STUDENT ARTICLE

TAO: a physical modelling system and related issues

MARK PEARSON

Department of Electronics, University of York, York YO1 5DD, England

This paper describes TAO, a system for sound synthesis by physical modelling based on a new technique called *cellular sound synthesis* (CSS). The system provides a general mechanism for constructing an infinite variety of virtual instruments, and does so by providing a virtual acoustic material, elastic in nature, whose physical characteristics can be fine-tuned to produce different timbres. A wide variety of sounds such as plucked, hit, bowed and scraped sounds can be produced, all having natural physical and spacial qualities. Some of the musical and philosophical issues considered to be most important during the design and development of the system are touched upon, and the main features of the system are explained with reference to practical examples. Advantages and disadvantages of the synthesis technique and the prototype system are discussed, together with suggestions for future improvements.

1. INTRODUCTION

This paper describes TAO, a system for sound synthesis by physical modelling. It is intended to be a tool for composers of electroacoustic music and was inspired by the beauty and simplicity of creating physical sounds in the real world. We are all capable of producing an astonishing variety of complex sounds by physical means, with no particular expertise or knowledge of acoustics or physics. One of the guiding ideas throughout this project has been that it ought to be possible to take a similar approach to creating sounds with a computer synthesis system.

It should be pointed out from the beginning that music is not about technology but about a deep love of sound in all its rich variety, and getting overly excited about technology for technology's sake is not conducive to creating rich, exciting sounds for electroacoustic music. A composer should make no compromise when it comes to the sources of material from which a composition is to be constructed. For this reason, in addition to giving a straightforward description of the TAO system this paper also covers several issues which have been considered crucial to its design and development.

2. THE INSPIRATION BEHIND TAO

Natural sounds, in addition to having basic spectral qualities, have a physical dimension. We feel them accelerating, dissipating, exploding, smashing, scraping, etc. The qualities described by these adjectives are difficult if not impossible to quantify in terms of spectral information since they are the result of subtle time domain information contained within the sounds. Natural sounds never repeat themselves exactly although sounds produced by one object will always inhabit a certain region of timbral space. This strong identity of acoustic objects (not necessarily just musical instruments) combined with their ability to produce fresh sounds all the time is one of their most appealing characteristics.

Chaos theory (Gleick 1991) and complexity theory (Waldrop 1994) have also had a part to play in the inspiration behind the system. Both of these theories have helped to encourage a paradigm shift, which has led to a world view in which greater importance is attached to the global nature of complex systems. Interrelationships between constituent parts of such systems can give rise to very complex emergent patterns of behaviour. These patterns are often capable of holding our attention for long time spans since they have a life of their own and are full of surprises.

Complexity has several ramifications for computer modelling of real systems. We have to be careful which factors of a real system we decide to omit from a model since we can easily end up with a model which does not behave realistically except in simple uninteresting cases. In the context of physical modelling for sound synthesis, such incorrect assumptions can lead to sounds which are lacking in the depth, richness and expressive qualities of their real counterparts.

3. AN INTRODUCTION TO THE TAO SYNTHESIS SYSTEM

TAO consists of two distinct components, a synthesis engine based on a new physical modelling technique called *cellular sound synthesis* (CSS) and a text-based interface allowing the user to gain access to the synthesis engine. A virtual acoustic material is provided by the synthesis model, the physical attributes of which can be altered in order to fine tune its timbral properties. Instruments can be constructed from

Organised Sound **1**(1): 43–50 © 1996 Cambridge University Press

https://doi.org/10.1017/S1355771896000167 Published online by Cambridge University Press

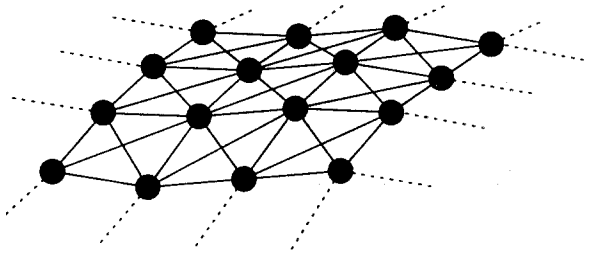


Figure 1. A close-up view of the virtual acoustic material used to create instruments.

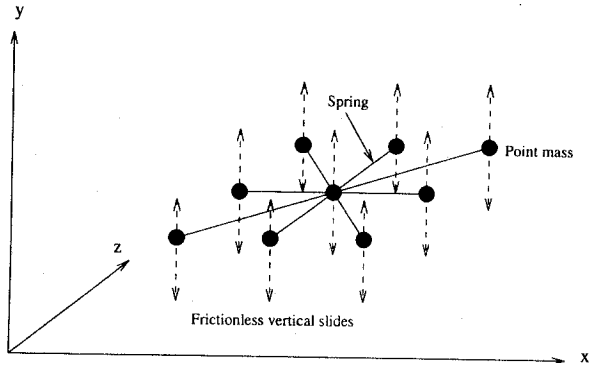


Figure 2. A single cell with its eight immediate neighbours.

several pieces of this material coupled together in the same way that real instruments are constructed from combinations of materials with different acoustic properties.

The script language allows the user to build instruments, declare various performance parameters, place virtual microphones at various locations in order to produce sound output, and finally provides a score language for describing the performance itself. Once an instrument has been created, it exists for the duration of the score during which it can be excited in a variety of ways.

Instruments created within TAO are tangible physical objects which place no restrictions on where excitations are applied or how many excitations are applied at once. This is one of TAO's most important features and was considered musically essential from the outset of the project, since it allows a very wide variety of timbres to be produced by one instrument in ways which are intuitive to musicians. As with a real instrument these timbres will be closely related and have an overall character defined by the structure of the instrument.

4. CELLULAR SOUND SYNTHESIS: A PHYSICAL MODELLING TECHNIQUE

The material from which instruments are constructed consists of point masses arranged in a horizontal two-dimensional grid, mounted on frictionless vertical sliders and connected to their neighbours by spring-like elements. Each point mass (referred to as a cell from now on) has a vertical position, velocity and force and a damping factor which exerts a frictional force, proportional to the cell's velocity, dissipating energy and slowing the cell down. Figure 1 shows a close-up of a small region of the material and figure 2 shows a single cell connected to its eight immediate neighbours. From a distance the material appears to be continuous and elastic in nature and wave phenomena such as reflection, refraction and diffraction emerge from the model without being explicitly programmed. Figure 3 shows six successive images of a circular sheet of material whose whole perimeter is held still, after a single impulse has been applied to the top right of the sheet. Even such a simple excitation gives rise to a complex pattern of propagating and reflecting wavefronts. From the user's point of view a higher level of abstraction is provided, and individual cells and springs are not dealt with directly. Instead the system facilitates the creation of ready-made pieces of material of various shapes and sizes. At the time of writing these building blocks include one-dimensional pieces of material (like strings or pipes), and circular, rectangular and triangular sheets of material. The synthesis model is capable of dealing with irregularly shaped pieces of material, but these are currently not supported.

The instruments are played at the fundamental level by applying forces or damping to individual cells or groups of cells. For example, to simulate hitting an instrument with a sharp implement we can apply a force to a small group of cells over a short time interval. To simulate plucking a harmonic on a string, we can set the string in motion by simulated plucking, momentarily apply some damping at a node on the string and finally return the damping to its default value after a short time interval.

Each cell can either be free to move or locked, in which case it is fixed rigidly in one position. It is necessary to lock at least one cell in an instrument to prevent all the cells from drifting away from the zero position, at which they begin their life, when an excitation is applied. Locking one or more cells forces all the other cells to oscillate about the zero position. Sound output is generated by choosing a cell (or possibly several) and writing a time varying trace of its position to a soundfile.

5. THE INTERFACE

The present implementation of TAO provides a script language for describing instruments and performances. A script consists of two distinct parts: the orchestra part, containing declarations of instruments, microphones and performance parameters; and the score part, which allows the instruments to be played. An example script is shown in figures 4

https://doi.org/10.1017/S1355771896000167 Published online by Cambridge University Press

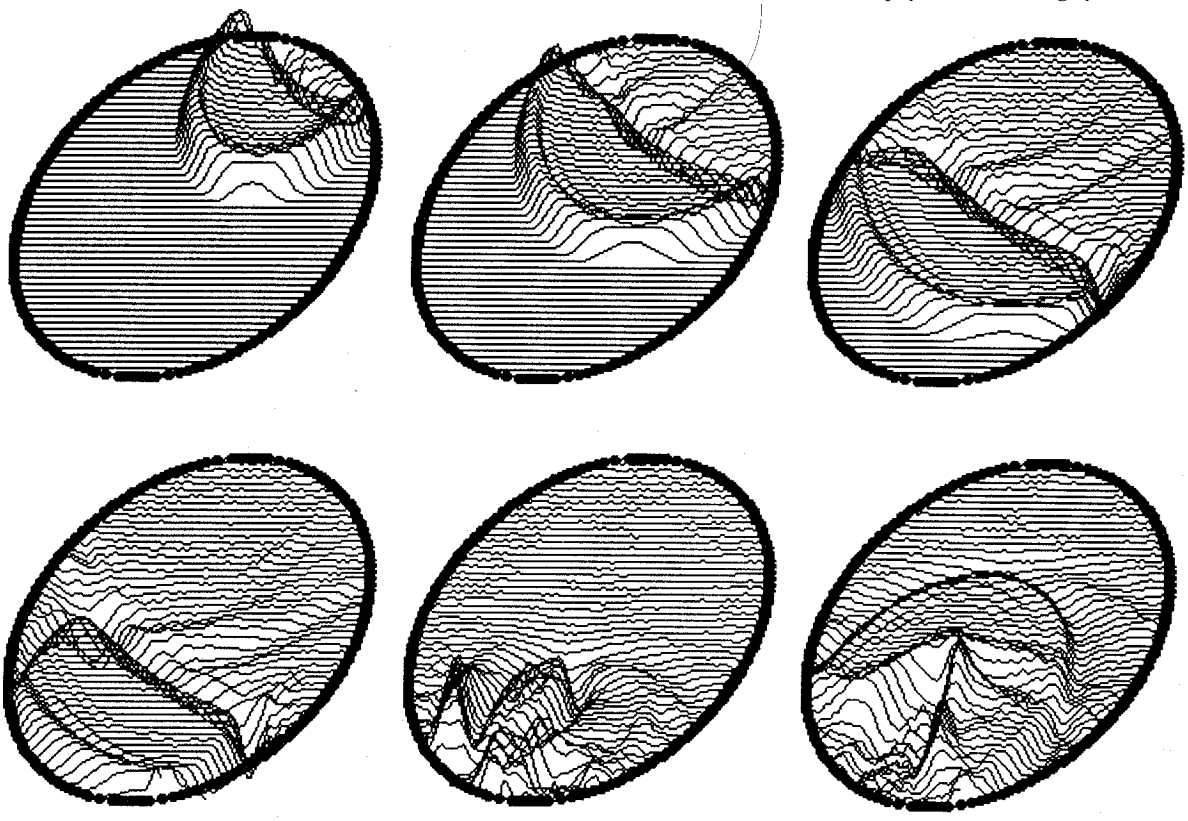


Figure 3. A circular piece of material after having been struck.

and 5. The orchestra and score are split for clarity in this paper but in reality they are contained within one text file. The orchestra declares some instruments, microphones and performance parameters, which are then used in the score, creating a performance lasting twenty seconds. There follows a detailed explanation of the structure of the script. Note that a `...` signifies the end of a block of instructions.

- Instrument declarations. Lines 1–16 create three instruments, `thestring`, `therectangle` and `thecircle`. An instrument declaration comprises a head and a body. The head specifies the shape required and the name of the new instrument. The body specifies the fundamental frequency (see section 6.2 for an explanation of the notation used) and overall decay time of the instrument and contains instructions specifying which parts of the instrument should be locked and, optionally, instructions which affect the amount of damping over regions of the instrument.

```
 1        String thestring:
 2            C#8, 10 secs;
 3            lockleft;
 4            lockright;
 5            ...
 6
 7        Rectangle therectangle:
 8            300 Hz, 450 Hz, 500 msecs;
 9            lock(left,top);
10            lock(right,top);
11            ...
12
13        Circle thecircle:
14            Bb6-1/2, 20 secs;
15            lockperimeter;
16            ...
17
18        Glue thestring(left) to thecircle(1/10,1/2);
19
20        Microphone mic1: file1, stereo, thestring(0.1), thestring(0.9);
21        Microphone mic2: file2, stereo;
22        Microphone mic3: file3, stereo;
23
24        Parameter now=0 secs, interval=1 secs, force, x, y;
```

Figure 4. A practical script example – the orchestra part.

https://doi.org/10.1017/S1355771896000167 Published online by Cambridge University Press

• Coupling instruments together. Line 18 causes the left end of the string to be glued to a point on the circular sheet one tenth of the way across and half way up.

• Microphone declarations. Lines 20–22 create three virtual microphones. The first microphone declaration specifies that `mic1` is a stereo microphone which takes its left and right signals from `the string` at points one tenth and nine tenths of the way from the left hand side, and writes its output to the file `file1`. `mic2` and `mic3` are also declared as stereo microphones but their output sources are determined by the score.

• Performance parameter declarations. Line 24 declares various performance parameters which are used in the score. Two of these, `now` and `interval`, are given initial values.

• Lines 25–64 in figure 5 comprise the score. The score consists of control structures which allow events to be scheduled at instants in time or over time intervals. Each control structure consists of a head and a body. The head specifies when the body, which may contain calculations, excitations to instruments, screen output, sound output or nested control structures, becomes active.

• Plucking the string. Lines 26–31 cause `the string` to be subjected to a force of magnitude one, for three milliseconds at a position one third of the way from the left hand side. The `At..for` control structure used is contained within the body of the `Score` control structure, and the keywords `start` and `end` refer to the start and end times specified in the head of the enclosing structure (in this case zero seconds and twenty seconds). The `Display` command allows text and data to be output to the screen during a performance.

• Playing a harmonic on the string. Lines 33–9 cause the third harmonic to be played on `the-string` three seconds into the performance by damping a point one third of the way along the string. The point is damped for one tenth of a second after which it is returned to its default value determined by the original decay time specified in the declaration of `thestring`.

• Striking the rectangle repeatedly. Lines 41–58 cause the rectangle to be repeatedly struck through-

```
25      Score 20 secs:
26          Before start + 3 msecs:
27                At start:
28                    Display "Pluck string at ", Time, "seconds", newline;
29                    ...
30              thestring(1/3).applyforce(1.0);
31              ...
32
33          At 3 secs for 1/10 secs:
34              At start:
35                  Display "damp node at ", Time, "seconds", newline;
36                  thestring.setdamping(1/3, 5%);
37                  ...
38              At end: thestring.resetdamping(1/3); ...
39              ...
40
41          ControlRate 100:
42              x=linear(0.1,0.9);
43              ...
44
45          At now for interval:
46              At start:
47                 y=random(0.1,0.9);
48                 force=random(1,10);
49                 ...
50
51              At start for 5 msecs:
52                 therectangle(x,y).applyforce(force);
53                 ...
54
55              At end:
56                 now+=interval;
57                 ...
58              ...
59
60          mic2.leftout:  thecircle(1/2,1/3);
61          mic2.rightout: thecircle(1/2,2/3);
62          mic3.leftout:  therectangle(1/10,1/5);
63          mic3.rightout: therectangle(9/10,4/5);
64          ...
```

Figure 5. A practical script example – the score part.

https://doi.org/10.1017/S1355771896000167 Published online by Cambridge University Press

out the performance at one second intervals. `therectangle` has a force `force` applied at a position `(x,y)` over a five millisecond interval at the beginning of each one second interval. `x` changes linearly from 0.1 to 0.9 over the duration of the score and is only calculated once every hundred samples to reduce overheads. `y` and `force` are determined randomly between the specified limits at the start of every new strike. `x` and `y` are normalised coordinates where zero means the left hand side or bottom and one means the right hand side or top of an instrument, respectively. The ability to specify how frequently a calculation should be made is reminiscent of audio-, control-, and initialisation-rate evaluation provided in the *Csound* audio processing language (Vercoe 1992).

- Creating sound output. The declarations of `mic2` and `mic3` in the orchestra provide no information about the actual sound sources for these two microphones. For this reason the output sources are explicitly given in lines 60–63 of the score. Note that the positions given can change dynamically throughout the performance if this method of specifying the sound sources is used, whereas `mic1` has a fixed sound source.

6. SUMMARY OF SCRIPT KEYWORDS

6.1. Instrument creators

The keywords `String`, `Rectangle`, `Circle` and `Triangle` are provided for the creation of instruments. `String` and `Circle` require one frequency to be specified determining the length of the string or the diameter of the circle. `Rectangle` and `Triangle` require two frequencies to be specified, one for the *x* direction and one for the *y* direction.

6.2. Pitch nomenclature

The frequency of an instrument can be specified in Hertz, conventional notation (including microtonal adjustments) or using a numerical notation of the form `pitch(<octave>.<semitone>)`. The conventional notation consists of a pitch class (e.g. `C`, `C#`, `Bb`, `G`) followed by an octave number (middle C is in octave 8) and an optional microtonal adjustment consisting of a `+` or `−` and a fraction representing the fraction of a semitone that the pitch should be sharpened or flattened by. For example `C#7`, `Eb6+1/2`, `pitch(8.02)` and `500 Hz` are all valid ways to specify the fundamental frequency of an instrument.

6.3. Microphone keywords

The keyword `Microphone` declares a virtual microphone. It is followed by the name of the microphone, the name of the file to which sound output is to be written, one of the keywords `mono` or `stereo` and optionally by the appropriate number of position specifications to determine where the microphone's sound sources will come from.

6.4. Nomenclature for accessing points on an instrument

As illustrated in figures 4 and 5 a point on an instrument is accessed using the notation `instrument(x,y)`. `x` and `y` are usually numerical constants or performance parameters, but the keywords `left`, `right`, `top`, `bottom` and `centre` are also provided as nomenclature.

6.5. Physical interaction with instruments

The physical parameters of a cell can be accessed using notation such as `instrument(x,y).force`, `instrument(x,y).velocity`, etc. Other physical parameters which can be accessed using similar notation are `position`, `mass` and `damping`. A force can be applied to a point using the notation `instrument(x,y).applyforce(force)`. Bowing is possible with `instrument(x,y).bow(bowforce,bowvelocity)` using a model loosely based on the one described in Woodhouse (1992). The bow's velocity, downward force and position can all change dynamically under algorithmic control.

6.6. Instrument modifiers

The keyword `lock` allows a single point on an instrument to be locked. It appears in the form `instrument.lock(x,y)` or just `lock(x,y)` in the body of an instrument declaration. The keywords `lockleft`, `lockright`, `locktop` and `lockbottom` allow whole sides of an instrument to be locked, and two other keywords `lockcorners` and `lockperimeter` are self-explanatory.

The `setdamping` and `setdecay` keywords allow the damping factor to be set at a single cell or over a region of cells. Damping can be set in terms of a decay time, or a percentage, where zero per cent means no damping at all and one hundred per cent means that the cell/cells are fixed rigidly in one position. The keywords `resetdamping` and `resetdecay` allow the damping to be reset to the default value defined when the instrument was created.

6.7. Screen output

The `Display` keyword allows for screen output of text, calculation results and parameter values. The

https://doi.org/10.1017/S1355771896000167 Published online by Cambridge University Press

keywords `newline` and `sameline` determine whether the next `Display` command will output at the beginning of the same line or the next.

6.8. Control structures dealing with time

The score is a hierarchical structure comprising nested control structures with the `Score` control structure at the top of the hierarchy. Control structures include `At`, `Before`, `After`, `From..to`, `At..for`, `Every` and `ControlRate`. A nested control structure can only be active when the head of the enclosing structure specifies. The following short score script illustrates the use of the control structures with the aid of comments in angled brackets:

```
Score 10 secs:
    Before 0.5 secs: < active from 0 seconds to 0.5 seconds > ...
    After 5 secs:    < active from 5 seconds to 10 seconds > ...
    Every 600 msecs: < active once every 600 milliseconds > ...
    At 1 secs for 4 secs:
        Before 3 secs: < active from 1 second to 3 seconds > ...
        ...
    ...
```

6.9. Mathematical functions

In addition to standard mathematical functions such as `sin`, `cos`, `sqrt`, `random`, etc., two special time varying functions `linear` and `exp` are provided. When placed in the body of a control structure, these functions automatically return a value which changes linearly or exponentially over the time interval specified in the head of that control structure. If they are used within the body of a `ControlRate`, `Every` or `At` control structure, they change over the interval specified in the enclosing structure's head.

6.10. Conditional control structures

Three more control structures are provided for making decisions within a score, `If`, `If..Else` and `If..ElseIf..Else`. These control structures have a head which contains a condition to be evaluated and the body is only activated if the condition evaluates to true.

6.11. Performance parameters

The keyword `Parameter` allows the user to declare any number of performance parameters with optional initial values and the keyword `Flag` allows parameters capable of holding the values `TRUE` and `FALSE` to be declared.

6.12. Time nomenclature

The keywords `secs`, `msecs` and `mins` are provided. When placed after a numerical value they automatically convert the value specified according to the units chosen. Also the keyword `Time` can be used within the score to refer to real time.

7. ADVANTAGES AND DISADVANTAGES OF TAO

7.1. Advantages

Since acoustic instruments are real objects and not black box algorithms, it is possible for them to be explored in different ways by different musicians. TAO instruments offer this same flexibility and place no restrictions on where, when and how an instrument is excited, or which points are used as sound sources. This allows the user to experiment creatively at an intuitive, gestural level, leaving the details of the synthesis to the naturally complex behaviour of the synthesis model.

Visual output in the form of instrument animations are provided by TAO and give valuable feedback about the behaviour of an instrument. Visualisation conveys a holistic picture of the behaviour of a system and in TAO such visualisations are an essential part of the synthesis technique. The script-based interface enables a performance to respond dynamically to the behaviour of instruments used within the script, making it possible to create sound events which have a life of their own not only at the timbral level but at higher structural levels also. It is possible to start producing sounds with the system with very little technical knowledge about physical modelling.

7.2. Disadvantages

The major disadvantage with the system is the amount of computational power required. Synthesis may be thousands of times slower than real time for complex instruments when running on a Silicon Graphics *Indy* workstation. Whilst this might seem like a step backwards in terms of performance, the musical flexibility afforded by this approach to sound synthesis makes the wait for faster or better suited technology worthwhile.

The script language, as it stands, does not provide facilities for creating higher level musical structures. It is difficult to know what kind of interface would highlight the strengths of the synthesis model, and

https://doi.org/10.1017/S1355771896000167 Published online by Cambridge University Press

suit the majority of composers in terms of musician-friendliness, without making musical assumptions which, whilst making the system accessible to some composers, would prevent others from achieving their goals. A text-based interface does not suit all musical situations.

8. WAYS FORWARD

There are a number of areas in which the system could be improved:

- Improving the basic timbral qualities of the material. The nature and extent of spring-like links with a cell and its neighbours could be changed leading to different materials suited to producing different classes of sounds.
- Encapsulation of instruments. It ought to be possible to describe an instrument as a collection of modules, where each module can be broken down into smaller modules, eventually reaching, at the bottom of the hierarchy, the primitive building blocks provided by the system. Such an encapsulation facility would make it possible to develop libraries of reusable modules.
- Encapsulation of excitations. The same kind of encapsulation could also be applied to excitations. For example, if we develop a score which explores a technique consisting of bouncing a virtual bow on an instrument whilst bowing, and this technique proves promising, it ought to be possible to give this new excitation a name and then use it repeatedly throughout the score in short-hand form.
- Other improvements to the script language. The whole script language needs to evolve through practical use in a compositional context. The score language, as it stands, only provides the ability to schedule events in terms of real time, but it is well understood that musical time is perceptually more complicated than the simple linear progression of real time. Tools for the musical description of time should be developed.
- Alternative interfaces. There is currently no facility for graphically designing instruments on screen, placing microphones, or providing performance data. Some further investigation into possible interfaces would ideally lead to a system in which graphical, physical and text-based interfaces all have their part to play.
- Speeding the model up. Since the synthesis engine consists of very many identical elements interacting on a local basis, implementing the model on parallel processors would seem to be a logical step forward to maximise the speed of execution. At the moment, for each cell in an instrument a minimum of 12 floating point calculations are required per time step.
- Creating irregularly shaped pieces of material. CSS is capable of supporting irregularly shaped pieces of material. These would greatly add to the ranges of possible timbre, but as it stands the script language only provides the ability to create certain predefined geometric shapes. This situation could be remedied with the addition of a graphical interface for instrument creation.

9. RELATED WORK

9.1. *Cordis-anima*

Cordis-anima (Cadoz, Luciani and Florens 1993) is a system for sound synthesis by physical modelling which has strong similarities to TAO in that the underlying synthesis model is based on point masses (cells) and springs (link elements). However, there are differences in the level of abstraction presented to the user. In TAO the microscopic structure of the material is an implementation detail which is hidden from the user, who sees an instrument as a continuous entity made of (initially) uniform elastic material.

In *Cordis-anima* more emphasis seems to be placed on individual cells and links as the principal building blocks of the synthesis model, although recent work has investigated the automatic generation of complex topologies (Incerti and Cadoz 1995). Another subtle distinction between TAO and *Cordis-anima* is that the former places emphasis on the waves propagating through the material and not on the material itself. These wave patterns give a holistic picture of the emergent properties of an instrument as a whole, and focus the user's attention away from the discrete nature of the model.

A graphical interface to *Cordis-anima* called *Genesis* (Cadoz, Florens and Luciani 1995) is currently under development and allows cells and links to be arranged on screen, with the usual facilities of cutting, pasting and copying.

9.2. Modal synthesis and *Mosaic*

Modal synthesis (Adrien 1991) is a physical modelling technique based on the premise that an acoustic object can be analysed prior to synthesis to determine its natural modes of vibration. This modal data is calculated from the geometric shape and physical properties of the object and, once obtained, can be used to model the object as a set of coupled masses and springs which act as elementary oscillators, each one tuned to a particular mode. *Mosaic* (Morrison and Adrien 1991, 1993) is a language for building and playing modal synthesis instruments.

The interface provided by *Mosaic* is based on the *Lisp* dialect *Scheme* (Abelson, Sussman and Sussman 1985). A detailed description of the language is out-

https://doi.org/10.1017/S1355771896000167 Published online by Cambridge University Press

side the scope of this paper, but it provides powerful abstraction capabilities allowing the user to build instruments and describe excitations in a modular fashion from elementary components provided by the system.

9.3. Digital waveguide synthesis

Another major approach to physical modelling is that of digital waveguides (Smith 1992) where resonators are modelled using delay lines and junctions. The delay lines propagate waves at very little computational cost and the junctions are placed at positions where excitations or reflections occur. These junctions contain all the mathematical modelling needed to simulate the effects of excitations and reflections.

9.4. Other physical models

A number of more specific models have been developed to simulate a variety of acoustic instruments. These include strings (Adrien and Rodet 1985, Adrien, Causse and Rodet 1987, Robinson 1987, Woodhouse 1992), wind instruments (Cook 1992, Keefe 1992), reed instruments (Smith 1986), and bar percussion instruments (Serra 1986). Other papers containing general overviews of physical modelling are by McIntyre (1983) and Borin, De Poli and Sarti (1992).

10. CONCLUSIONS

TAO is capable of creating an infinite variety of instruments having vastly different timbral qualities, using just a handful of consistent principles. The instruments are tangible physical objects which can be excited in a variety of ways. Multiple simultaneous excitations are possible and successive excitations give rise to natural sounding transients. The synthesis model produces sound by way of wave properties which emerge naturally from the material without being explicitly programmed, leading to sounds which have natural physical and spatial characteristics.

In practice the system has been shown to be capable of producing a wide variety of percussive sounds including bells, gongs, cymbals, metallic and wooden bars, pitched sounds such as strings and acoustic tubes, continuously excited sounds such as bowed strings and inharmonic bowed sounds, and other classes of sounds such as granular textures. The interface currently consists of a script language, with graphical feedback in the form of instrument animations. It is intended to develop the system further in terms of the possible excitations, the power of the script language, the speed of operation and of the overall quality and diversity of the sounds.

REFERENCES

Abelson, H., Sussman, G. J., and Sussman, J. 1985. *Structure and Interpretation of Computer Programs*. Cambridge, MA: MIT Press.

Adrien, J. M. 1991. The missing link: Modal synthesis. In *Representations of Musical Signals*. Cambridge, MA: MIT Press.

Adrien, J. M., Causse, R., and Rodet, X. 1987. Sound synthesis by physical models, application to strings. In *Proceedings of International Computer Music Conference*, pp. 264–9.

Adrien, J. M., and Rodet, X. 1985. Physical models of instruments, a modular approach, application to strings. In *Proceedings of International Computer Music Conference*, pp. 85–9.

Borin, G., De Poli, G., and Sarti, A. 1992. Algorithms and structure for synthesis using physical models. *Computer Music Journal* **16**(4): 30–42.

Cadoz, C., Florens, J. L., and Luciani, A. 1995. Musical sounds, animated images with cordis-anima and its multimodal interfaces. Demonstration at *International Computer Music Conference*.

Cadoz, C., Luciani, A., and Florens, J. L. 1993. Cordis-anima: a modeling system for sound and image synthesis, the general formalism. *Computer Music Journal* **17**(1): 19–29.

Cook, P. R. 1992. A meta wind instrument physical model and a meta controller for real time performance. In *Proceedings of International Computer Music Conference*, pp. 273–6.

Gleick, J. 1991. *Chaos – Making a New Science*. London: Cardinal.

Incerti, E., and Cadoz, C. 1995. Topology, geometry, matter of vibrating structures simulated with *cordis-anima*. Sound synthesis methods. In *Proceedings of International Computer Music Conference*, pp. 96–103.

Keefe, D. H. 1992. Physical modeling of wind instruments. *Computer Music Journal* **16**(4): 57–73.

McIntyre, M. E. 1983. On the oscillations of musical instruments. *Journal of the Acoustical Society of America* **75**(5): 1325–45.

Morrison, J. D., and Adrien, J. M. 1991. Control mechanisms in the Mosaic synthesis program. In *Proceedings of International Computer Music Conference*, pp. 19–22.

Morrison, J. D., and Adrien, J. M. 1993. Mosaic: a framework for modal synthesis. *Computer Music Journal* **17**(1): 45–56.

Robinson, C. Q. 1987. Real time synthesis of bowed string timbres. In *Proceedings of International Computer Music Conference*, pp. 125–9.

Serra, X. 1986. A computer model for bar percussion instruments. In *Proceedings of International Computer Music Conference*, pp. 257–62.

Smith, J. O. 1986. Efficient simulation of the reed bore and bow string mechanisms. In *Proceedings of International Computer Music Conference*, pp. 275–80.

Smith, J. O. 1992. Physical modeling using digital waveguides. *Computer Music Journal* **16**(4): 74–91.

Vercoe, B. 1992. *Csound Audio Processing System Manual*.

Waldrop, M. M. 1994. *Complexity: the Emerging Science at the Edge of Order and Chaos.* London: Penguin.

Woodhouse, J. 1992. Physical modeling of bowed strings. *Computer Music Journal* **16**(4): 43–56.

https://doi.org/10.1017/S1355771896000167 Published online by Cambridge University Press