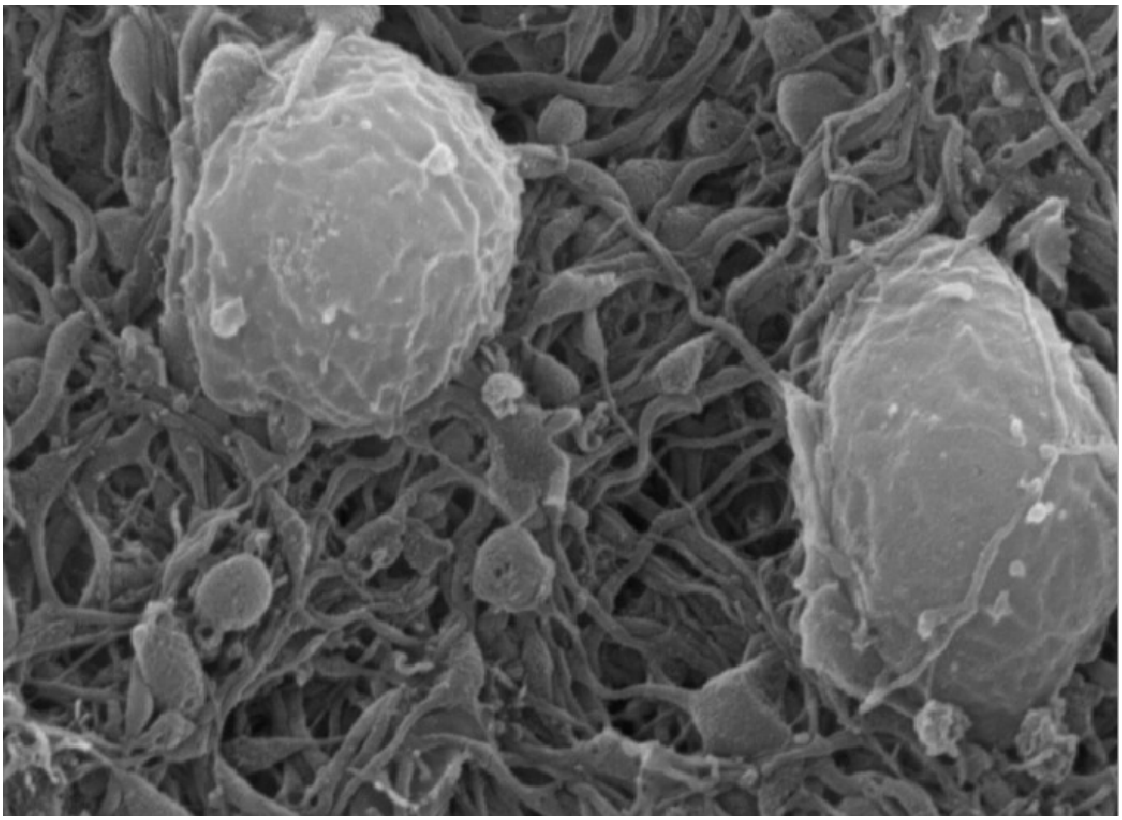

Figure 12: Image of a typical chicken embryo aggregate neuronal culture on a scanning electron microscope, magnified 5000 times. (Courtesy of Larry Bull, University of the West of England, UK.)

Stimulation at the input electrode typically consisted of a train of biphasic pulses of 300mV each, coming once every 300ms. This induced change in the stream of spikes at the output electrode, which was recorded and saved into a file. Stimulation sessions lasted for 60secs, with a 600-second rest between them. An increase in the spiking behaviour was observed after each session, which was an indication that such stimulations fostered self-organisation within the spheroid. The neuronal networks formed in such a way that external stimulation caused significant excitation within the neuronal network.

Figure 13 plots an excerpt lasting for 1sec of typical neuronal activity from one of the sessions. Note that the neurones are constantly firing spontaneously. The noticeable spikes of higher amplitude indicate concerted increases of firing activity by groups of neurones, which were responses to input stimuli.

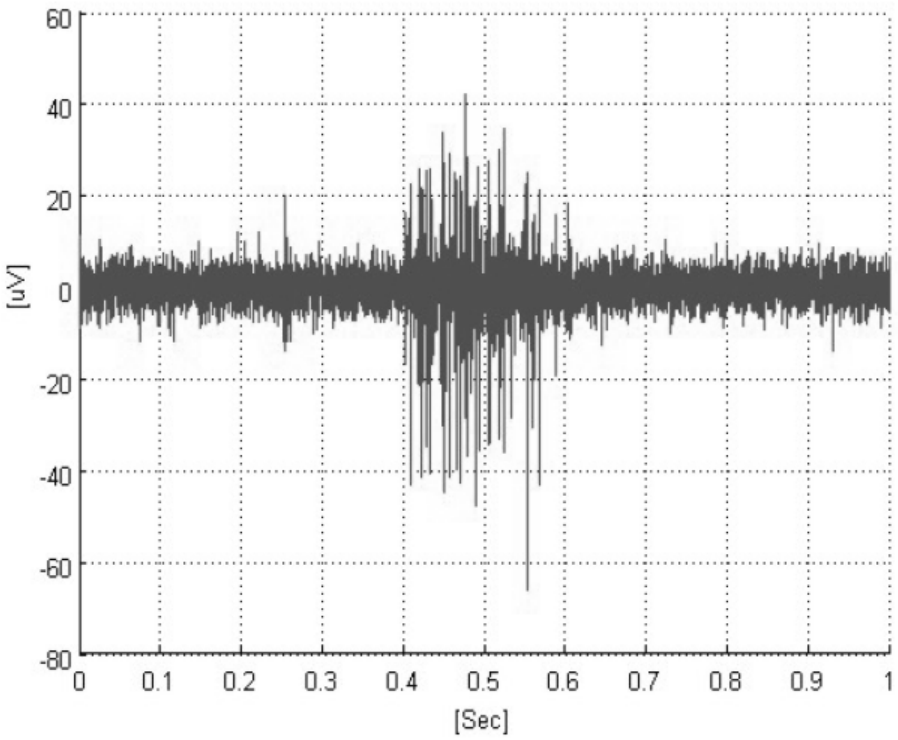

Figure 13: Plot of the first 1 second of a data file showing the activity of the spheroid in terms of μV against time. Induced spikes of higher amplitudes took place between 400ms and 600ms. Reprinted from (Miranda et al. 2009).

In order to listen to the neuronal activity, we developed a sonification technique, which combined aspects of additive synthesis and granular synthesis (Miranda 2002). The technique employed nine sinusoidal oscillators, each of which produced a partial for the resulting waveform. That is, the sound was composed of 9 sine waves. The system required 3 input values to generate a sound: frequency (*freq*), amplitude (*amp*) and duration (*dur*). Therefore, each reading from the output electrode yielded three values for the synthesiser: *freq*, *amp* and *dur*.

Essentially, the synthesiser was additive, comprising 9 sine wave oscillators. Each electrode reading generated *freq* and *amp* values for one of the oscillators, technically referred to as the fundamental oscillator. The values for the other 8 oscillators were relative to the values of the fundamental oscillator; e.g., $freq_{osc2} = freq_{osc1} \times 0.7$, $freq_{osc3} = freq_{osc1} \times 0.6$, and so on. The same applies for the amplitudes of the partials. The synthesiser was also granular because each of these readings produced a very short burst of sound. In effect, each spike of the spike train was translated into a grain of granular sound synthesis.

The frequency of the fundamental oscillator was calculated in Hz as follows: $freq = (datum \times \varphi) + \alpha$. We set $\alpha = 440$ as an arbitrary reference to 440Hz; changes to

this value will produce sounds at different registers. The variable φ is a scaling factor, which accounted for the range of values in the data file. This scaling factor needed to be variable because the range of μV values produced by the spheroids may vary with different experimental conditions. Typically $\varphi = 20$.

The synthesiser's amplitude parameter was a number between 0 and 10. The amplitude was calculated as follows: $amp = 2 \times \log_{10}(abs(datum) + 0.01) + 4.5$. This produced a value between 0.5 and 9.5. In order to avoid negative amplitudes we took the absolute value of the datum. Then, 0.01 was added in order to avoid the case of logarithm of 0, which cannot be computed. We later decided to multiply the result of the logarithm by 2, in order to increase the interval between the amplitudes. Since $\log_{10}(0.01) = -2$, if we multiplied this result by 2 then the minimum possible outcome would have been equal to -4. We added 4.5 to the result because our aim was to assign a positive amplitude value to every datum, even if it values 0μV.

The duration of the sound was calculated in seconds; it was proportional to the absolute value of the datum, which was divided by a constant c: $dur = \frac{abs(datum)}{c} + t$. Typically $c = 100$. Initially, the sonification technique produced a sound grain for every datum. However, this generated excessively long sounds. In order to address this problem, we developed a method to compress the data, which preserved the behaviour that we wanted to sonify, namely patterns of neural activity and induced spikes. For a detailed explanation of the compression method, please refer to (Miranda et al. 2009).

Figure 14 shows the cochleogram of an excerpt of a sonification, where one can clearly observe sonic activity corresponding to induced spiking activity.

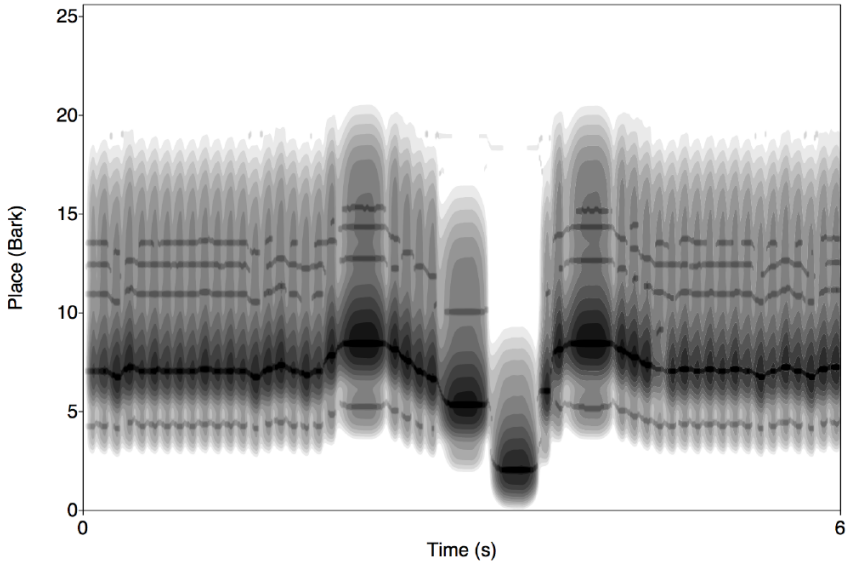

Figure 14: Cochleogram of an excerpt of a sonification of the spiking data plotted in Figure 13. Note that spikes of higher amplitude produced variations in the spectrum of the resulting sound, as shown in the middle of the cochleogram. Reprinted from (Miranda et al. 2009).

With this experiment we were able to test the hypothesis that it is possible to build a musical instrument using *in vitro* neural networks. We developed a method to listen to the electrical activity of the cells. Moreover, we were able to play the instrument, i.e., produce sound variations, by inducing spiking behaviour through electrical stimulation.

The idea of harnessing the naturally elegant and efficient problem-solving methods of biological organisms to build novel computing systems is an important approach to research into UC, and the one that is probably most accessible to computer musicians as we write this chapter. A variety of reports have been published describing research into harnessing properties of biological tissues or organisms to perform certain types of computational tasks (Armstrong and Ferracina 2013; Adamatzky 2016). However, research into harnessing *in vitro* neurones is currently unrealistic for the great majority of investigators and entrepreneurs looking into exploring practical applications of UC developments, including computer music. Nonetheless, emerging research into using the plasmodial slime mould *Physarum polycephalum* is proving to be an affordable alternative: this organism is openly obtainable, economical to culture, safe to handle (not toxic), and does not require expensive equipment to develop experiments.

Figure 15: *Physarum polycephalum* is a suitable biological medium to develop research into UC for musical applications.

2.5 Probing the Potential of Slime Mould Computing

Physarum polycephalum, henceforth referred to as *P.polycephalum*, is naturally found in cool, moist and dark environments. It exhibits a complex lifecycle, but the point of interest here is its vegetative plasmodium stage. During its vegetative plasmodium stage, it exists as a single amorphous cell visible via the human eye, with a multitude of nuclei; hence the term 'polycephalum', which literarily means 'many heads'. The plasmodium is capable of responding with natural parallelism to surrounding environmental conditions: it grows towards chemo-attractants (food) and moves away from chemo-repellents (e.g., salt). As it propagates along gradients of stimuli it develops a network of protoplasmic filaments connecting areas of colonization. The cytoplasm contains a semi-ridged cytoskeleton embedded with actin-myosin filaments that rhythmically contract and expand, producing the shuttle streaming of its internal fluid endoplasm. These rhythms are coupled with spatially distributed biochemical oscillations. Indeed, the topology of the slime mould can be described as a network of biochemical oscillators: waves of contraction or relaxation, which collide inducing

shuttle streaming. This intracellular activity produces fluctuating levels of electrical potential as pressure within the cell changes. Typically this is in the range of ±50mV, displaying oscillations at periods of approximately 50-200sec with amplitudes of ±5-10mV dependent on the organism's physiological state (Meyer and Stocking 1970). Research has been put forward demonstrating that such patterns can be used to accurately denote behaviour (Adamatzky 2010). This is one of the main features of *P.polycephalum* that renders it attractive for research into UC.

A natural characteristic of the slime mould is the time it can take to span an environment. Depending on how the experimental environment is set up, it can take several hours to exhibit substantial growth and exhaust available sources of nutrients. Much research is being conducted worldwide to utilise more instantaneous behavioural aspects of the organism; e.g., using intracellular activity as real-time logic gates. In the meantime, one solution that has been adopted by a number of research laboratories is to work with computer models of the slime mould (Jones 2010). Another approach, which is the one adopted for our investigation at ICCMR, is to record the behaviour of the slime and subsequently use the data off-line. This approach enables one to experiment directly with the biological substrate.

The plasmodium is relatively easy to culture in Petri dishes with scattered sources of food, such as oat flakes (Figure 16). It is possible to prompt it to behave in controlled ways by placing attractants and repellents on the dish. The ability to manipulate growth patterns has underpinned the early stages of research into building *P.polycephalum*-based machines to realize tasks deemed as computational: e.g., the organism was prompted to find the shortest path to a target destination through a maze (Adamatzky 2010) and solve the classic combinatorial optimization Steiner tree problem (Caleffi et al. 2015). However, it is the electrical properties of the organism that have more recently been the focus of research: electrical current can be relayed through its protoplasmic filaments and its intracellular activity can function as logic gates.

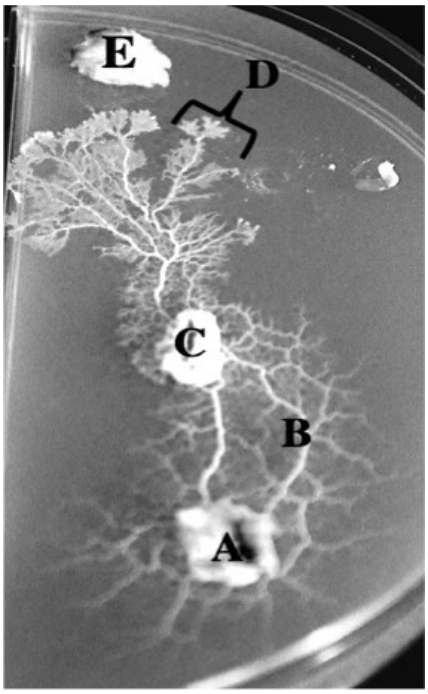

Figure 16: Photograph of a Petri dish with the slime in plasmodium state, showing: (A) a place where it has been inoculated, (B) protoplasmic network connecting areas of colonisation, (C) colonised region containing nutrients (in this case an oat flake), and (D) extending pseudopods forming a search front along a gradient towards another oat flake, marked by (E).

Miranda and his team have conducted a number of experiments investigating ways to harness the behaviour of the slime mould with a view on building audio and music systems, and ultimately a musical biocomputer. In a preliminary study, a foraging environment was constructed with electrodes embedded into areas containing oat flakes. Electrical potentials were recorded from these electrodes as the slime mould navigated within the foraging environment. The recorded data from each electrode were rendered as frequency and amplitude values for a bank of oscillators forming an additive synthesiser. In order to record the behaviour of the slime mould we use a combination of time-lapse imagery and/or electrical potential data by means of bare-wire electrodes (Miranda et al. 2011). Subsequent experiments include the development of a musical step sequencer, a sound synthesiser and a generative music system. Those initial experiments produced encouraging results, which paved the way to our current research into building a *P.polycephalum*-based bio-processing device: the biomemristor. Below we introduce the preliminary work that paved the way to our biomemristor research, which is detailed in Chapter 8.

2.5.1 Musical Step Sequencer

Musical step sequencers are devices that loop through a defined quantity of steps at given time intervals. Each of these steps can normally exist in one of two states: active or inactive. When active, a given sound event will be triggered as the sequencer reaches its respective position in the loop. No sound is produced when the reached position is inactive.

In order to implement the slime mould step sequencer, we designed an environment that represents a step sequencer's architecture as schematically shown in Figure 17. It consisted of a Petri dish divided into six electrode zones, representing sequencing steps (S1, ..., S6), arranged in a circular fashion (representing the sequencer loop) with a central inoculation area.

Figure 17: Step sequencer architecture.

Oat flakes were placed on the electrode zones and in the inoculation area, and the slime mould was placed in the inoculation area. In order to record the behaviour of the slime we used two forms of hardware: a USB manual focus camera and high-resolution data logger. Each experiment took place in a 90mm Petri dish with the camera centred above. In order to guarantee an environment that promoted growth, a black enclosure was placed over the Petri dish limiting the light intensity level imposed on the slime mould. Lighting for image capture was achieved by means of an array of white LEDs, which turned on and off periodically. Electrical potential levels were recorded using an ADC-20 high-resolution data logger. Within each Petri dish, bare-wire electrodes are placed through small holes in the base with wiring underneath secured using adhesive tack (Figure 18, left hand side). The electrodes are arranged with one reference electrode and six measurement electrodes, one for each step of the sequencer: the reference resides in the centre (i.e., in the inoculation area) and gave a ground potential for each of the measurement electrodes.

Each electrode was coated in non-nutrient agar, which kept humidity high for the slime to grow. Due to the agar substrate being liquid-based and thus a conductor, a non-conductive plastic sheet isolated each step (electrode zone) area, allowing electrical potentials to be recorded across the environment without interference.

Oat flakes were placed on each step on top of the respective electrode wire, maintaining an equal distance from the centre to entice the propagation and facilitate colonisation (Figure 18, right hand side).

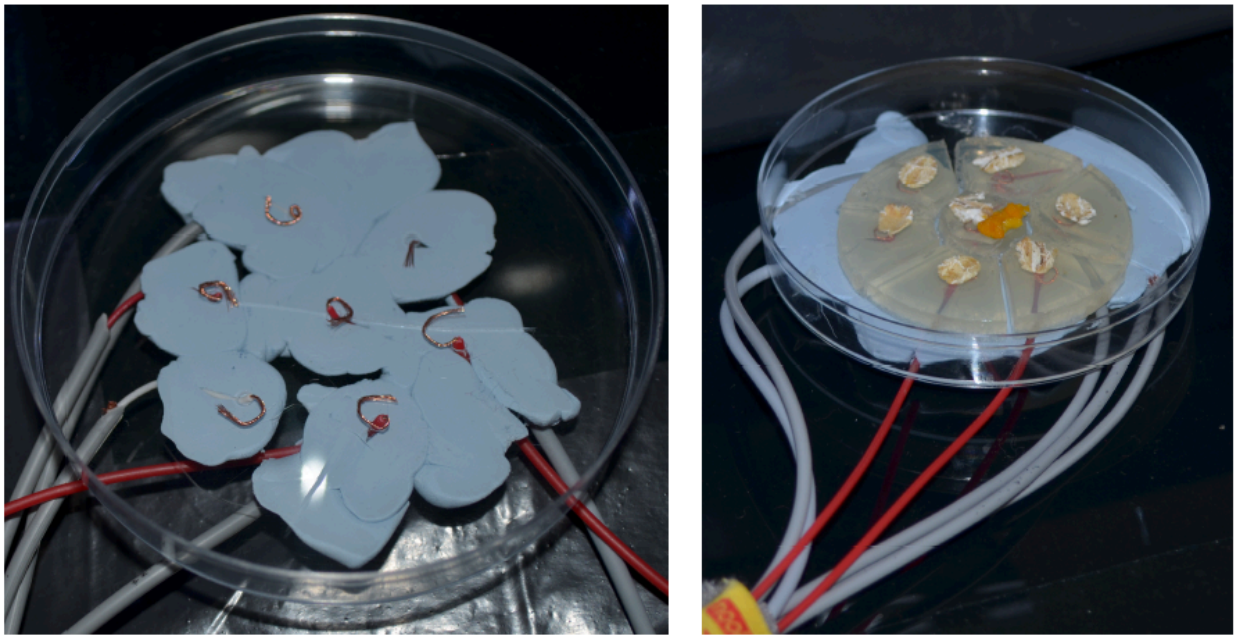

Figure 18: Photographs showing the construction of the environment for the step sequencer. Shown on the left hand side is the bare-wire electrode array wired into position. The right hand side shows the completed growth environment with each electrode embedded within blocks of agar.

To start the process, we inoculated a piece of slime mould in the centre region and began to record data. Inoculation sources were extracted from a small *P.polycephalum* farm that we maintained at ICCMR and put through a period of approximately six hours of starvation before the experiment began. This starvation process accelerates initial propagation speed.

We programmed the system to take 100 data samples from each electrode at intervals of 1 sec throughout the duration of the experiment. Samples were then averaged to give a single reading for each second. This level of recording detail was necessary in order to capture the natural gradients associated with various progressions, some of which are fairly prompt. Image snapshots were taken at intervals of 5 minutes with the LEDs turning on 5 sec before and staying active for 10 sec.

In order to use the recorded date for music, we first established a data recall system. Here, the user could define how fast they wished the electrical potential data to be recalled in terms of number of entries per second. Upon doing so, the system defined the frame rate for the time lapsed imagery, in order to play back the images in motion with perfect synchrony. The interface was built around the time lapsed imagery

playback, forging a connection between the user and the organism. Once in the system, each electrical potential entry was broken into its six individual readings and adjusted to become an absolute value. The system then stepped through each measurement taking a reading at a user-defined speed in terms of beat per minutes. A step only became active within the sequence once it was colonised by the slime. Otherwise, no reading was taken. Once activated, the system looked for a level of change in electrical potential in order to retire steps from triggering sounds when the slime mould was no longer active. This was achieved by storing readings over a short period of time and reviewing any oscillatory behaviour.

We developed several variations of the sequencer to probe its usability in realistic musical production. One of these versions looked at harnessing the slime mould's behaviour to extend the functionality of a conventional step sequencer by triggering different sounds as a function of each step's electrical potential readings. The readings were used to trigger one of four sound samples assigned to each step. The system allowed us to associate 4 sounds to each step of the sequencer. Each of them was subsequently associated with a voltage range; e.g., sound A would be triggered if the voltage values between -15mV and +15mV, sound B if it values between 16mV and 30mV, and so on.

The user interface (Figure 19) provided an interactive graph showing the combined electrical potential readings (on the top right hand side). This provided the ability to change the current position of the data being recalled, creating means to restructure the output of the sequencer.

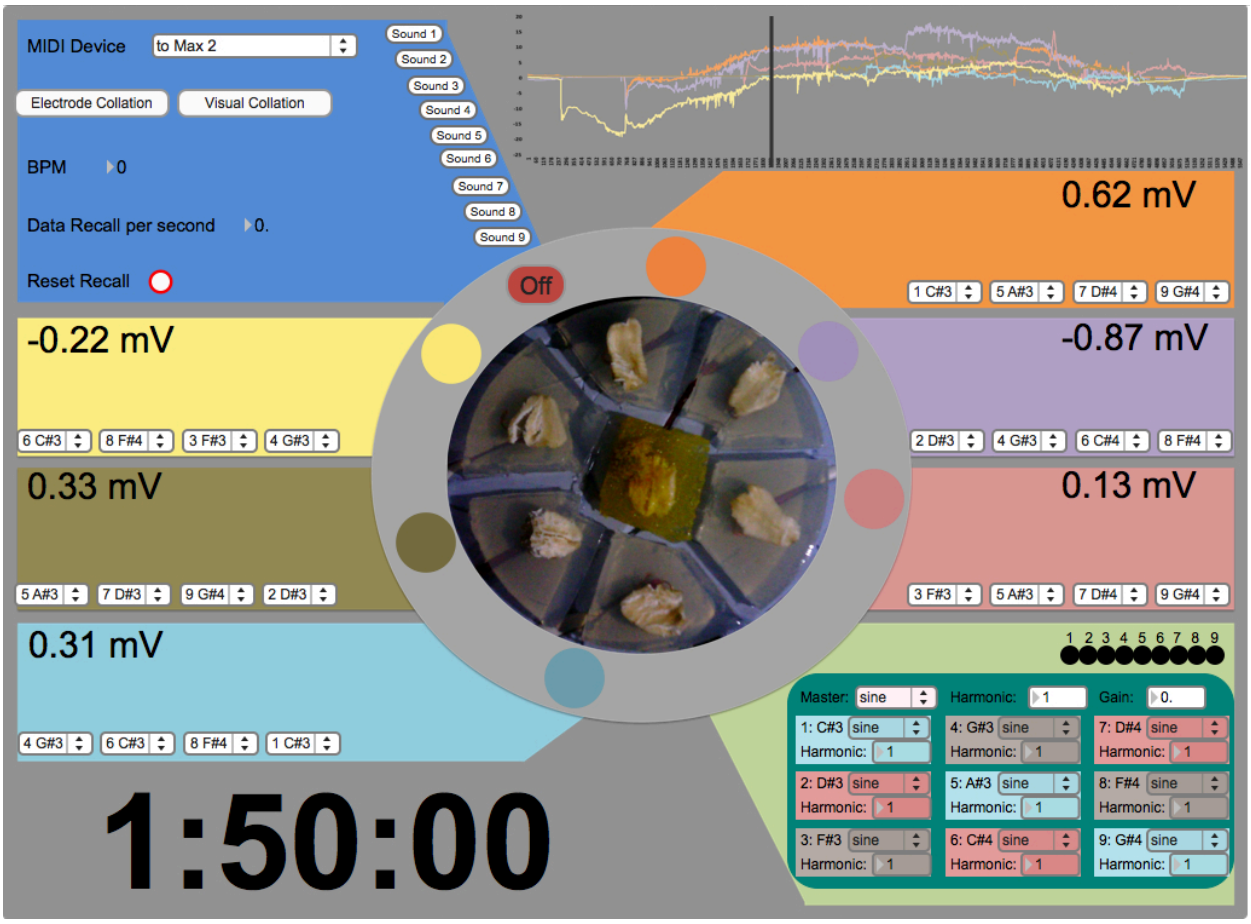

Figure 19: The slime step sequencer user interface.

2.5.2 Slime Granular Synthesis

As we have seen in section 2.2, granular synthesis works by generating a rapid succession of very short sound bursts. There have been various approaches to composing sounds with granular synthesis algorithmically. For instance, Valle and Lombardo (2003) developed a method that employed a directed graph. Sequences of grains were produced according to a graph actant that moved between connected vertices, each of which represented a sound generator.

Inspired by Valle and Lombardo's graphs-based system, we exploited the slime mould's ability to create and reconfigure networks of protoplasmic veins between sources of nutrients to build a granular synthesiser. In a nutshell, the slime mould's protoplasmic network created sequences of grains that were directly sampled from the organism's oscillatory behaviour.

The slime mould environment that we created for the synthesiser resembled the one that we designed for the step sequencer introduced above. Oat flakes were placed in a

Petri dish, on zones furnished with electrodes. Each of these electrode zones was associated with a different grain generator: the electrodes read electrical data from the slime, which were used to generate sound grains (Figure 20). A generator started producing grains as soon as the slime mould colonised the respective electrode zone, and started digesting the oat flake. It ceased to produce grains only when the respective oat flake had been consumed.

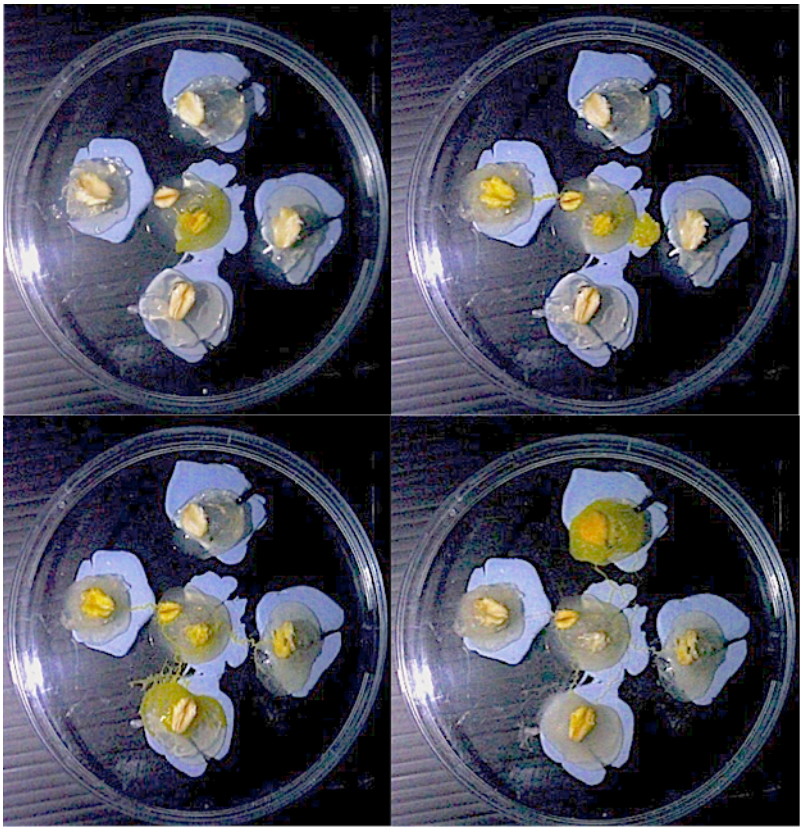

Figure 20: The slime environment for the granular synthesiser.

At a predetermined granular sampling rate, the organism's oscillatory behaviour was sampled to produce sound grains. The electrical readings from each electrode were scaled into audio buffers. At defined time intervals, the appropriate buffers were addressed to produce sound grains, which were subsequently sequenced together in descending order according to each electrode's running average potential. The amplitudes of the grains were established by scaling the respective electrode's electrical potential reading at the time of streaming from the audio buffers to a pre-defined range. Their durations were defined using a potential difference value scaled to a composer-defined minimum and maximum range.

In order to ensure that grains were only produced from electrodes colonised by the slime mould, our system monitored each electrode's readings for oscillations. When no

oscillatory behaviour was registered across the whole environment, the system halted and rendered the final audio.

We set the parameters of our system to produce grains between 30 and 100 milliseconds long every 5 hours. The results were spectrally rich and dynamic sounds (Figure 21). Such morphology was a direct consequence of using a slow adapting ever-changing living entity.

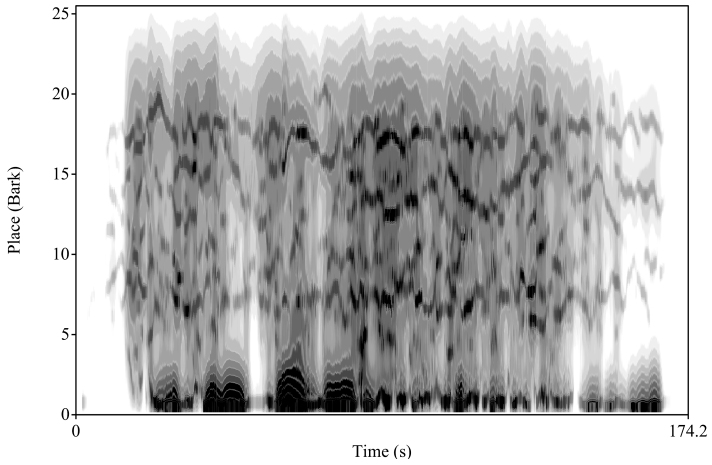

Figure 21: Cochleogram of a typical sound produced by the slime granular synthesiser.

2.5.3 Kolmogorov-Uspensky Musical Machine

A Kolmogorov-Uspensky (KU) machine (Kolmogorov and Uspenskii 1958) is an abstract computation model that computes the same class of functions as the Turing machine (Turing 1936). However, in contrast to the Turing machine's tape, a KU machine utilizes a finite undirected connected graph with bounded degrees of nodes and labels as its storage structure, which it can reconfigure. Only one node can be active at a given time step, and each node and edge must be uniquely labelled. Thus, every passage from the active node can be described as a sting of their unique labels. A fixed radius around the active node is known as the active zone. According to the isomorphism of the active zone, the program executes the following instructions:

- Add new node with a pair of edges connecting to the active node
- Remove node and its incident edges
- Add edges between nodes
- Remove edges between nodes
- Halt

The computational process moves on the graph, activating nodes and adding and removing edges in accordance with a program. Once the program has executed the

instructions, the internal state changes accordingly. Adamatzky provided a step-by-step comparison of a KU machine and the *P.polycephalum*'s behaviour, speculating that the organism is the best biological realization of a KU machine one can possibly find in nature (Adamatzky 2007).

In this section, we report on an algorithmic composition method that harnesses Adamatzky's framework for a KU machine implemented with *P.polycephalum*. His implementation of a biological KU machine exploited the plasmodium's natural ability to form planar graphs, which it can dynamically reconfigure over time. Figure 22 depicts a visual comparison of a culture of plasmodium and a KU machine.

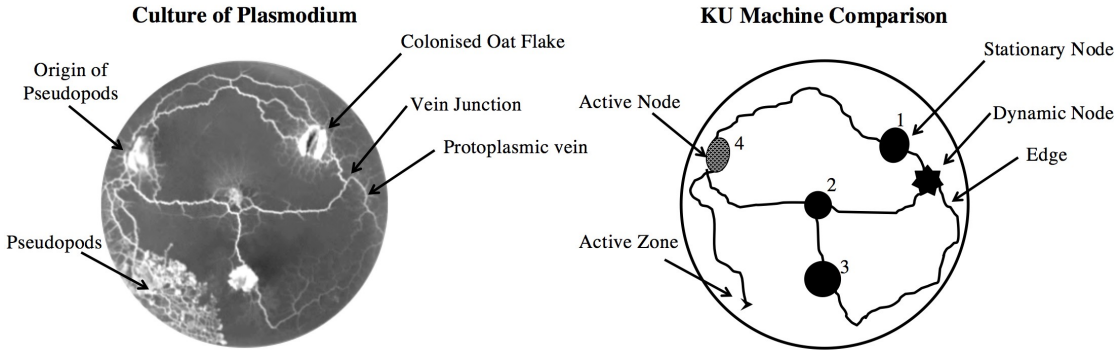

Figure 22: A visual comparison of the culture of plasmodium (left) displayed in Figure 1 and a KU machine (right).

The elements of the KU machine depicted in Figure 22 are summarised as follows:

- **Active zone**: At every time step there must be an active node. This is a built-in function of the plasmodium: the organism generates waves of vein contraction, which cause a pressure gradient to build up in the tubes. Such pressure results in the periodic movement of protoplasm back-and-forth, changing direction approximately every 50 seconds with greater net flow occurring in the direction of propagation.

- **Nodes**: A *P.polycephalum* KU machine has two types of nodes: stationary and dynamic. Sources of food represent stationary nodes while all other sites where two or more veins originate (protoplasmic vein junctions) represent dynamic nodes. Often, the organism forms dynamic nodes when extending pseudopods that break off into two or more directions, resulting in a single vein branching into two or more.

- **Edges**: Protoplasmic veins that connect nodes represent edges. A KU machine's storage graph is undirected. For example, if nodes *a* and *b* are connected, then they are connected with edges (ab) and (ba). A *P.polycephalum*

KU machine implements this with a single vein but with the periodic movement of protoplasm back-and-forth.

- **Data input and program**: Data input and program are represented by the spatial configuration of stationary nodes (oat flakes).

- **Addressing and labelling**: *P.polycephalum* does not implement any aspects that provide a direct method of uniquely labelling nodes and edges. Adamatzky suggested and experimented with using food colouring. However, in our approach to a biological KU machine, we will use either handwritten or digital labels.

- **Results**: The plasmodium halts the computation when all data nodes are utilized - when it has exhausted all available food sources - or when humidity levels are too low for it to continue foraging. When this occurs, the plasmodium enters is dried up dormant state (or sclerotium state). The result of a computation is the final graph structure formed by the plasmodium's protoplasmic vein network.

For our algorithmic composition approach, each node represented a predetermined musical phrase ($P^1, \ldots P^n$), which was input by the user. Nodes were distributed within the computational arena at the beginning of an experimental run. This configuration defined the system's state at the beginning of the experimentation (t = 0). Upon the active zone connecting nodes with edges, the connected node's musical phrases were transformed using a set of composer-defined rules to create a new iteration (P^n). These rules could either be universal, unique to each node, or dependent on the isomorphism of the active node's neighbourhood. The respective node's memory was subsequently updated with the new iteration. As a KU machine's storage graph is undirected, both nodes' phrases were transformed, with the destination node being processed first. Once the system has updated the destination node's memory, it placed the newly transformed phrase into an output sequence, creating a progressive arrangement of musical phrases. The algorithm at each time step looks like this:

ADD EDGE (P^a, P^b)
    READ P^a
    TRANSFORM P_i^a
UPDATE P^a with P_i^a
OUTPUT P^a
ADD EDGE (P^b, P^a)
    READ P^b
    TRANSFORM P_i^b
UPDATE P^b with P_i^b

In addition to our *P.polycephalum* KU machine framework, there were ways a composer could further augment the algorithmic composition process. By harnessing phenomena that attract, repel or retard the organism, a composer could gain additional control. Such phenomena include glucose and various carbohydrates for attractants, salt and metal ions for repellents, and experimental temperature for retardants. Through the use of these stimuli, a composer could, for example, restrict access to certain nodes, intensify a node's stimuli gradient and cause a computation to prematurely halt.

In the field of algorithmic composition, practitioners often incorporate chance and/or pseudo-random processes. An interesting consequence of using a *P.polycephalum* KU machine for algorithmic composition was that these are given by default. As the plasmodium is an ever-changing living entity, we cannot predict its propagation trajectory between nodes with absolutely certainty. Furthermore, in some cases environmental factors out of our control may impact the organism's behaviour. For example, nodes (in this case, oat flakes) may become infected, causing them to alter classification from attractant to repellent. It is also likely that computations within the same experimental setup will have varying results.

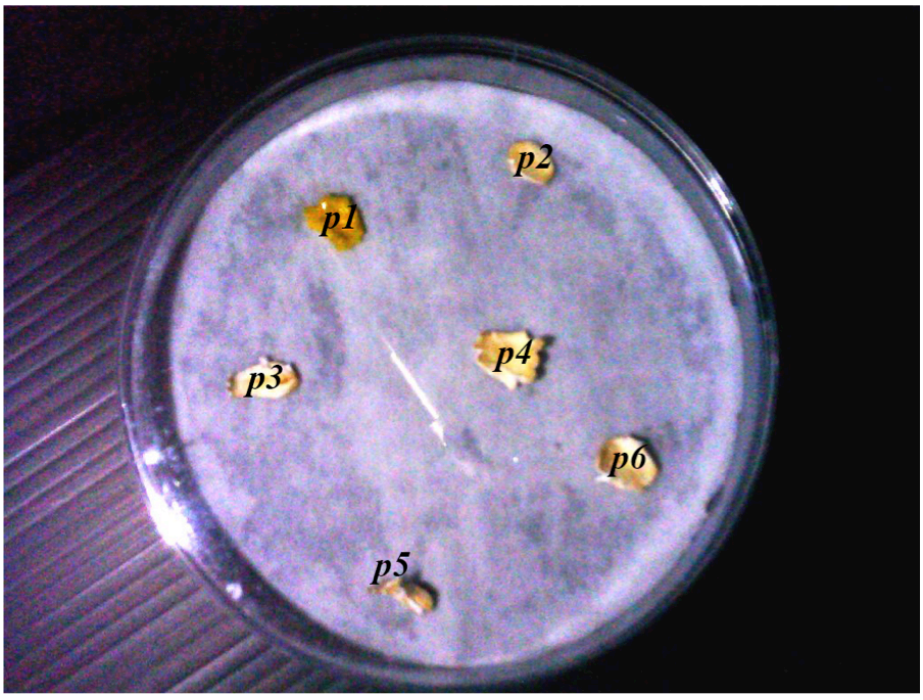

Figure 23: A photograph of the experimental space at the start of an experiment.

To experiment with our approach, we distributed five oat flake nodes within 90mm Petri dishes lined with a moistened filter paper (Figure 23). A short musical phrase (Figure 24) was assigned to each of the five nodes. By way of creating the active zone and initiating the machine, we inoculated the plasmodium into the space using a colonized oat flake. As this oat flake also represented a node, we assigned it a musical phrase as well.

In order to interface with the KU machine, to interpret the plasmodium's behaviour and process the musical phrase transformations, we designed some custom software split into two sections, the recorder and the interpreter.

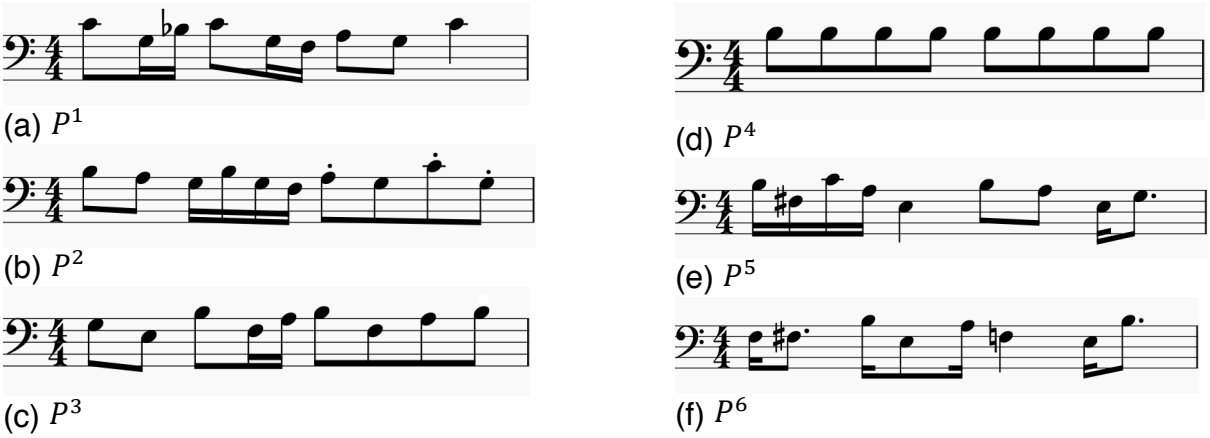

Figure 24: Musical phrases assigned to the nodes of the *P.polycephalum* KU machine.

The recorder section was used to apply digital labels and monitor the computation using time-lapsed imagery; we took snapshots every 30 minutes. Once the computation had halted, the interpreter section implemented the algorithmic composition. Here, the software was preloaded with our musical phrases (using MIDI files), and required us to inform it of the active zone's movement at each time step: e.g., $(P^1 \rightarrow P^2)$. In this example, we programmed the interpreter to transform musical phrases using universal rules, instead of assigning each node individual rules.

The rules split each phrase into four sections $(P^n x^1, \ldots, P^n x^4)$, and transform each MIDI note value, the delta time between note onsets and each note's duration. The rule compared values within each section against the mean μ of their respective counterpart section. The resulting difference values were then divided by the other section's respective standard deviation σ, and rounded to the nearest whole number. However, if the σ was equal to 0, the software combined both section's phrases and calculated a new value for σ. Subsequent values were next multiplied by their own section's σ and added to the mean μ, resulting in the transformed musical phrase. This is formalised as follows (1):

$$[P^a \rightarrow P^b] = \left(\left(\left(\frac{(P^b x_i^n - \mu P^a x^n)}{\sigma P^a x^n}\right) \times \sigma P^b x^n\right) + \mu P^b x^n\right) \quad (1)$$

If, however, $\boldsymbol{P^n x_i^n} < 1$, then the algorithm would replace the value with the μ of their respective counterpart section.

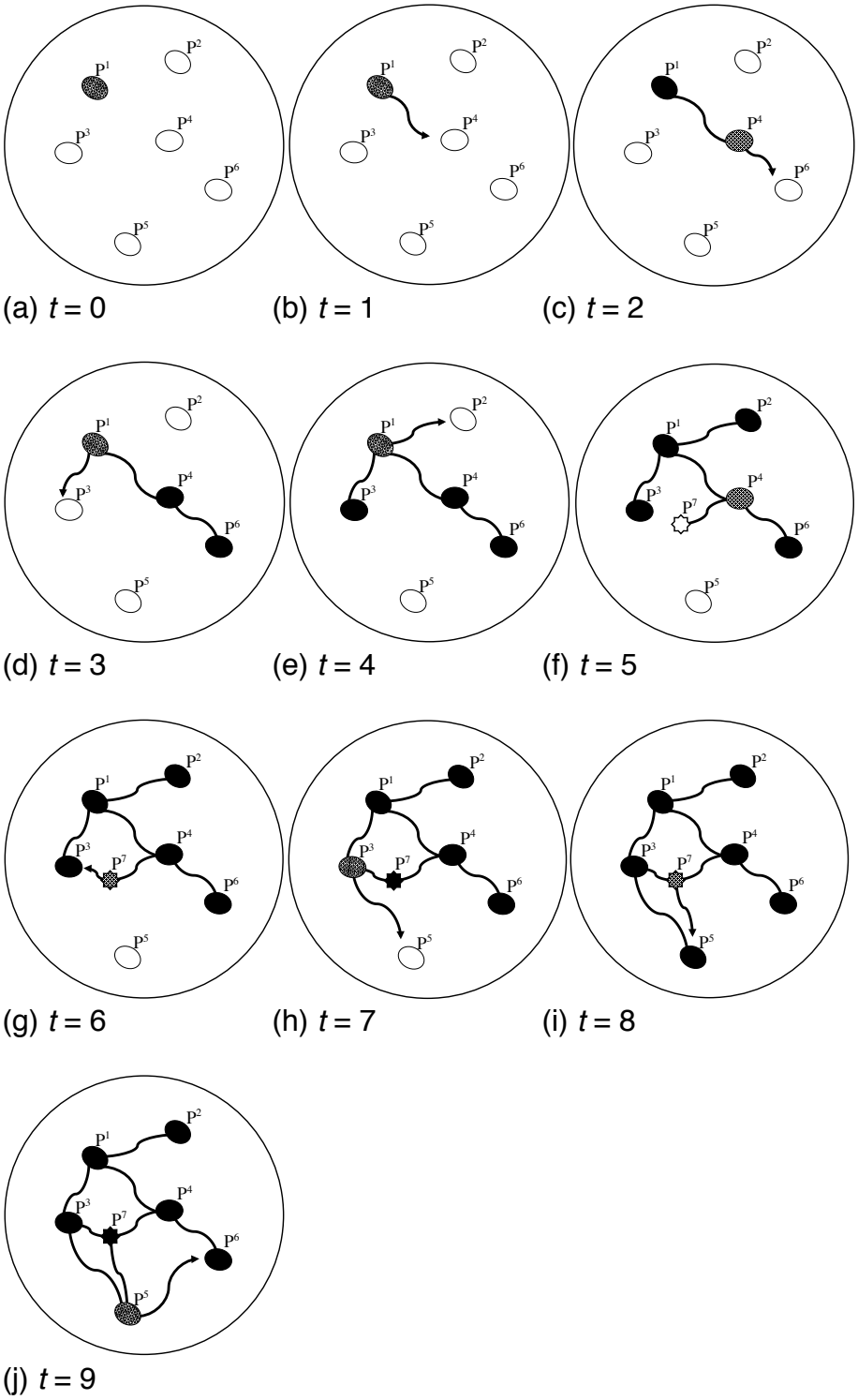

Figure 25: An overview of our experiment with the *P.polycephalum* KU machine.

An overview of an experiment with the *P.polycephalum* KU machine is shown in Figure 25, depicting the active zone dynamics described in Table 1. These

illustrations are of the experimental setup shown in Figure 23. Each illustration correlates to one of the sections in Figure 26. The time step when the plasmodium added a dynamic node to the storage structure is shown in 25(f), which is also shown in photograph form in Figure 28.

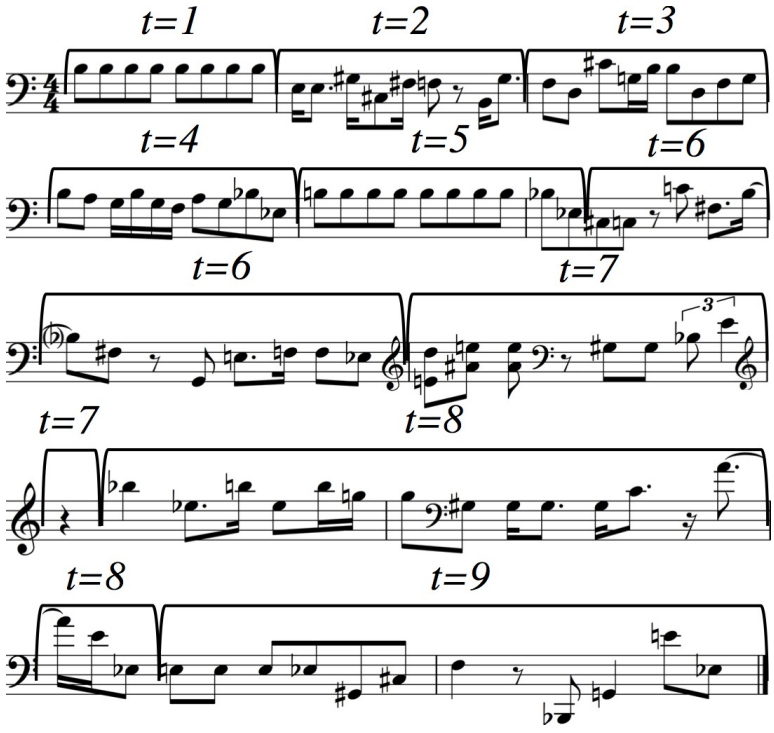

Figure 26: An example of a musical result. This score is labelled to correlate with the time steps displayed in Table 1 and the illustrations depicted in Figure 25.

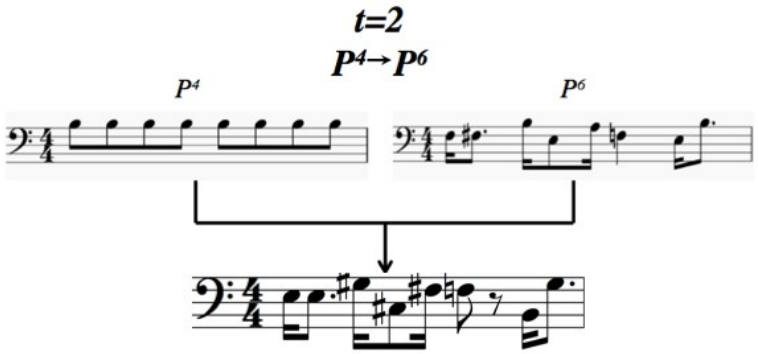

Figure 27: One of the transformations made by our system at t = 2, using the rules presented in Equation 1. As depicted in Figure 25(c), P^4 was the origin node and P^6 the destination.

Timestep	Origin Node	Destination Node
1	$P1$	$P4$
2	$P4$	$P6$
3	$P1$	$P3$
4	$P1$	$P2$
5	$P4$	$P7$
6	$P7$	$P3$
7	$P3$	$P5$
8	$P7$	$P5$
9	$P5$	$P6$

Table 1: The active zone movements of the example experiment.

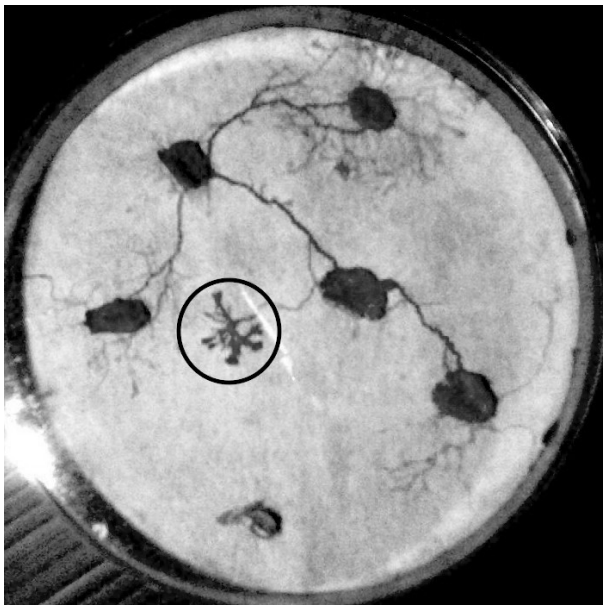

Figure 28: A photograph of t = 5, schematically shown in Figure 25(f). In this case, the active zone added a dynamic node to the storage structure by forming pseudopods that dispersed on several different trajectories.

If the organism added a dynamic node to the storage structure (a protoplasmic vein junction), our system combined the two phrases of the nodes at either end of the edge where the junction originated, and saved this phrase under a unique label P^n. However, if the plasmodium created the dynamic node from extending pseudopods that dispersed in more than two trajectories, then the node was assigned the original node's musical phrase. As the organism's protoplasmic vein network can become very complex, we only classified what we considered to be major vein junctions as dynamic nodes. An example of our dynamic node classification process is depicted in Figure 27. Our rationale for only classifying major junctions as dynamic nodes was

due to the techniques and imaging equipment we would require to analyse the smaller (microscopic) parts of the organism's network. Once the system had processed the transformations, it rendered the composition into a standard music file, which can be loaded into most music editing software.

The experiment described here took approximately 92 hours to complete, with the computation halting as the organism entered its dormant sclerotium phase. The organism entered this dormant phase as a result of it exhausting available sources of food and the filter paper substrate drying out. Table 1 summarizes active zone dynamics while Figure 25 displays a sequence of illustrations of the time-lapse photos. During the experiment, the plasmodium added one dynamic node $\boldsymbol{P^7}$ to the storage structure, which occurred at $t = 5$. This $\boldsymbol{P^7}$ node formed from an active zone originating from $\boldsymbol{P^4}$, which formed pseudopods that dispersed on two different trajectories (Figure 28), first arriving at $\boldsymbol{P^3}$ at $t = 6$, then arriving at $\boldsymbol{P^5}$ at $t = 8$. Figure 26 shows the algorithmic composition result of our experiment. As the organism created a dynamic node, the algorithm created a new musical phrase $\boldsymbol{P^7}$, which was a $\boldsymbol{P^4}$ phrase at $t = 5$. Depicted in Figure 27 is one of the transformations made by the system at $t = 2$, showing the transformation of nodes $\boldsymbol{P^4}$ and $\boldsymbol{P^6}$, resulting in a new iteration of $\boldsymbol{P^6}$. Note that the result of this algorithmic composition approach was reminiscent of the set of given musical phrases, but with modifications.

At this point of our research we were able to confirm that *P.polycephalum* exhibits properties that can be harnessed to implement systems for generative audio and music. The experience we gained from working with the organism and implementing the systems prepared the ground for our current research, which is aimed at building general-purpose bio-processors with this organism.

2.6 *Biocomputer Music*: Towards *Physarum polycephalum* Bio-processors

The material make up and scheme of our conventional computer's hardware have remained relatively unchanged throughout the years, with the main developments concerning reduction in size and heightened efficiency. The material base of those computing architectures has revolved around the three fundamental passive circuit components: capacitor, inductor and resistor. In 1971, Chua theorised a fourth fundamental component (Chua 1971): the memristor. Unlike the other three components, the memristor is non-linear and possesses a memory. The trajectory of these developments seems to be following the trajectory of Turing and von Neumann, in the sense that Chua came up with the theory and now someone needs to implement it in hardware (Strukov et al. 2008). There is no memristor commercially available to date; it still needs to be engineered for mass production.

The memristor is exciting because it has the potential to revolutionise the material basis of computation. The memristor changes its resistance according to the amount of charge that has previously run through it. As a result of its resistance function, a memory of previous states can be accessed by applying voltages across the component's terminals. For a detailed introduction to the memristor, please refer to Chapter 6.

In 2009, Pershin and colleagues published a paper that described *P.polycephalum*'s adaptive learning behaviour in terms of a memristive model (Pershin et al. 2009). Subsequently Gale and colleagues demonstrated in laboratory experiments that the protoplasmic tube of the slime mould displayed behaviour consistent with memristive systems (Gale et al. 2014). We followed these works with research aimed at implementing *P.polycephalum*-based memristors, or biomemristors. We have been developing increasingly sophisticated versions of biomemristors based on *P.polycephalum*, which we have used to build various prototypes of interactive musical biocomputers. Chapter 8 discusses this work in more detail.

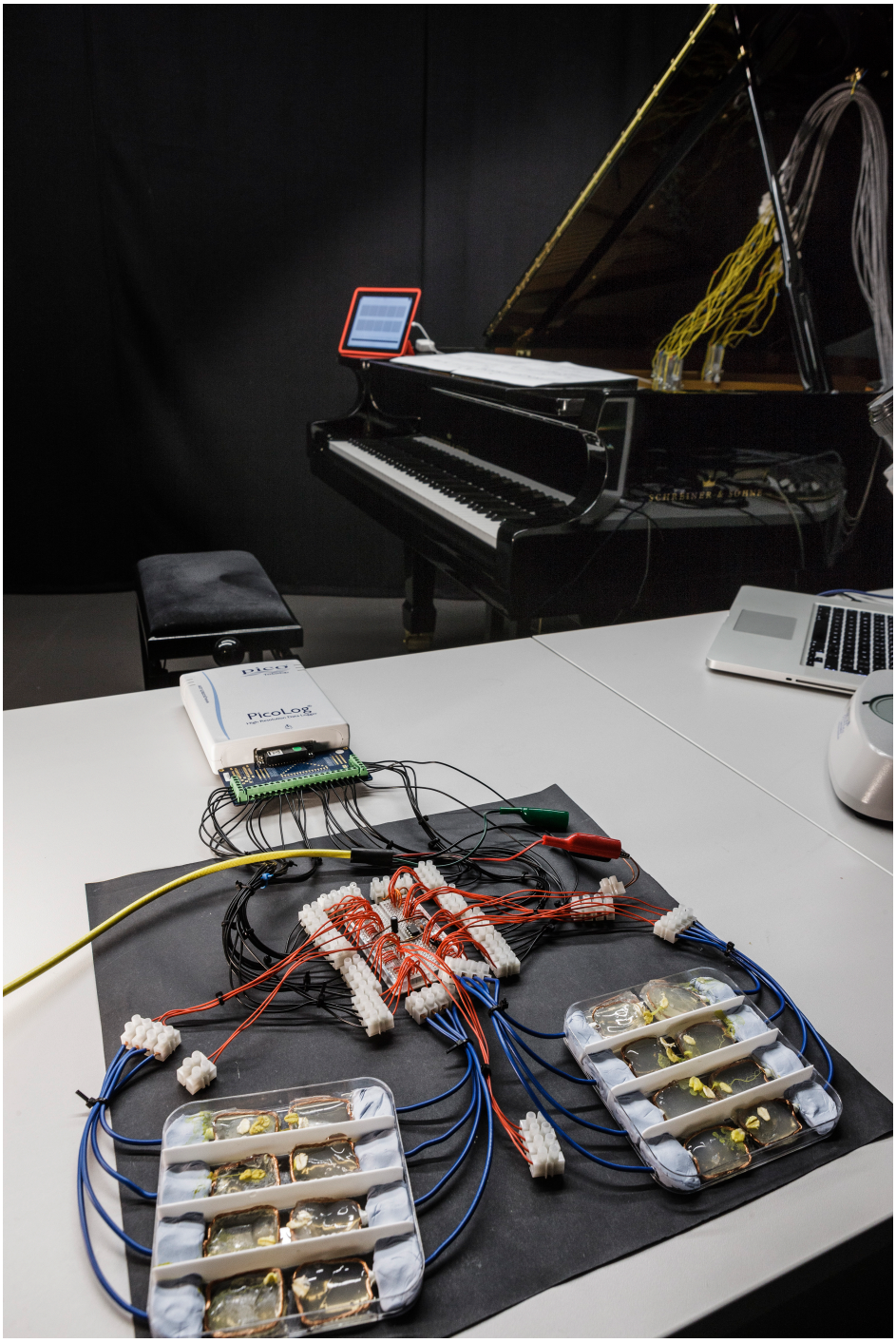

Figure 29: The first version of ICCMR's interactive musical biocomputer.

Miranda has composed two unprecedented pieces of music for piano using two different instances of the interactive musical biocomputer. The first, entitled *Biocomputer Music*, was premiered at PACMF 2015, Plymouth, UK. Here, the biocomputer listens to the pianist and generates musical responses in real-time, which are played on the same piano through electromagnets that set its strings into vibration (Figure 30). The second, entitled *Biocomputer Rhythm*, was composed for the 2016 edition of PACMF, where the biocomputer also plays percussion instruments. Here the biocomputer was programmed with the ability to learn how to produce sequences of musical responses based on the sequences played by the pianist.

Figure 30: The biocomputer plays the piano through electromagnets placed close to the strings.

A recording of *Bicomputer Music* is available through SoundCloud (Miranda 2014) and video documentaries introducing the research and both compositions are available through Vimeo (Miranda 2015, Miranda 2016).

2.7 Concluding Remarks and Further Work

Research into UC for sound and music is no longer in its infancy, but it is far from mature yet. Most developments are still in the proof of concept stage, but this has

not deterred the Computer Music community to develop musical experiments, and produce compositions and performances at professional level.

One important vein of research that we are developing at ICCMR concerns the development of the biomemristor, which is discussed in more detail in Chapter 8. To this end we are working towards gaining a better understanding of *P.polycephalum*'s material make up and the parameters for handling its memristance. What is exciting about this organism is that it is a biological system that displays complexities, which might be harnessed to implement different classes of memristors or variations thereof (Chua 2015).

Once the components of *P.polycephalum* are better identified and characterised, a natural progression would be to investigate the possibility of engineering, using these biological components, a comparable system to operate *in vitro* rather than *in vivo.* In doing this, we may be able to reduce system volatility and standardise components by re-engineering some of the existing biological elements. In essence, this is a Synthetic Biology approach. Synthetic Biology is an engineering approach to biology involving the rational design of devices and systems using biological materials and components of known technical specifications. An additional benefit of producing components this way is the ability to engineer elements that do not exist in nature. By this we mean add components that are not part of the natural biological system. This could involve, for example, coupling the electrical conducting system to various actuators. Adding other electrical components such as different types of switches (toggle, selector, proximity) should also be possible. Success here would widen the application of *P.polycephalum*-based biomemristors.

Another important vein of research concerns the development of suitable encoding methods to represent musical information on biomemristors and develop methods by which the system processes and generates music, which flesh out the non-linear analogue nature of those components. We believe that research at this front will also shed light on musical representation and processing for other modalities of UC, such as quantum computing music.

2.8 Acknowledgements

Most of the work presented in this chapter was developed in collaboration with colleagues within our University and beyond. We thank Antonino Chiaramonte, Anna Troisi, John Matthias, Nick Fry and Cathy McCabe of Plymouth University for their contribution to the musical experiments with particle physics. Larry Bull and Ivan Uroukov, at University of West of England, Bristol, and ICCMR post-graduate student Francois Gueguen, contributed to the work with *in vitro* neural networks.

2.9 Questions

1. Why are Cellular Automata suitable for modelling UC?
2. Are there any other methods for simulating UC on conventional computers? If so, give an example of such simulation.

3. Can you think of ways of using the Cellular Automata reaction-diffusion model to generate musical sequences instead of synthesise sounds?
4. Why it was necessary to apply data compression to the data produced by the *in vitro* neuronal networks?
5. How does the violin interact with atomic particle tracks in the *Cloud Chamber* experiment?
6. What is the main barrier that computer musicians face in order to experiment with UC?
7. What is *Physarum polycephalum* and why this organism is suitable for research into UC and music?
8. Why has the granular techniques to synthesise sounds been adopted so widely in the works presented in this chapter?
9. What is a memristor and why is the UC community excited about it?
10. What is Synthetic Biology and how does this connect with research into UC?

2.10 References

Adamatzky, A. (Ed.) (2016). *Advances to Unconventional Computing*. Vols 1 & 2. Springer International Publishing.

Adamatzky, A. (2010). "Advances in Physarum Machines Gates, Hulls, Mazes and Routing with Slime Mould". In Bosschere et al. (Eds.) *Applications, Tools and Techniques on the Road to Exascale Computing*. IOS Press, pp. 41-55.

Adamatzky, A. (2007). "Physarum Machine: Implementation of a Kolmogorov-Uspensky Machine on a Biological Substrate." *Parallel Processing Letters* 17(04):455–467.

Adamatzky, A., Costello, B., Melhuish, C. and Ratcliffe, N. (2003). "Experimental Reaction-Diffusion Chemical Processors for Robot Path Planning", *Journal of Intelligent Robotic Systems* 37(3): 233-249.

Armstrong, R. and Ferracina S. (Eds.) (2013). *Unconventional Computing: Design Methods for Adaptive Architecture*. Riverside Architectural Press.

Aspray, W. (1990). *John von Neumann and the Origins of Modern Computing*. The MIT Press.

Block, I., and Wohlfarth-Bottermann, K. (1981). "Blue Light as a Medium to Influence Oscillatory Contraction Frequency in Physarum." *Cell Biology International Reports* 5(1):73–81.

Bontorin, G., Renaud, S., Gerenne, A., Alvado, L, Le Masson, G. and Thomas, J. (2007). "A Real-Time Closed-Loop Setup for Hybrid Neural Networks", *Proceedings of 29th Annual Conference of the IEEE Engineering in Medicine and Biology Society*, Lyon, France, pp. 3004-3007.

Caleffi, M., Akyildiz, I. F. and Paura, L. (2015). “On the Solution of the Steiner Tree NP-Hard Problem via Physarum BioNetwork”, *IEEE/ACM Transactions on Networking* 23(4):1092-1106.

Chua L. (2015). “Everything You Wish to Know About Memristors But Are Afraid to Ask”, *Radioengineering* 24(2):319-368.

Chua, L. (1971). ‘Memristor-the missing circuit element. Circuit Theory’. *IEEE Transactions*, 18(5):507-519.

DeMarse, T., Wagenaar, D. A., Blau, A. W. and Potter, S. M. (2001). “The Neurally Controlled Animat: Biological Brains Acting with Simulated Bodoies”. *Autonomous Robots* 11(3):305-310.

Doornbusch, P. (2004). “Computer Sound Synthesis in 1951: The Music of CSIRAC”*, Computer Music Journal*, 28(1):10-25.

Ellis, J. (2006). “Standard Model of Particle Physics”. In *Encyclopedia of Astronomy & Astrophysics*. IOP Publishing.

Gale, E., A. Adamatzky, and Costello, B. (2014). ‘Are Slime Moulds Living Memristors?’. Available online at: http://arxiv.org/abs/1306.3414 (Last visited 31/10/2014).

Gale, E., Matthews, O., Costello, B. and Adamatzky, A. (2013). ‘Beyond Markov Chains, Towards Adaptive Memristor Network-based Music Generation’. Available online at: http://arxiv.org/abs/1302.0785. (Last visited 10/10/2014)

Jones, J. (2010). ‘The Emergence and Dynamical Evolution of Complex Transport Networks from Simple Low-Level Behaviours*’.* I*nternational Journal of Unconventional Computing*, 6(2):125-144.

Kamioka, H., Maeda, E., Jimbo, Y., Robinson, H. P. C., Kawana, A. (1996). “Spontaneous Periodic Synchronized Bursting During Formation of Mature Patterns of Conectionsin Cortical Cultures.” *Neuroscience Letters* 206(1-2): 109-112.

Kike, A., Chiaramonte, A., Troisi, A. and Miranda, E. (2013). *Could Chaber: A Duet for Violin and Subatomic Particles*. Movie on YouTube: http://www.alexiskirke.com/#cloud-chamber

Kirke, A., Miranda, E., Chiaramonte, A., Troisi, A. R., Matthias, J., Radtke, J., Fry, N., McCabe, C. and Bull, M. (2011). “Cloud Chamber: A Performance Involving Real Time Two-way Interaction Between Subatomic Radioactive Particles and Violinist”, *Proceedings of International Computer Music Conference* (ICMC 2011), Huddersfield (UK).

Kolmogorov, A. N. and Uspenskii, V. A. (1958). “On the Definition of an Algorithm.” *Uspekhi Matematicheskikh Nauk* 13(4):3–28. (In Russian) English translation in *AMS Translations* 1963 Series 2, Vol. 21, pp. 217-245.

Meyer, R. and Stocking, W. (1970). “Studies on microplasmodia of *Physarum polycephalum* V”, *Cell Biology International Reports* 3(4): 321–330.

Miranda, E. R. (2016). *Music Biocomputing*. Video documentary on Vimeo: https://vimeo.com/163427284 (Last accessed on 20 Sep 2016).

Miranda, E. R. (2015). *Biocomputer Music: A Composition for Piano & Biocomputer*. Video documentary on Vimeo: https://vimeo.com/111409050 (Last accessed on 20 Sep 2016).

Miranda, E. R. (2014). *Biocomputer Music*. Musical composition on SoundCloud: https://soundcloud.com/ed_miranda/biocomputer-music (Last accessed on 10 Sep 2016).

Miranda, E. R., A. Adamatzky, and J. Jones (2011). “Sounds Synthesis with Slime Mould of Physarum Polycephalum”. J*ournal of Bionic Engineering*, 8(2):107-113.

Miranda, E. R. (2002). *Computer Sound Design: Synthesis Techniques and Programming*. Elsevier/Focal Press.

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

Miranda, E. R. (1994). *Olivine Tress*. Musical composition on SoundCloud: https://soundcloud.com/ed_miranda/olivine-trees (Last accessed on 10 Sep 2016).

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

Newby, G., Hamley, I., King, S., Martin, C., Terrill, N. (2009). “Structure, rheology and shear alignment of Pluronic block copolymer mixtures,” *Journal of Colloid and Interface Science* 329(1).

Nicolis, G. and De Wit, A. (2007). *Raction-diffusion systems*. Scholarpedia 2(9):1475. Available on-line: http://www.scholarpedia.org/article/Reaction-diffusion_systems (Last accessed on 21 Sep 2016)

Novellino, A., D’Angelo, P., Cozzi, L., Chiappalone, M., Sanguineti, V., and Martinoia, S. (2007). “Connecting Neurons to a Mobile Robot: An In Vitro Bidirectional Neural Interface”. *Computational Intelligence and Neuroscience* Vol. 2007, Article ID 12725.

Pershin, Y.V., S. La Fontaine, and Di Ventra, M. (2009). “Memristive model of amoeba learning”. *Physical Review E* 80(2).

Potter, A. M., DeMarse, T. B., Bakkum, D. J., Booth, M. C., Brumfield, J. R., Chao, Z., Madhavan, R., Passaro, P. A., Rambani, K., Shkolnik, A. C., Towal R. B. and Wagenaar, D. A. (2004). “Hybrots: Hybrids of Living Neurons and Robots for

Studying Neural Computation", *Proceedings of Brain Inspired Cognitive Systems*. Stirling, UK.

Radtke, J. (2001). *Diffusion Cloud Chamber Owner's Guide*, Version 2.5. Reflection Imaging.

Shu J. J., Wang, Q. W., Yong, K. Y., Shao, F. and Lee, K. J. (2015). "Programmable DNA-Mediated Multitasking Processor", *The Journal of Physical Chemistry* 119(17):5639-5644.

Strukov, D. B., Sneider, G. S., Stewart, D. R. and Williams, R. S. (2008). "The missing memristor found"*. Nature*, 453:80-83.

Swade, D. (1991). *Charles Babbage and his Calculating Engine*s. London Science Museum.

Truax, B. (1988). "Real-time granular synthesis with a digital signal processor", *Computer Music Journal*, 12(2):14-26.

Turing, A. M. (1936). 'On computable numbers, with an application to the Entscheidungsproblem'*. Proceedings of the London Mathematical Society*, 42(2):230-265.

Uroukov, I., Ma, M., Bull, L. & Purcell, W. (2006). "MEA Recordings of the Spontaneous Behaviour of Hen Embryo Brain Spheroids". In *Proceedings of the 5th International Meeting on Substrate-Integrated Micro Electrode Arrays*. BIOPRO, pp232-234.

Valle, A. and Lombardo, V. (2003). "A two-level method to control granular synthesis", *Proceedings of the XIV Colloquium on Musical Informatics (XIV CIM 2003)*. Firenze, Italy.