BRISBANE NOCTURNE - An algorithmic composition using SoftStep.

Warren Burt
Freelance composer.
PO Box 2154
St. Kilda West, Vic. 3182
waburt@melbourne.dialix.com.au

Abstract

This paper describes the structures used in *Brisbane Nocturne*, a real-time interactive composition made using John Dunn's "Softstep" algorithmic Windows software. I have been involved in the design of Softstep, first as a beta tester, then more recently as a consultant and module designer. The piece, made initially as a demonstration of some of the algorithmic functions of the software, quickly assumed a life of its own, and became music as well as a demonstration of various structures. The paper describes the functioning of the following Softstep modules: Ball, Rhythm Generator, Bio-Sequencer, Probability Generator, Fractal Generator, Table Write, Page Sequencer, Corruption, Delay, Chaos Generator, Image, and Number Generator, all of which are used to generate aspects of the piece. Aspects of some of these modules (Probability and parts of Chaos Generator) have been designed by me, although the bulk of the work on the program (99%+) is by John Dunn. The functioning of each of these modules will be demonstrated with examples from the piece. The result of combining all these structures is a composition with, hopefully, great subtlety, and this kind of subtlety is proposed as one of many possible ways forward for algorithmic composition.

Brisbane Nocturne Overview

In mid 1999, I became a beta tester for John Dunn, while he was developing SoftStep, an algorithmic sequencer based MIDI controller for the Windows environment. I had been using his earlier DOS based Kinetic Music Machine for many years, and was delighted that he was finally porting that over to Windows. My involvement as a beta tester quickly escalated to that of a consultant as I began plying him with suggestions, which he responded favourably to. Finally, I began designing and suggesting a few modules for the program, and eventually, many of my compositional dreams of the past 20 years became reality, thanks to John's programming skill, hard work, and musical intuition. I am now very excited by all the possibilities made available in SoftStep. It has a unique set of possibilities not found in any other program for any other platform.

Brisbane Nocturne is a piece made specially for this conference and this paper. It uses John Dunn's Softstep to control the synthesis engine in Martin Fay's Vaz Modular sound synthesis software. The piece is performed on a Pentium II Windows 98 laptop running both programs simultaneously, which are linked by Hubi's Midi loopback driver. Originally, the piece started off just as a series of demonstrations of various algorithmic function modules in Softstep. However, somewhere along the line, it also became a piece of music which pleases me greatly. Perhaps this is because of the way simple things combine to make a complex whole. I remember back in the 1970s, Dary John Mizelle, surely one of the most important and yet mysteriously underrated complexist composers around, said to me that he liked to pile structure on top of structure until chaos resulted. I think that is what might be going on in this piece, and that, in addition to the pleasant FM timbres, may be why I find the piece so attractive.

The piece, in 13-tone equal temperament, consists of three strands of music. I like the way the bluesy, "compressed" intervals of 13-tone tuning combine with the FM timbres used in this piece. Two of the musical strands are timbrally similar, while one is both registrally and timbrally different. The first strand is a two voice canon, in which the leading voice consists of a low melody and a high melody, each produced by a different algorithmic process. These two melodies are switched between randomly, making a composite melody which behaves differently in the high and low registers. This composite melody is then played canonically by two voices, the second voice being delayed by 61 clock ticks (each clock tick is equivalent to a demi-semi-quaver here) and is transposed up by seven 13-tone semitones.

The second strand is structurally similar to the first, but uses different algorithms. That is, it consists of a low melody and a high melody which are switched between to make a registrally varying composite melody. Here, however, different algorithms than were used in the first strand produce the low and high source-melodies, and the two source-melodies are switched between in such a way that a gradual transformation from one to the other takes place. This composite melody is then also played canonically by two voices, but here the second voice is delayed by 53 clock ticks, and is transposed up by five 13-tone semitones.

Additionally, the timbres of the two FM strands are slightly different. The first strand is made by having the modulating oscillator (a triangle wave) two 12-tone semitones higher than the carrier (also a triangle wave), while the second strand has the modulating oscillator two 12-tone semitones lower than the carrier. In the second strand, both oscillators are triangle waves as well. Finally these two voices are processed through resonant filters, panners, and amplifiers, all controlled by low frequency sine-wave functions, so that the timbre, spatial position and amplitude of each voice slowly and continuously change.

The third strand uses an additive synthesis tone made with Nicholas Fournel's Virtual Waves software. This is controlled by an algorithm called the Thue-Morse sequence, which in this case, produces arpeggiated melodies with self-similar properties. This voice is played in a high register, and is only played occasionally in the piece. Its function is to act as a contrast to the more continuous sounding of the first two strands.

Strand 1

SoftStep has a number of different rhythm generators, and this piece uses three of them. The first strand is controlled by a cute little generator called a ball. In this, a "ball" bounces around a frictionless four sided space, its speed controlled by x and y directional inputs. Each time the ball touches one side of the space (the space may be sized to any rectangular dimensions desired), a pulse is given off. Further, contact with each side of the space generates a unique pulse, which can be extracted (in any combinations desired) with a Mask Logic module, so that one ball can control a number of different functions in different, but related ways. (I guess if I wanted to use techno-jargon, I'd describe the ball as a physically modelled virtual rhythm generator, but I prefer to think of it as just fun.) Here, the speed of the ball is controlled by a mouse pad controller (an x-y axis position controller), and each aspect of the structure of the first strand is controlled by a different combination of side-contacts of the ball.

As mentioned earlier, each of the first two strands is made up of a melody consisting of high and low register components. The first strand's high melody is made with the Bio-Sequencer module. This is a module which reads DNA or protein sequences and outputs the result as numerical information. It is a very complex module, but here I use it very simply. There are twenty amino acids which make up proteins. So if a protein sequence is being read, values from 1 to 20 are output, as each new amino acid is read on each new clock pulse. Here the protein sequence of the human blue cone pigment gene is simply read out as a series of 20 values to select between 20 possible high pitches. (Free sources of genetic sequences are the Swiss Protein Data Bank www.ebi.ac.uk/textonly/swissonly.html and the NIH GenBank www.ncbi.nlm.nih.gov/. There will undoubtedly be a race to have the first composition using the complete human genome, once mapping of it is completed!) Rhythmically, the Bio-Sequencer is controlled by the right wall of the ball module. The Midi Voice Out is triggered off by a more complex combination of the left, right and bottom walls of the ball, but the module is set in sustain mode so that only new pitch information will trigger off a new pitch. This prevents repeating pitches, something I found useful for these particular melodies. Those wishing to know more about the use of biological data in music are referred to "Life Music, The Sonification of Proteins" by John Dunn and Mary Anne Clark
http://geneticmusic.com.

(Demonstrate Example 1, the Ball and the Bio-Sequencer.)

The low melody of strand one is made with a combination of the Probability module and the Page module. The Probability module, one of my suggestions, allows a user to set up any probability table they want. That is, any of 128 elements can have its own probability of occurrence. One simply draws the probability curve one wants, and the output is weighted in that way. The module can then select, in real time, between 128 of these curves that are preset by the user, or, using the Table-Write module, any of these 128 curves may be reset in real time. The Page module is a sequencer module, where a preset sequence (drawn by the user in the "Fill" utility module), is accessed either sequentially or randomly. Again, it is possible to switch in real-time between 128 sequences. To make this melody, the Page module is switched between three tables, which give the closest possible 13-tone equivalents to an ancient Greek Dorian diatonic mode, and the major and minor scales. Every 47 notes the scale that is being used changes. (Keeping the lower component of this composite melody within some notion of "diatonicism" is my concession to the notion that bass lines ought to be simpler, harmonically, than melodic lines!) The choice of notes from these diatonic scales is made by the Probability module, which is using a probability curve that is continually being rewritten. The two modules doing the rewriting are a Fractal generator generating a Mira fractal to determine which step of the probability table is to be rewritten, and a Chaos generator set to the "Burt Shift" algorithm to determine what value to set the chosen step to. (The "Burt Shift" algorithm is a shift-register-feedback circuit which is similar to that used in my "Aardvarks IV" hardware machine, and also similar to that used in Greg Schiemer's "Monophonic Variations".) So the resulting low component of the melody will always have some notes of the scale played more than some other notes, but which notes these are will be always changing. Suffice it to say that the melody will NOT simply sound like the result of equally weighted random numbers, but will hopefully have a much richer moment to moment structure than that produced by a simple random generator.

Those wanting the details of the implementation of fractals in SoftStep are referred to the help file, and to Robert Greenhouse's "The Well-Tempered Fractal."

(Demonstrate Example 2, the Page module and low melody, strand 1.)

These two melodies, the low and the high, are now switched between, using the Corruption module. This is a module which allows either switching between or adding of two incoming streams of information. The switching or adding is controlled by a probability setting. That is, one can set probability to, say, 25%, and then there would be a 25% chance that for each new event, an element from the second input (here called "offset") would be either substituted for or added to the corresponding element from the first input. This module was suggested by an article by Laurie Spiegel, called "An Information Theory Based Compositional Model".

In this piece, the two melodies are put into the two inputs of the Corruption module. Then, any top-side contact on the ball module generates a trigger, which selects a different random number between 1 and 100. (This is a simple equally weighted random number generator. (Okay, for the pedants, it's a pseudo-random number generator, but it's close enough for jazz.)) This determines the probability of choice between the melodies until the next top-contact of the ball, at which time this probability changes. So the high melody is triggered off by right-side contacts of the ball, the low melody is triggered off by left- and bottom-side contacts of the ball, and the probability of switching changes with top-side contacts of the ball. This gives us the possibility of a wide range of selections from, and combinations of, the various rhythm pulses generated by the ball.

(Demonstrate Example 3, strand 1 switching.)

Finally, to thicken the plot, and add an element of traditional structure to this, the pitch and rhythm information for the composite melody is also delayed by 61 clock ticks, and is applied to a Midi Key out module (a slave to a Midi Voice out - it allows polyphonic pitches on the same Midi channel, but doesn't send any pan, program, or controller information) which is transposed up by seven 13-tone semitones. This gives a canon at a "somewhat-less-than-a-fifth" interval (646 cents, to be precise).

(Demonstrate Example 4, canonic imitation, strand 1.)

Strand 2

The second strand is controlled by the Rhythm generator. This is a clock which enables 7 synchronised rhythm patterns of up to 32 pulses to be generated and combined in many different ways. Each of the 7 rhythm patterns can have any combination of off and on pulses and can be of any length up to 32 pulses. Using the Mask Logic modules, these can be combined in any way desired. And the clock of the rhythm generator itself can be controlled from any external signal, allowing for tempo modulations of any desired degree of flexibility. In this piece, the seven patterns are set to lengths of 11, 13, 17, 19, 23, 29 and 31 pulses: prime numbers that ensure maximum length sequences resulting from the combining of their pulses. Each element of control of the second strand uses a different combination of these 7 patterns.

The high melody component of the second strand is controlled by a Chaos generator. This one uses the Henon attractor to generate its values. It is triggered off by a 13 against 17 pulse pattern from the Rhythm generator, and chooses 14 pitches for one very high octave of pitches in 13-tone equal temperament.

(Demonstrate Example 5, Rhythm Generator and strand 2 high melody.)

The low melody is controlled by an Image Generator module. This is a module which can read any 128 x 128 pixel graphic (bitmap) image and read colour values either singly, or in combination for each pixel. It can generate its own Mandelbrot and Julia Dragon images internally, or accept images from the Fractal generators, or use an externally generated image. I was tempted to use an image of the cartoon character Bill the Cat for this function, but Bill just didn't have the pitch variation I was looking for (ack!), so I settled on an image I generated using the "Toy Universes" cellular automata in James Gleick's classic "Chaos" program. The pixel to be read from the Image Generator is chosen with inputs to its x and y parameters. These can be counters for linear readings, or random generators for samplings of various areas of the image. Here, I'm using two counters, one of which is reading an 11:19:31 pulse from the Rhythm generator, the other of which is reading a 17:23:29 pulse from the Rhythm generator. This produces a quasi-random walk across the image, moving generally in a downward diagonal direction, but deviating from a straight diagonal line according to the combination of the pulses. The values the image produces from this quasi-random walk across it are then scaled so they control a range of 16 low adjacent pitches in 13-tone equal temperament.

(Demonstrate Example 6, low melody produced with Image Generator, strand 2.)

As in strand 1, these two components are put into a Corruption module, and a composite melody is then produced by switching between them. A counter (controlled by a 17:23:29 pulse from the Rhythm Generator) simply counts from 0 to 127. This count is put into the probability control on the Corruption generator. Low numbers from the count mean there will be a greater probability of the high melody notes being chosen. The higher the count gets, the greater will be the probability of low melody notes being chosen. So for each time through the count, there will be a different transition from mostly high melody notes to all low melody notes. Note that the probability generator reads all values of 100 and above as a 100 percent probability that the second input will be read. Therefore, this progression, using a count from 0 to 127, is biased in favour of the low melody notes.

(Demonstrate Example 7, switching to produce strand 2 composite melody.)

Finally, this second strand of composite melody is also delayed canonically, this time by 53 clock pulses, using the Delay modules, and the result is transposed up five 13-tone semitones, for transposition upward of "less-than-a-perfect-fourth" (462 cents, to be exact).

(Demonstrate Example 8, strand 2 canon.)

Strand 3

In performance the tempo of these two voices is constantly changed. The ball is used to control the tempo of strand 1, while a bar control at the bottom of the screen is used to control the tempo of strand 2. But on top of these two, a third strand is sometimes added. This is a high melody of contrasting timbre (and one which doesn't pan) which is controlled by the Number Generator module. This is an expanded implementation of the Thue-Morse sequence, a number sequence which exhibits great self-similarity.

A good, simple explanation of the Thue-Morse sequence can be found in Gustavo Diaz-Jerez' article "Fractals and Music" in the October 1999 issue of "Electronic Musician." In this implementation, the sequence can count in any base from 2 to 127, any step size can be used, and one can begin from many different points in the sequence. And each of these parameters can be re-set in real time. For example, I randomly change the step size between 1 and 2 while generating my melody. This produces a melody with two sizes of basic generating intervals, 13-tone "seconds" and "thirds". Further, the output of this sequence is added to an offset and then fed back into the input of the Clock (the third, and simplest type of rhythm generator used in this piece) which is driving the Number Generator. This makes low notes have a shorter duration than high notes. In performance, the offset is changed with a second bar control at the bottom of the screen, changing the tempo of this Thue-Morse melody. This is a melody of upward arpeggios, and the pitches that begin each new upward arpeggio themselves form an upward arpeggio, etc. I included this melody as a timbral foil to the first two strands, and also to have one very clear example in the piece of one melody that was unambiguously produced by one clearly heard algorithm.

(Demonstrate Example 9 - Number Generator melody.)

Conclusion

In *Brisbane Nocturne,* I use a wide variety of algorithms to control different parameters of the piece. Normally, I wouldn't do this - my propensity is to thoroughly explore one kind of algorithm in a piece. However, the desire to show a number of different functions of SoftStep at once made me choose to structure the piece in this way. Conceptually, the structural result should probably be, quite simply, a mess. However, my ears tell me it's not. I hear both variety and coherence in this piece - of a sort that intrigues and satisfies me. I think what is happening here is that the end result of combining all these individually simple structures is a musical object which has a lot of subtlety to it, and this subtlety works on a variety of levels. So, for my ears, the piece has a satisfying structure.

However, there's a larger issue here, which is what I see as our increasingly complex understanding of the worlds of chaos, randomness and algorithmic structure. In the early aleatoric works of Cage and Xenakis, for example, one algorithm is often used. This gives the works great structural purity, and despite their sometimes chaotic surfaces, often a great sense of underlying simplicity. Now, with so many kinds of algorithms so easily available, we can quite freely combine them into structures which will possess an underlying sense of structural intricacy. (Whether the resulting music will be any better or not is up for grabs, I'll admit, and is probably irrelevant to the point I want to consider here.) In this progression I see an analogy (a fairly shoddy analogy, its true, but....) to what happened to musical structure in the mid-18th century. In the first works of the Mannheim school, the contrapuntal complexities of the Baroque were cast aside in favour of simpler homophonic structures. By the middle-late decades of the century, in the works of such composers as
C. P. E. Bach, contrapuntal thinking began to work its way back into the music, and structures of greater complexity were made, resulting finally in the structures of the late-Classical music of the 1780s and '90s. In my view, both the early works of Cage and Xenakis, and the earliest of minimalist works also were made in this same "paring-away" manner. These works, for this analogy, are the equivalent of the early Mannheim works. Now, with our greater understanding of how probability works,

and of the nature of a plethora of different deterministic and non-deterministic structures, we are perhaps on the verge of creating works which will be the structural analogies to the middle period works of C. P. E. Bach. Producing works with these tools that could be considered analogs to the complex late-Classical works of the 1780s and beyond is a task that could provide some fun for the next few years.

Thanks

A huge thank you to John Dunn for his inspiration and effort in developing SoftStep. Also immense thanks to the Music Fund of the Australia Council for their 1998-2000 Composers' Fellowship, during which time I worked on this software, this piece, and this paper.

References

Burt, Warren. 1975. "Aardvarks IV - A Real-Time Electronic Music Performance Machine." Unpublished Masters Thesis, University of California, San Diego.

Diaz-Jerez, Gustavo. 1999. "Fractals and Music." *Electronic Musician* October:108-113.

Dunn, John. 1999-2000. "SoftStep" Help Files. http://geneticmusic.com

Dunn, John and Clark, Mary Anne. 1997. "Life Music: The Sonification of Proteins" now included in "SoftStep" help files, ibid. http://mitpress.mit.edu/e-journals/ Leonardo/isast/articles/lifemusic.html

Gleick, James and Rucker, Rudy. 1991. "Chaos - The Software" Autodesk, Inc.

Greenhouse, Robert. 1995. "The Well Tempered Fractal" User Manual.

Schiemer, Greg. 1989. "Monophonic Variations." NMA 6. NMA Publications, Melbourne.

Spiegel, Laurie. 1998. "An Information Theory Based Compositional Model." Leonardo Music Journal 7(January).