Physically inspired interactive music machines – making contemporary composition accessible?

Richard Polfreman
Music Department
University of Hertfordshire
r.p.polfreman@herts.ac.uk

Martin Loomes
Computer Science
University of Hertfordshire
m.j.loomes@herts.ac.uk

Robert Wright
Music Department
University of Hertfordshire
r.e.wright@herts.ac.uk

Abstract

Much of what we might call 'high-art music' occupies the difficult end of listening for contemporary audiences. Concepts such as pitch, meter and even musical instruments often have little to do with such music, where all sound is typically considered as possessing musical potential. As a result, such music can be challenging to educationalists, for students have few familiar pointers in discovering and understanding the gestures, relationships and structures in these works. This paper describes on-going projects at the University of Hertfordshire that adopt an approach of mapping interactions within visual spaces onto musical sound. These provide a causal explanation for the patterns and sequences heard, whilst incorporating web interoperability thus enabling potential for distance learning applications. While so far these have mainly driven pitch-based events using MIDI or audio files, it is hoped to extend the ideas using appropriate technology into fully developed composition tools, aiding the teaching of both appreciation/analysis and composition of contemporary music.

1. Introduction

The nature of much contemporary music composition is unfamiliar when judged against the traditional concepts of pitch, musical keys, rhythm, etc. Any sound can now be used in a musical context, developing from the Musique Concrète of Pierre Schaeffer in the 1950's, while atonal (12-tone) works had abandoned musical keys long before. Moreover, information technology has taken over from the recording studio, and much of what we listen to everyday is produced on computers. Although there are many software applications for traditional/popular music creation and tuition, accessibility of more 'avant garde' styles remains difficult, with few tools to aid the educator. Software by IRCAM allowing children to manipulate sound as typically used in electroacoustic music [1] and by INA-GRM providing an history of electroacoustic music, together with analyses and interactive sound manipulation tutorials [2] provide some useful materials in this direction, but do not necessarily make it easy for students to understand the music itself.

In previous work we have taken Task Analysis techniques from the domain of HCI and applied these to music composition [3]. The resulting generic task model has been used to aid the development of software tools for composers. That research, however, did not concern itself with listeners' understanding of music. In this new work we introduce bi-modal systems, where the visual element provides a pseudo-physical explanation of the music being heard, by presenting a plausible causation mechanism to the listener. While experiments created with Macromedia's Flash™ provide only limited user interaction with the composition engine, our Agentsheets projects allow students to construct their own composition spaces using predefined building blocks, and potentially to extend behaviours and add new types.

Unlike typical visualizations of music, such as those in Microsoft's Windows Media Player or Apple's iTunes, here the visual and audio counterparts are generated from the same calculations (at the event level). We are certainly not the first to enter this region of combined visual and audio (multimedia) art works using off-the-shelf technologies, many examples can be found on the Internet [4]. However, in our work the graphical element serves primarily as a novel description of generative processes for composition (i.e. the application of pseudo-physical motions), and is not claiming artistic interest in its own right. This is opposed to the general multimedia approach, where both aspects claim artistic merit and are not usually integrated by a shared causal link but via interpretative means. Often the associations are deliberately obscure, and have to be discovered by the audience, while in our work the link has to be as explicit as possible. Some work by others does fall into a similar category to ours, e.g. Music Pond by Adrian Ward [5], where the user initiates ripples on a surface that strike objects in the 'pond' triggering MIDI notes

Proceedings of the The 3rd IEEE International Conference on Advanced Learning Technologies (ICALT'03)
0-7695-1967-9/03 $17.00 © 2003 IEEE

Algorithmic composition, where generative processes are used to specify musical events, is not new and several software tools exist for aiding the composer in this field. Examples include Common Music [6], Symbolic Composer [7] and OpenMusic [8]. We have developed a library for controlling physical model synthesis within OpenMusic [9] and so have some experience in this area. These systems typically involve the application of both mathematical and programming knowledge on the part of the composer (all three cited are Lisp based), which makes these challenging environments for music students.

One system we have already developed, FrameWorks [10], shares some properties with algorithmic composition systems, the inclusion of transformations of musical materials, but without requiring the composer to utilise mathematical or programming skills for the task. However, this program has yet to adopt generative mechanisms, the design not intended as a complete algorithmic system, but more concerned with supporting the editing of musical structure.

The new systems we are currently developing provide explicit process models applied to music generation, allowing students to think of music composition in new ways, helping them to escape from the more traditional and music theoretical approaches towards a physicality of gesture and relationships. This in turn should aid the development of students' listening skills when presented with such music. Our intention is to do this without recourse to standard algorithmic approaches with their associated programming and mathematical requirements, rather to make use of students' experience of physical/spatial behaviours in creating the music.

2. Approach 1: Using Flash™

Our first series of experiments utilised Macromedia's Flash which provides a technology for interactive, web deliverable, multimedia content that is relatively simple to use and with a few audio features that, while somewhat limited, provide sufficient functionality for some interesting music applications.

2.1. Wind Chime Marimba, Flash Player Piano

Wind Chime Marimba (WCM) and Flash Player Piano (FPP) [11] are both aleatoric compositions, i.e. are based upon chance events. They are both infinite as the composition engines will run indefinitely. Figure 1 shows the generation system for WCM. The basis for event triggering here is the motion of a trigger ball and other balls bouncing around an enclosed two-dimensional space. The larger trigger ball (containing the quaver symbol) will play a particular marimba note depending upon which of the other balls it hits. The smaller balls all appear identical, so the listener does not know which pitch will play, only that a note is about to occur. These smaller balls do not interact with each other, but simply pass over one another.

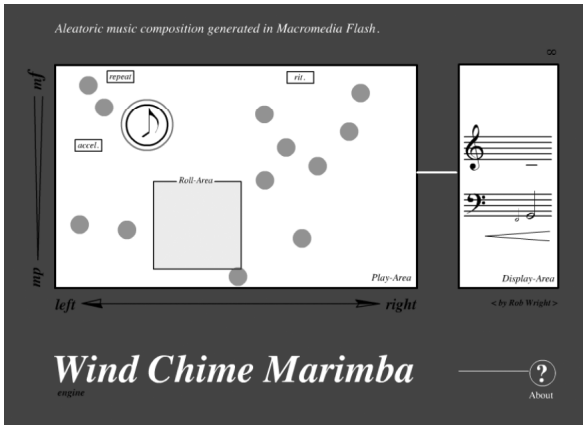

Figure 1. Wind Chime Marimba showing composition engine

There are also several modifier objects moving around the space. The objects *accel* and *rit* speed up or slow down the movement of the trigger object. If collisions occur while the trigger intersects with the large roll area, then an octave roll is played rather than a single hit. Finally the repeat object is toggled on and off by contact with the trigger, and causes a delayed repeat of events triggered since it was turned on. In the figure, the trigger has just been struck by an object and radiating (red) circles are shown to indicate this. The location of a collision within the space determines the amplitude (louder at the top) and pan (left to right) of the sound triggered. Initially the directions and speeds of the objects in the space are randomised.

On the right hand side of the display is an instantaneous musical representation of the events based on standard Western notation, with additions to indicate the ro n the stave les to nothi e the engin ut to happe any partic s the engin n. In this r ening exper ation about rn of event ee to place rhaps perce icitly presented by the aleatoric processes involved. It may be difficult for the listener to build up an adequate model of the composition simply by listening, whereas the listener who can see the generation process is provided with a complete visual explanation for what they are hearing.

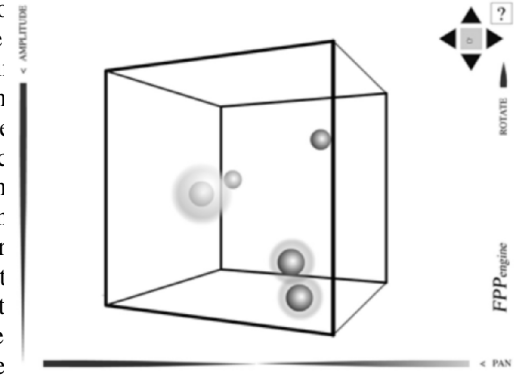

Figure 2. Flash Player Piano composition engine

Flash Player Piano is similar to WCM, but extends the space into three-dimensions and uses a slightly different triggering mechanism. As before the engine may be hidden or shown, but there is no notation view in this example. Figure 2 shows FPP with the composition engine revealed. There are five moving spheres enclosed in a 3D cube. Each face of the cube represents a piano (or occasionally bass) sample from a 6-note chord. When a sphere strikes a face, the corresponding note is triggered, with amplitude and pan derived from the X-Y position of the collision. If two spheres collide, a leading sample (for example, a descending series of notes) is played and the six notes change to a different set of pitches. The listener can control rotation of the cube itself in both horizontal and vertical directions. Since the position of a collision in X-Y determines amplitude and pan, this can have some effect over the resultant music, although only in relatively minor ways.

2.2. Pattern Chain

This example blurs the distinction between instrument and generative composition. The instrument part sets up a marimba-like tuned percussion, with a series of small squares indicating the ends of bars, and coloured black or white according to a piano keyboard layout (figure 3). A series of red circles represent beaters, and the user can specify from 1 to 12 to be present. There are two main models for controlling the motion of the beaters (composition part), one uses a chain of elastic connections between them and the edges, as shown in figure 3, while the other keeps them independent, hanging on elastic threads pendulum style. Either way, the user can click-and-drag a beater and then let go, the elastic forces then pulling the beater to return to its rest state. Given the elasticity, once a beater is set in motion it will set up a rhythmic pattern where every time it crosses a bar on the 'marimba', it will trigger the appropriate pitch. The user can choose to display on a score at the top of the screen, which notates the musical events as they occur.

In the chain mode, moving just one of the beaters will create movement in the others, resulting in complex patterns of notes. In order to make the program more compositionally useful, the user has a few other controls in addition to changing the type and number of beaters. First, the horizontal positions of the beaters can be fixed, and so they will only trigger particular pitches. In this way, each beater can be dragged horizontally so it is above the required note, and then set in motion in order to create patterns of deterministic pitch sets. Second, event triggering can be set to up and down across the bars (default) or down only. Finally a damping factor can be switched on so that the movements of the beaters dies away to nothing, and new patterns have to be initiated by the user.

While currently it is difficult to set up precise rhythms that stay in phase, as would be required in serious composition tool, the principle of the generation system is easy to understand and a more sophisticated adaptation of the interface would allow such control to be imposed.

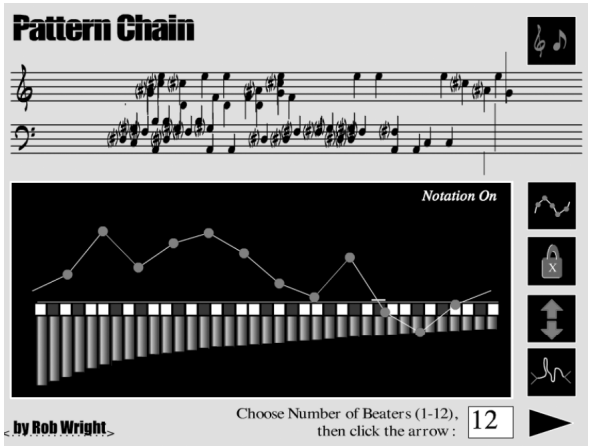

Figure 3. Pattern Chain – chain of moving beaters striking percussive bars

3. Approach 2: Agentsheets

While Flash offers many advantages for developing these types of interactive systems, significant development time is required for each program, and only certain user control can be easily built-in. Agentsheets [12] is a program designed to enable students to construct their own simulations. Users design their own graphical *agents*, placing them on a 2D grid in which they can interact. A set of *behaviours* are defined for an agent type, consisting of *methods* containing one or more rules, each rule of the form *if condition(s) then action(s)*. The advantages of the system when compared to Flash/ActionScript include its simplicity, lack of programming skills required and ease of editing, including end-user customisation and extension. The end result can be published to a Java Applet for Internet distribution (although there are some limitations with this currently). Disadvantages include limited programming constructs and graphic representation. Our interest in the system lies in the possibilities for very rapid prototyping of composition mechanisms, which can be very quickly modified in the light of user evaluations. Also, we hope to leverage pedagogical benefits of the system into the music composition domain, since Agentsheets (and its *tactile programming* language Visual AgentTalk) has been designed for educational applications and refined through testing with students in practice [12].

Figure 4 shows an Agentsheets Worksheet using agents based on the WCM and FPP approaches. The active agents are the circular objects. These move around the

space bouncing off the walls (but not each other), triggering MIDI notes currently, but they could play sound files. Since the agent and grid sizes are the same, the agents can only move in 8 possible directions (45 degrees between each). Each striker can have its own pitch and speed of movement. The default behaviour is for a striker's note (and graphic image) to fade each time a collision occurs until the MIDI velocity reaches zero and the agent is removed from the scene. This behaviour can be over-ridden, either to have no fading or for agents to re-spawn instead of being removed. Blue walls bounce the strikers in an inelastic manner (i.e. the speed of the striker remains the same), while red walls impart energy to the strikers, accelerating them. There are also purple walls that dampen strikers and green walls that are inelastic but direct the striker to play an octave roll rather than a single note. Each striker belongs to a group, and individual groups can be started/stopped independently. The default is for all groups to be active.

In figure 4, the strikers along the bottom belong to one group, those at the top to another. The bottom group have start velocities in the direction up and to the right. This configuration sets up a repeating pattern of notes: two overlapping broken chords as each striker hits the left wall and diagonal wall in turn, followed by the chord all at once as they all end up in a column hitting the right wall simultaneously.

From this vertical alignment, a similar motion begins ending up back in the original row. This group then provides a constant 'accompaniment', while the 'melody' is created by the group of strikers at the top. The pitches of the bottom group are a series of rising fifths, while the top group uses tonic, major third, fourth, fifth, octave, octave + major third. The light grey (red) wall along the top makes each striker in this group accelerate each time an impact with the wall occurs. Each striker has a different distance between walls, the right-most (shortest distance) striker accelerates very rapidly and fades quickly, followed by each striker in turn moving to the left. If the re-spawn option is on, an infinite composition is produced, the accompaniment fading to nothing and beginning again, with the continual phasing of the melodic pitches above, accelerating to very rapid oscillations and then beginning once again.

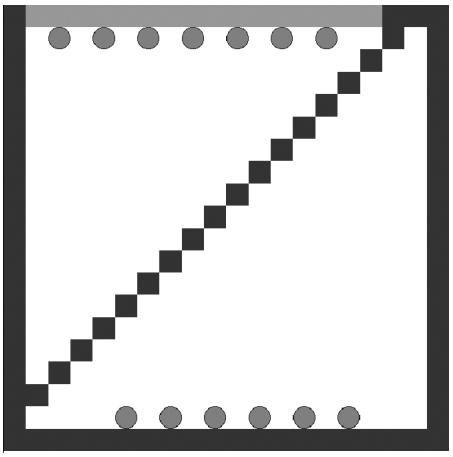

Figure 4. Agentsheets (Musical) Worksheet

The key aspect to the Agentsheets work is that once the agents have been defined the user can then fill a worksheet space with whatever configuration of agents they decide. In this case, any pattern of walls can be constructed and then striker agents placed within to produce the desired result – the composer being able to freely experiment with ideas. The user can also 'perform' with the worksheet, treating it as an instrument and dynamically inserting agents in specific places at specific times, or adding/removing walls to create different patterns of movement.

Various 'simulation properties' have been defined for these examples and are used to set global settings and default values for agents. These include 'skip' (default speed for strikers), 'on' (default on/off state for strikers), 'pitch values' etc. Taking advantage of further capabilities of Agentsheets, a property *getWeather* is specified. If this flag is set, when a striker agent is placed on a worksheet, the agent will connect to the Internet and take its pitch from the current temperature in Fahrenheit of one of seven places around the world named 'London'.

Another agent included is called Gstart. This agent acts as a switch that can toggle on or off a group of strikers. This agent also acts as a wall, in that strikers will bounce off it (inelastically), and when struck by a striker the Gstart toggles its on/off state. With this agent present in a worksheet, not only can the user intervene during a performance, manually invoking or cancelling sets of moving agents, but also striker agents can set off or stop other agents, allowing complex machineries of music generation to be constructed. Figure 5 shows a worksheet in which there are five striker groups in separate areas. A member of each group will bounce on a switch that controls the on/off state of the next group, clockwise from the left. By using different pitch sets and/or wall geometries in different striker groups within a worksheet, Gstart agents offer the potential for generation systems that instigate both rhythmic and key/mode changes.

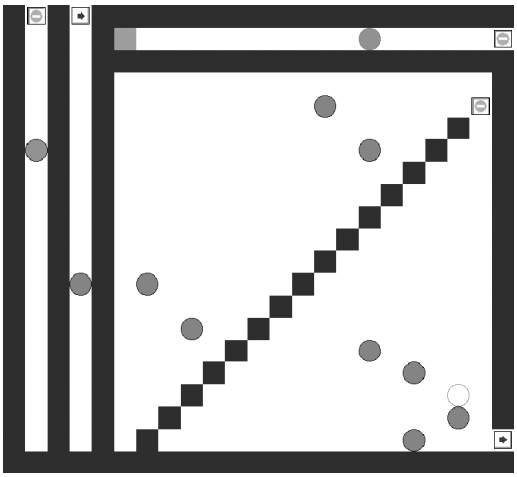

Figure 5. Agent switching – each area contains a striker agent that can switch the next group on/off.

4. Conclusions and Further Work

It should be stressed that this work is still in its early stages, and our purpose in this paper has been to present ideas and promote interest in the approach. We are currently prototyping, using both Flash and Agentsheets, a wide range of objects and interactions providing users with interesting choices of generation mechanisms prior to evaluation within educational settings. We are also continuing to probe the technical limitations of the approaches and finding ways to overcome these.

Our Flash prototypes, for example, lack facilities for serious construction of a composition by the user. In the aleatoric compositions only minor alteration by the user is possible, and essentially the music is the product of the composer's intent in designing the Flash program. Pattern Chain moves further towards the user defining their own composition, or compositional elements, but currently lacks the necessary precision and ability to change sound sets (or drive MIDI).

With Agentsheets it is simple to provide the user with great flexibility and the facility to build their own composition spaces. Indeed, the environment is such that students should be able to investigate the details of the agent behaviours and to edit/extend these as they please. In the work so far only one physical model is used (i.e. that of bouncing balls), and further mechanisms await exploration, e.g. springs, connecting rods or rotary systems, although there are graphical limitations to be considered. The Internet connection offers interesting possibilities within an educational environment, for example, the teacher could place different data on a given web page at different times and so change the compositional material available to students (for example using different musical keys or pitch sets).

A key issue with both development systems is that of their musical limitations, both in terms of general capabilities (e.g. one cannot control duration of MIDI notes in Agentsheets) and in the matter of timing. Neither system can provide guaranteed timing of events, and both typically slow down according to the number of interacting objects within a space. To achieve effective educational tools this issue must be addressed and so we will exploring the use of other technologies, e.g. Macromedia's Director / Shockwave system, together with any necessary music Xtra's (software extensions) such as IRCAM's jMax Xtra.

References

[1] "10 Jeux d'Écoute" interactive CR-ROM, IRCAM - Centre Georges Pompidou / Hyptique, 2000.

[2] "La Musique Electroacoustique" interactive CD-ROM, INA – GRM / Hyptique, 2000.

[3] R. Polfreman, M.J. Loomes, "A TKS Framework for Understanding Music Composition Processes and its Application in Interactive System Design", *Proceedings of the AISB01 Symposium on Creativity in Arts and Sciences*, The Society for the Study of Artificial Intelligence and the Simulation of Behaviour, 2001, pp75-83.

[4] www.soundtoys.net

[5] www.generative.net/products/musicpond/

[6] H. Taube, "Common Music: A Music Composition Language in Common Lisp and CLOS", *Computer Music Journal*, MIT Press, **15**(2), 1991, pp 21-32.

[7] http://www.mracpublishing.com/scom

[8] G. Assayag, C. Rueda, M. Laurson, C. Agon, O, Delerue, "Computer-Assisted Composition at IRCAM: From PatchWork to OpenMusic", *Computer Music Journal*, MIT Press, **23**(3), 1999, pp 59-72.

[9] R. Polfreman, "Modalys-ER for OpenMusic (MfOM): Virtual Instruments and Virtual Musicians". To be published in: *Organised Sound* **7**(3), CUP, 2002.

[10] R. Polfreman, "Supporting Creative Composition: the FrameWorks Approach", *Les Actes des 8e Journées d Informatique Musicale*, Institut International de Musique Electroacoustique, 2001, pp 99-111.

[11] R.E. Wright, "Internet Audio Experiments", *New Notes Journal*, Society for the Promotion of New Music, October 2002, pp 1-2.

[12] Rader, C., G. Cherry, C. Brand, A. Repenning, et al., "Principles to Scaffold Mixed Textual and Iconic End-User Programming Languages," *Proceedings of the 1998 IEEE Symposium of Visual Languages*, Nova Scotia, Canada, Computer Society, 1998, pp. 187-194.