Music Research August 1989 − June 1990

Martin Guy

ABSTRACT

This memo briefly covers some aspects of music research carried out by me at the University of Kent at Canterbury (UKC).

1. Csound

Much of this work centres around **Csound**, a program written in C to implement generalised sound synthesis and sound processing algorithms.

Normally, Csound reads two files. One, the orchestra, contains a description of the structure of the instruments to be used in the performance; the other, the score, contains a list of musical events, predominantly note events with starting time, duration and pitch.

The resulting sample values are written to a file as they are computed, to be played when the computation run is complete.

1.1. Interactive Csound

Csound was designed to be used, and is currently used all over the world, in batch mode. The latest generation of high-speed workstations (Sun's SPARCStation and DEC's DECStation) is fast enough to generate sound fast enough to give live performance, albeit at only 8192 samples per second, so I modified Csound to let you send sound data to the DAC as it is generated. These modifications, among others, have been incorporated into the version distributed from MIT by Barry Vercoe, the author of Csound.

Experiment 1

I tried the following experiment to use Csound as a synthesizer driven by a piano-style keyboard.

To get MIDI events from the physical MIDI keyboard into Csound, we can use an Atari ST to convert the physical layer of MIDI from the MIDI specification (31250 bps) to RS232, and connect this to a serial port of a host (figure 1).

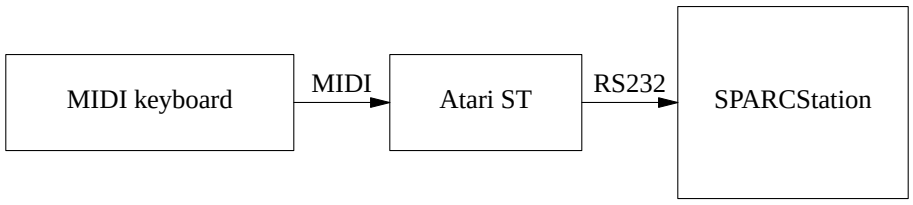

Figure 1

A process running on the host timestamps the MIDI events and writes them out in Csound score notation. This is piped straight into Csound with an appropriate orchestra description (figure 2).

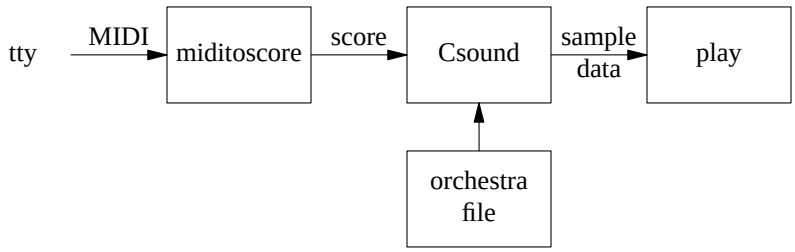

Figure 2

This setup worked, although it was rather like playing an organ in a large cathedral; the system would settle down to a constant delay between keypress and sound output of about two seconds. Most of this was not caused by the processing time involved, but by two main factors: Csound needs to know the timing of the next score event before it can start processing the current one (so that it knows when to stop), so it is necessary for **miditoscore** to insert dummy score events at regular intervals (in this case, simply triggered by the ‘‘active sense’’ MIDI codes, which happen about every 100ms.) This is responsible for part of the delay, the other source being buffering on Csound’s output (of the order of 8k, representing one second of sound). Both of these delays could be greatly reduced.

Experiment 2

Csound already accepts prepared files of sound data as audio signal sources. I modified it to allow the standard input as a possible source of sound data.

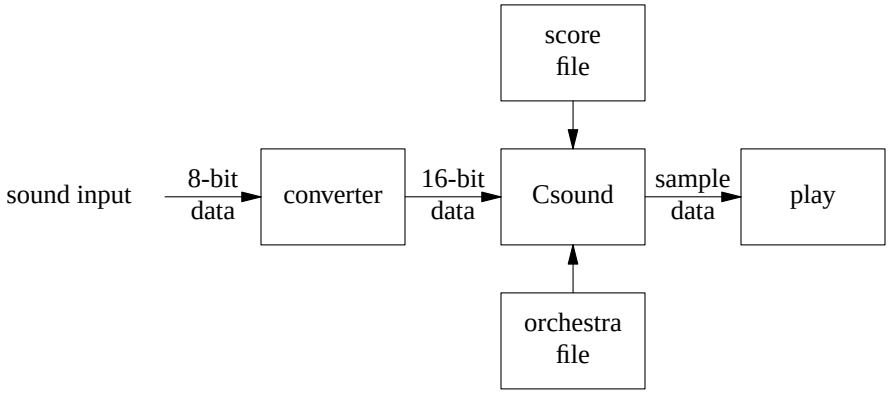

Figure 3

I got as far as displaying voice input from a microphone, both as as waveform and a spectrum analysis, and observed the effects of different vowel sounds. Once an audio signal has been introduced into Csound in this way, arbitrary processing can be performed using the signal. Lack of equipment prevented any further progress in this direction.†

1.2. X11 support for Csound

I spent some time adding code to use the X Window system to display function tables, time-amplitude and frequency-amplitude graphs (oscilloscope traces and frequency spectrum analyses) of signals within instruments. This produces smooth animated graphs for audio signals, and changing Fourier transforms at the rate of a cheap cartoon. These are shown on the accompanying screen dump.

While this code is usable, it is not in a form that can be released, except as a basis for further work. Currently it simply creates a new window for each displayed graph. I had hoped to develop this into a combined display panel and simple visual environment for developing and performing Csound orchestras and scores.

2. Sound Synthesis

Sound synthesis itself is an art as broad as painting, which I am only beginning to explore. Some example tones are:

2.1. Organ tone

This is produced by additive synthesis, the sum of the first eight harmonics of the intended pitch, in appropriate proportions. Assuming a strength of 100 for the fundamental, the relative strengths of the first seven overtones are 70, 60, 50, 40, 35, 0 (zero) and 30. These are the relative strengths recommended in the Art of Organ Building (Rev. W.H.Audsley, pub. 1904) for a compound stop composed of several ranks of pipes. In practice, the seven sine waves are summed over one cycle of the fundamental and stored in a table.

2.2. Wobba

This is an entirely synthetic instrument, which consists of four sine wave oscillators tuned to once, twice, three and four times the stated pitch, where the amplitude of each partial is controlled by a low frequency sine wave generator whose frequency is proportional to that of the partial it controls. This produces an effect reminiscent of the rotating ‘‘Leslie’’ speakers used on Hammond organs.

3. Dynamic spectrum display

I had an idea to display the harmonic structure of a piece of music over time, not as the traditional three-dimensional wire-frame graph, but as a bitmap where the axes represent time and frequency, and the blackness of each pixel represents the energy in the sound spectrum at that frequency.

Plate 1 shows an analysis of a test piece: a sine wave swept in frequency from 20Hz to 4000Hz plus three quiet tones at 20Hz, 200Hz and 2000Hz as markers. Note that, while the tone sweep at a constant number of octaves per second, the frequency scale on the graph is linear at present.

Plate 2 shows an analysis of ten seconds of a single note of the ‘wobba’ instrument. The different rates of fluctuation of the four harmonics can clearly be seen.

Plate 3 is an analysis of the opening chords of ‘‘Also Sprach Zarathustra’’, starting with a single organ tone. In the final chord, notice beats appearing where two harmonics of different notes of the chord coincide.

Finally we have two contrasting pieces of music.

Plate 4 is the accompaniment to ‘‘Rock Steady’’ by Sting, with a single string bass line, with short puffs of a three-note chord. It can be seen in each individual note in the bass line (particularly the first) that the plucked string’s higher harmonics die away faster than the fundamental.

Plate 5 shows the opening section of the fast movement of the Italian Concerto by J.S.Bach, played on ‘wobba’. It is clearer here than anywhere else that these graphs need a logarithmic frequency axis. Also, darkness is at present determined by the peak amplitude of any particular frequency, whereas the ear‘s frequency response is roughly logarithmic, being more sensitive to high notes.

4. Machine loading

Sound processing is traditionally very expensive in CPU time, in disc space, and in system throughput during performance. Stereo CD quality sound, for example (16 bits, 44,100 samples per second) requires 172 kilobytes of sample data per second of performance.

Because of the low sampling rates being used, CPU usage is moderate, and only 8 kilobytes of sample data are needed per second of telephone-quality-with-bass sound available on a SPARCStation.

Typical performances do not use anything like the full speed of these machines (in fact, because they are multi-user machines, a performance which does require anything like the whole machine’s speed, will not keep up in practice), and because for most testpieces sound is directly output, sound data is not stored.

5. The future

I have a vision of an algorithmic synthesizer which performs all of its synthesis and sound processing in software, in real time. There are obvious trade-offs between sound complexity, number of simultaneous voices and sampling rate. This would involve entering a representation of the synthesis algorithm graphically, and producing a runnable form from it.

Object-oriented programming is known to be well suited to user interface applications, in particular those where the user builds a structure which is composed of active objects, but are also highly relevant in sound synthesis where the relation between an instrument template and a running invocation of that instrument (performing a note) are particularly easily representable.