Computer Sound Transformation

A personal perspective from the U.K.

Trevor Wishart

Introduction

For the past thirty years I have been involved in developing and using sound transformation procedures in the studio, initially working on analogue tape, and then through various types of computer platforms as computer music came of age. Over these years I've developed a very large number of procedures for manipulating sounds. Being a composer, I refer to these processes as musical instruments and they are developed as part of my musical work. However, I have not been prominent in publishing this work in academic journals as I'm primarily a working artist. Nevertheless, the processes (and source code) have all been available to others with the facilities to use (or develop) them through a composers' cooperative organisation based in the UK, the *Composers' Desktop Project*. As there has been a recent surge of interest in the Phase Vocoder [1] as a musical resource, I've been advised by friends in the academic community to put my contribution to these developments on record.

Origins

The earliest successful transformations I developed can be heard in the piece *Red Bird* (1973-77) [2]. The musical structure of the piece was conceived in terms of such transformations between sound types, but techniques for achieving this had to be developed on an ad hoc basis - through discovering what was practicable with the facilities available in the local analogue studio. The transformations, all from the voice to other sounds, include 'lis' (from the word 'listen') to birdsong, 'rea' (from the word 'reason') to animal-sounds, 'reasonabl-' to water, and various machine-like events constructed from vocal subunits. They were achieved by combining the elementary studio facilities available (tape editing, mixing, mixer eq) with extended vocal techniques (developed while working as a free improvising vocal performer [3]). A discussion of the approaches used in *Red Bird*, and the concept of Sound Landscape, can be found in *On Sonic Art* [4]. A more detailed description of the composition of this piece can be found in *Red Bird, A Document* [5].

Realising that these notions of spectral transformation could in principle be generalised in a computing environment, when major computer music facilities became available in Europe (at IRCAM in Paris) I submitted a proposal for a work based on vocal transformation and was invited on the induction course in 1981. There I discovered a potential transformation tool (Linear Predictive Coding [6]), and was invited to compose a work. Unfortunately the mainframe system at IRCAM, and much of the indigenous software, was changed immediately following this visit, and the project could not proceed until 1986, when the research and composition for *Vox 5* was finally commissioned. It was suggested to me that the CARL Phase Vocoder (Moore, Dolson) might be a better tool to use, but no-one at IRCAM at that time had inside knowledge of the workings of this program, so I took apart the data files it produced to work out for myself what was going on.

I eventually developed a number of software instruments for the spectral transformation of sounds which were then used to compose *Vox 5*. These instruments massaged the data in the analysis files produced by the Phase Vocoder. The most significant of these were **stretching the spectrum** (see below) and **spectral morphing** – creating a seamless transition between two different sounds *which are themselves in spectral motion*. These are described in a Computer Music Journal article [7].

Establishing a personal computer based development environment in the U.K.

Returning from IRCAM to the UK musicians were faced with an entirely different development environment. There was no independent national research centre for music – music research was confined to University music departments. Most of these were small and very poorly funded – they were seen as primarily sites of humanities research and hence could not attract the money required for advanced computing equipment, which at that time was very expensive. A number of departments had PDP-11 computers accessible to a few research students and staff – updating this equipment was a constant financial worry.

During 1986-7, a group of composers (initially Andrew Bentley, Archer Endrich, Richard Orton, and myself) and developers (Martin Atkins and David Malham) based in York, (and all ex-graduates or current staff of the University of York), working in financially astringent circumstances [8], ported Richard Moore's *Cmusic* [9] and the Mark Dolson Phase Vocoder to a desktop platform. I then implemented the instruments developed at IRCAM. The platform chosen at the time was the Atari ST as this machine was *just* fast enough to be able to play stereo soundfiles running at a sample rate of 48,000 – at that time the Macintosh was not fast enough [10]. This was the start of a larger project to make computer music tools available to composers and institutions without significant financial resources. This project was called the *Composers' Desktop Project* (CDP). In fact the idea of a user-group development-environment based on personal computers originated out of necessity in this environment, predating subsequent developments at IRCAM and elsewhere by several years.

The instruments ran initially in a command-line environment (a graphic environment was developed later by Rajmil Fischman and others working at the University of Keele). They were later ported from the Atari ST first to the Atari TT then to the PC where they ran (and, in 2000, still can be run) under MS-DOS. Almost all later platforms were chosen partly for their low cost – many CDP users, including myself, did not have departmental salaries or budgets to buy expensive personal computers or to constantly update them. The software also ran on the Silicon Graphics machines at the University of York and elsewhere. More recently two different graphic interfaces have been developed to drive the software, my own contribution being the *Sound Loom*, written in TK/TCL so that it is potentially portable from one computer platform to another.

Immediately after the IRCAM project in 1986, working in the CDP environment, I developed a large number of other spectral transformation tools using the Phase Vocoder data as a starting point. Subsequently, I also created a number of original time-domain instruments (e.g. **waveset manipulation**, **grain manipulation**, **sound shredding**) and extensions of existing instruments (e.g. **brassage** [11]). In 1994, a complete description of all the spectral, time-domain and textural transformation possibilities available on the CDP system was published in the book *Audible Design* [12]. The book has been used subsequently as a source by other software developers (for example by Mike Norris who implemented many of my waveset manipulation procedures on the Macintosh, now available from *Sound Magic*) some of whom may well have had access to the CDP code.

I would stress that the work of researchers and developers at IRCAM (notably Steve McAdams who introduced me to contemporary psycho-acoustic research on the 1981 induction course and, later, Miller Puckette), and at the G.R.M. [13] – where I attended the composition course in 1993 – were an important source of knowledge, ideas and inspiration for my work. However, when the sound morphing and spectral stretching instruments for *Vox 5* were originally developed as part of the public domain system shared by IRCAM, Stanford and other major sites, IRCAM's research priorities were focused elsewhere. The instruments did make their way to the USA via the University of Santa Barbara and Dan Timis (the resident computer wizard at IRCAM when I was working there). Later IRCAM did decide to pursue Phase Vocoder based transformation and the *Super Phase Vocoder* (SVP) group was established (the basis of the later *AudioSculpt*). During the development phase of SVP, when the CDP spectral transformation suite was already quite large, I was a visitor at IRCAM, and discussed possible transformational approaches with some of the team working on the program.

The pioneering development work of the CDP has remained largely unknown or forgotten about as the vast majority of the Computer Music community eventually opted for the Macintosh as the machine of choice. Furthermore, being developed primarily by a group without official financial support from within the University infrastructure, the project was always short of resources. An initial grant from the Gulbenkian Foundation helped propel us forward in the first 18 months, but this was exceptional. Nevertheless the CDP continued to make both its instruments and its code available to interested users and developers. I am indebted to the work of many others developers (including in particular Richard Orton and Richard Dobson) and to the C.D.P. Administrator, Archer Endrich, for continuing to promote, support and develop the system, and make it more accessible to users, despite the lack of financial rewards. The system has tended to be adopted by independent composers or small educational institutions with limited budgets. However, the source code has been available at a number of UK (and other) University sites at different times, even after these moved to a primarily Mac-based studio system. And some institutions, notably the Institute of Electronic Music in Vienna, developed sophisticated graphic interfaces of their own.

The Instruments – (1) Spectral Transformation using the Phase Vocoder

There is not enough space to describe all the C.D.P. procedures in this article, so I will describe only the more interesting ones, or those not available elsewhere. Full descriptions of all these processes can be found in *Audible Design*.

In the first phase of development (post 1986), many spectral transformations were implemented in the Atari environment. These included **spectral morphing** (see above), and various types of **spectral shifting** and **spectral stretching**, from a linear shift (adding a fixed value to all frequency data, thus e.g. making a harmonic spectrum become inharmonic), through a multiplication (preserving harmonic relations between data, but transposing the pitch – but with the ability to split the spectrum at a given frequency, and hence produce doubly-pitched output sounds) to differential multiplication of the data (**spectral stretching**, a more sophisticated way to convert harmonic into inharmonic spectra, used in *Vox 5* to convert vocal sounds into bells). **Time-variable time-stretching** procedures were also implemented, more general than those existing in the CARL Phase Vocoder implementation itself. These are important if one wishes to preserve the attack characteristics of a sound while time-stretching the sound (as a whole) by a large factor.

Spectral cleaning was developed using a comparative method – part of the spectrum deemed to be (mainly) noise (and, in some options, part of the spectrum deemed to be clear signal) being compared with the rest of the signal and appropriate subtractions of data or other modifications made.

From a musical point of view, the most innovative early new developments were **spectral banding**, a rather complicated 'filter', which enabled the spectrum to be divided into bands, and various simple amplitude-varying (and in fact frequency-shifting) processes to be applied to the bands, **spectral tracing** and **spectral blurring**.

Spectral tracing simply retains the N channels with the loudest (highest amplitude) data on a *window-by-window basis*. If N is set to c. 1/8th the number of channels used in the PVOC analysis, this can sometimes function as an effective noise reduction procedure (the value of N which works best depends on the signal). When N is much smaller than this, and a complex signal is processed, a different result transpires. The small number of PVOC channels selected by the process will vary from window to window. Individual partials will drop out, or suddenly appear, in this elect set. As a result, the output sound will present complex weaving melodies produced by the preserved partials as they enter (or leave) the elect set. This procedure is used in *Tongues of Fire* [14].

Spectral blurring is an analogous process in the time dimension. The change in frequency information over time is averaged – in fact, the frequency and amplitude data in the channels is sampled at each Nth window, and the frequency and amplitude data for intervening channels *generated* by simple interpolation. This leads to a blurring or 'washing out' of the spectral clarity of the source.

Arpeggiation of the spectrum (a procedure inspired by vocal synthesis examples used by Steve McAdams at IRCAM to demonstrate aural streaming) was produced by 'drawing' a low frequency simple waveform onto the spectrum. This oscillator rises and falls between two limit values – values of frequency in the original spectrum – specified by the user. Where this waveform crosses the spectral windows, the channel (or surrounding group of channels, or all the channels above, or all those below) is amplified. **Spectral plucking** was introduced to add further amplitude emphasis (and an element of time-decay of the emphasized data) to the selected channels.

A number of other processes (such as **spectral freezing** and sustaining of the spectrum at particular moments, and **spectral interleaving**, timewise, the spectra from different sources) were implemented in this early phase.

Tuning the spectrum was introduced a little later. **Tune spectrum** works by selecting channel data lying close to the partials of a specified set of pitches, and moving the frequency of that data to (or towards) the desired partial frequency. The spectrum can also be traced (see above) before doing this. **Choose partials** selects channels which should contain frequencies close to those of a specified set of partial frequencies (harmonics of, odd harmonics of, octaves above, linear frequency steps away from, or a linear frequency displacement from harmonics of a given fundamental). As analysis channels above the 21st are sufficiently narrow to focus on a semitone band of frequency or less, the channel number itself is sufficient to grab the desired partials.

After discussing possible algorithms with the SVP developers, I implemented some of their ideas for spectral filters (defining filters in a more conventional way than the banding procedure described above), and implemented various types of **low pass**, **high pass**, **band pass**, **notch** and **graphic e.q. spectral filters**, together with a **chorusing** procedure suggested by Steve McAdams' work (introducing jitter into the partials data).

After discussions with Miller Puckette about his work on tracking the pitch produced by instrumentalists performing in real time, procedures to **extract the pitch** of PVOC data were finally developed into a useful form, and instruments to correct the data, to transform the pitch data (**quantise, shift, vibrato, approximate, or randomise the pitch**, and **exaggerate, invert or smooth the pitch contour**), and **apply the pitch to other sounds**, were developed.

At the same time, the **extraction of formants** from the PVOC data was implemented satisfactorily for the first time within the CDP environment. This enabled the **inner glissando** procedure to be developed. Here, the process retains the time-varying spectral envelope (the formant envelope) of the sound, but replaces the signal itself by an endlessly glissandoing Shepard Tone signal [15].

Shuffling the sequence of windows, and **weaving** a specified path (including possible repetitions and omissions) through the windows were implemented at an earlier stage. A **'drunken-walk'** through the analysis windows was suggested by Miller Puckette's work in MAX. Miller also suggested the procedure of **octave pitch-shifting** through selective partial deletion, while Oyvind Hammer of NOTAM [16] proposed **scattering** of the spectral data.

The Instruments – (2) Original Time-domain procedures

Alongside this spectral transformation work, a large number of time-domain procedures have been developed for sonic composition.

Waveset distortion was developed for the CDP while composing *Tongues of Fire*. I defined a waveset as the signal between any pair of zero-crossings. With a simple sine-wave the waveset corresponds to the waveform. But even with a harmonic tone with very strong partials, the waveform may cross the zero more than twice in a complete cycle. In this case the wavesets are shorter than the waveform. With complex signals (e.g. speech) containing noise elements, the definition of the waveset produces many varieties of technically arbitrary, but potentially musically interesting, artefacts. A whole suite of procedures was developed to manipulate wavesets. I have used three at prominent moments in compositions.

The first of these involves replacing each waveset with a standard-shape waveform (e.g. a sinewave). This produces a very pronounced spectral transformation of the source, but one where the zero-crossings of the result are exactly aligned with those of the source. It is thus possible to use a simple mixing procedure (another CDP process, **Inbetweening**, does this) to produce a sequence of sounds intermediate between the source and the new sound. These two procedures were developed and used to produce the 'Wood' to 'Drum' transformations in *Tongues of Fire*.

The second, **waveset averaging**, involves extracting the shape of each waveset, and then averaging this shape over a group of N adjacent wavesets. Again, this produces an extreme modification of the source (usually a relatively harsh sound and often a transformation so distant from the source that little audible connection is apparent!) and is used in the 'fireworks' transformation immediately after the rhythmic climax of *Tongues of Fire*. The article *Sonic Composition in 'Tongues of Fire'* [17] discusses this in more detail.

Finally, **waveset repetition** generates unusual pitch artefacts in complex signals. In particular, any small fragment of a noise signal, if repeated a number of times, generates a tiny pitch artefact. The second movement of *Two Women* (18), based around the voice of Princess Diana, uses this instrument to ornament and fragment the vocal material, different repetition rates being used in the left and right channels to produce an irregular panned echo/delay, with **iteration** (see below) being used as a reverb-like process to sustain various pitch elements which arise from the first procedure.

Familiarity with the G.R.M.'s work on the classification of sounds [19] drew my attention to the difficulty of time-stretching iterative sounds i.e. sounds like a rolled 'r' or a low contrabassoon pitch, where the sound is perceived as a series of individual attacks. In (realistic) time-stretching, we need to avoid time-stretching the event attack itself as stretching this can dramatically alter our recognition (or mental classification) of the source. Hence we would usually apply a time-stretching parameter which itself varies through time, being 1.0 (no stretch) during the attack, and increasing rapidly to the desired stretch ratio immediately after the attack. With iterative sounds, however, we are faced with a whole stream of attacks, and this simple solution is not available. To deal with these a number of **Grain manipulation** instruments were developed. These instruments extract the (loudness) envelope of the sound by gating it. Using this envelope the source can be fragmented into attacked elements and these elements repositioned in time (or in pitch or both) in the output sound. (The process can also track the *overall* amplitude of the source and adjusts the gate level for the grains correspondingly).

This approach also allows one to **reverse an iterative sound**. Most sounds have an asymmetric form with a (relatively) loud initiating event at the beginning, and a tailing away to zero at the end (these features themselves can have a vast number of forms). Playing a sound backwards therefore rarely results in a sound that we recognise as being a close relative of the original. Only sounds of (on average) steady amplitude which have attack and decay as inverses of one another e.g. a slow fade in matching a slow fade out, will appear similar when we reverse them. Iterative sounds are particular difficult in this respect as every attack within them gets reversed. If we extract the grains and then sequence them in the reverse order, without reversing the grains themselves, we achieve a convincing sense of retrograding the sound without change of source recognition.

Similar sound-structural considerations apply to extending sounds using looping procedures. Recording a rolled 'r', isolating a single tongue-flap sound, then looping it to generate a 'rolled-r' at the same rate as the original produces an entirely mechanical artefact sounding completely unlike the original rolled-'r' source. Natural 'repetition' is usually micro-inexact. Thus the **Iteration** instrument allows a signal to be looped, but imposes (user-controlled) random pitch, amplitude and timing fluctuations on the repeated elements. Using Iteration the sound generated from the single flap can be extremely natural (but, of course, more distant transformations are also possible). Grain manipulation and Iteration were both developed and used while composing *Tongues of Fire*.

Various instruments allow scrambling of a sound through simple editing and rejoining of the edited segments. In particular, in **Sound Shredding**, I cut up a sound (at random time-points) into a number of separate segments, shuffles these segments, and reassembles them to the exact duration of the original. The *resulting* sound is then cut up again, differently, and the reassembly repeated. This process can take place any (user-specified) number of times. Applying the process about 400 times to rapid speech material produced a sound very similar to that of water running around rocks in a small stream, and this transformation can be heard spanning a 2 minute section of *Tongues of Fire*.

The Instruments – (3) New perspectives on existing procedures

What are now referred to as **granular synthesis** procedures, but applied to input sources, were developed at any early stage of the CDP. (The CDP instruments are almost exclusively concerned with the transformation of existing sources, rather than with synthesis). These are described as **texture generation** instruments. Initially these procedures generated scripts for a simple *Csound* [20] instrument which read (any number of) input sounds and then distributed them in the texture according to the instructions given by the user. The dependence on *Csound* scripts was superseded by direct use of the soundfiles themselves.

Texture Generation was (and is) able to use an arbitrarily large number of input sounds, to generate a stream of events where all the following parameters can themselves vary through time:

- the average time between event repetitions (the density of events) *or* the specification of a sequence of event times
- the scatter (or randomisation) of event timings (which means the instrument can generate anything from dance-music-like regularity to complete arhythmicity)
- a quantisation grid for times (or none)
- a specification of which range of input sounds are to be used
- the range and range-limits of pitch-transposition of the events
- the range and range-limits of event amplitudes
- the range and range-limits of durations of the individual events in the texture
- the spatial centre of the texture on the stereo stage, and its motion
- the spatial bandwidth of the texture on the stereo stage.

A neutral texture is generated from independent events over a transposition range without regard to tuning, tempering etc. However, the texture can also be generated...

- over a harmonic field (not necessarily tempered) which can itself change through time
- clustered into groups of events of specified or random pitch-shape
- formed from a line with arbitrary or specified decorating patterns (which themselves have properties with independent parameters of their own).

The texture generation instrument are used extensively in all my sonic art pieces since *Vox 5*.

The **brassage** techniques extensively and powerfully developed by the G.R.M. and implemented (in various guises) in *G.R.M. tools* I have independently implemented in the CDP environment. The G.R.M. have divided brassage into a series of sub-categories based on musical outcomes (based on many years of musical experience), providing the user with control of parameters over a musically meaningful range for each resulting tool. I admire this approach and accept that it is much more accessible to the user who is a computer *user* rather than a programmer. However, compositionally I often find it interesting to explore the areas where a process pushes against its limits and falls over into another area of perception. E.g., if the size of grains used in a time-stretching brassage routine (as used in the *harmonizer*) exceeds a certain threshold we begin to hear the resulting sound as a rapid collage of elements rather than as a simple timestretch. Hence the CDP brassage routine offers **timestretch, pitch-shift, granulation** and **source-scrambling** as independent modes, but also allows access to *all* the parameters of the brassage process at the same time:

- timestretch or compression and its range
- segment density and its range
- segment size
- segment transposition and its range
- segment amplitude and its range
- segment splice-length

- segment spatial position
- segment spatial scatter and its range
- segment timing randomisation
- segment search-range in the source

where every parameter can also be varied in time.
The process can also be applied to more than one input sound.

Modifying the loudness contour (envelope) of a sound is a fairly standard procedure. **Envelope extraction and superimposition** (written by Richard Orton) and **envelope manipulation** (which I developed from Richard's programs) were some of the earliest processes to be developed in the CDP. These allow the envelope to be extracted at different resolutions (e.g. a tremolo sound which crescendos has a small-scale, rapidly-varying loudness envelope defining the tremolo, and a large scale overall envelope defining the crescendo. These can be extracted separately using a different window-size for the envelope extraction). The envelope can then be changed (**envelope warping** – normalise, limit, compress, exaggerate, corrugate etc), and applied to the original, or a different sound. Even applying the envelope comes in two varieties. Simple envelope superimposition is found in most mixing packages (often implemented by drawing the envelope), where an envelope contour is imposed on the existing sound. However, we can also **envelope replace**, where the new envelope replaces (rather being superimposed over) the envelope of the processed sound. In this case we force the original sound to have a flat level throughout (treating in a special way points in the sound where the envelope approaches zero), then apply the new envelope to the flattened sound.

Enveloping can obviously be used to produce tremolo. More radically, in a reversal of the Karplus-Strong synthesis procedure [21], we can **produce a plucked attack** on an existing sound. (The procedure involves finding the first steady-pitch wavecycle in the source – assuming there is one – then preceding it by copies which become increasingly loud and noisy). This process was developed and used in *Tongues of Fire* but is not 'automatic' in its operation and is quite tricky to sculpt.

Mixing is now usually carried out in a graphic environment displaying pictorial representations of waveforms (and envelope and panning contours) in tracks on the screen. The CDP Submix instrument (which I developed from existing CDP mixing facilities developed by Andrew Bentley and others) is based on a much earlier paradigm, mixing from a (text) list of soundfiles. Despite being much less friendly than screen based mixing, this does allow for some powerful global procedures to be applied to mixes. The CDP submix should be thought of as a way to generate a new event from several source sounds, rather than as a conventional track-mixing environment (although I do *all* my mixing in this environment).

First of all, there is no limit to the number of 'tracks' used (apart from the memory space of the computer). Any number of sounds can be superimposed. Secondly, global operations on the mix are available, from simple features like doubling (or multiplying by any number) the distance between event onsets, or randomising them (very slightly or radically), to randomly swapping around the sound sources in the mix, automatically generating particular timing-sequences for event entry (from regular pulses, to logarithmic sequences etc.), or redistributing the mix output in the stereo space in a new, user-defined way.

More specialised procedures involve **synchronising the mix events** (e.g. at their mid-point, or end, as well as at their start), or **synchronising the event-attacks** (where the search-window for the attack peak can be delimited by the user). These latter procedures are particularly useful for building complex sonorities out of less rich materials e.g. by superimposing transposed copies of the sound (over the original duration, or in a different duration) onto the original. Similarly, **Inbetweening** allows the generation of sets of closely related sonorities (see above), while **Cross-fading using a balance function** allows a sound to gravitate between its original form and a transformed variant in a time-varying way.

There are no original filter algorithms in the CDP, but some powerful filter design frameworks are available. In particular **filter varibank** allows one to define a filter over a set of pitches which itself varies in time, where each pitch element has an associated amplitude (which can go to zero so that pitches, or moving-pitch-lines, can be 'faded out' or cut). The number of harmonics of those pitches (and their relative level) can be specified (these serve to define further individual filter frequencies), and the filter Q can also vary through time. This filter-building algorithm was developed and used during the composition of *Fabulous Paris* [22].

Finally, at a time when synthetic bell-sounds seemed to dominate computer music works, I decided a bit of grittiness would be welcome, and developed instruments which **lower the resolution** of the sound (reducing the effective bit-representation, or the sampling rate), **ring modulate** and **inter-modulate** sources, and even attempt (rather unsuccessfully) to simulate manually **scrubbing a tape over the heads** of an analogue tape-recorder.

Additional Aids to Composing

Over the years I have also developed a large number of utilities which I find indispensable as a composer, starting with an instrument which **searches a tape of source recordings and extracts significant segments** from surrounding silences or clicks, using gating, and selection parameters specified by the user. Next there are facilities to **compare sounds**, or **compare the channels of a single sound**, (are they the same, or almost the same to within specified limits?), to **balance the level of sources**, or the channels of a source, to **invert (or narrow) spatial orientation**, and to **invert phase** (which, apart from anything else, can be used to gain more headroom in a mix).

In the various instruments described in this article, alomst all parameters can vary through time. Data for this is provided in simple textfiles containing time+value pairs. To aid in working with such data, hundreds of automatic data-creation and data-modification processes have been implemented, and are made available in the **Table Editor**, now also driven from the graphic interface. I have used it to design and modify complex filter specifications, to generate 'random funk' accentuation patterns as envelopes over an existing stream of events (*Birthrite A Fleeting Opera* [23]) – and even to do my tax returns(!). As an additional aid, a **Music Calculator** allows easy conversion between a great variety of musical and technical units.

The future...

Currently (October, 2000) all this software works in non-real-time in a PC environment, mainly with 16-bit soundfiles. The next version (early 2001) will handle all currently available soundfile formats. In addition the sound-buffering is being modified so that those instruments which could, in principle, run in real-time can be enabled to do so. There is also no reason (apart from lack of time or resources) why this entire environment should not run on the Macintosh, or any other platform, as it is written in 'C' and TK/TCL. These two latter tasks could be accomplished without great difficulty by someone with the time and enthusiasm to commit.

Footnotes

1. A process which divides the source sound, timewise, into tiny (overlapping) 'windows', performs a fourier analysis on each window to determine the spectrum of the sound in the window, then deduces the frequency of the components in the window by considering the change of phase from one window to the next. (Return to main text.)
2. This can be found on the CD ***Red Bird: Anticredos*** (EMF CD022) (Return to main text.)
3. The vocal techniques were documented in a catalogue of extended vocal techniques, *The Book of Lost Voices* (1979), later incorporated as a chapter in the book ***On Sonic Art*** (see footnote 4). (Return to main text.)
4. Originally published privately in 1984: republished (edited by Simon Emmerson) by Harwood Academic Publishers, 1996. (ISBN:371865847X) (Return to main text.)
5. Originally published privately by Trevor Wishart, 1978. See publications. (Return to main text.)
6. A procedure developed for the analysis and resynthesis of speech. It first differentiates noise and pitch based elements in the source. It then generates a sequence of filter specifications for consecutive moments in the source which, when applied to a buzz (rich in harmonics) tone or a noise source, reproduces the original sound. (Return to main text.)
7. *The Composition of Vox 5 at IRCAM* : Computer Music Journal Vol. 12: no 4: Winter 1988. (Return to main text.)
8. Despite the involvement of staff members, no financial support was forthcoming from the University authorities at that time. I donated 100 to help pay for materials to build the first 'SoundSTreamer', the buffering device, designed and built by Martin Atkins and David Malham, which enabled us to get sound in and out of the ROM port of the Atari ST. (Return to main text.)
9. General purpose software sound-synthesis environment. (Return to main text.)
10. To all those chuckling into their anoraks I would add that the Atari ST was 100% reliable. It simply never crashed in all the years it was used. (Return to main text.)
11. A process that cuts the sound, timewise, into segments (possibly overlapping, possibly separated in time), then reconstructs the sound by splicing these back together in different ways. (Return to main text.)
12. Published by *Orpheus the Pantomime*, UK. (ISBN : 0951031317) (see Publications). (Return to main text.)
13. Groupe de Recherche Musicale, Paris. (Return to main text.)
14. *Tongues of Fire* is available, by itself, on CD, or on the album ***Voiceprints***. (Return to main text.)
15. A tone which appears to rise (or fall) in (chromatic) pitch forever while remaining in the same tessitura. (Return to main text.)

16. The Norwegian Centre for Computer Music. (Return to main text.)
17. Computer Music Journal: Vol 24 No 2 Summer 2000 (Return to main text.)
18. On the CD *Voiceprints*.. See publications. (Return to main text.)
19. *Solfege de l'objet sonore* by Pierre Scaheffer and Guy Reibel. (Return to main text.)
20. General purpose software sound-synthesis environment, by Barry Vercoe. (Return to main text.)
21. A compact algorithm to synthesize plucked-string sounds of many types. (Return to main text.)
22. On the CD *Or Some Computer Music: 1* from *Touch*. (Return to main text.)
23. *Birthrite A Fleeting Opera* with Max Couper: River Thames, London, 2000. A score of ***Birthrite*** is also available, for dryland performance. (Return to main text.)