INFNO: GENERATING SYNTH POP AND ELECTRONIC DANCE MUSIC ON DEMAND

Nick Collins
University of Sussex, Department of Informatics
N.Collins@sussex.ac.uk, http://www.informatics.sussex.ac.uk/users/nc81

ABSTRACT

Infno is an algorithmic generator of electronic dance music (EDM) and synth pop fully implemented in SuperCollider 3, the latest generative music work in a line of 'Infinite Length Pieces'. The program attempts to model the production of electropop and dance music styles with a closer union of parts than typical in many previous algorithmic composition systems. Voices, including a percussion section, bass, chord and lead lines, are not independently created; the parts, generated in any order, are influenced by underlying harmonic ideas, rhythmic templates and already generated lines. In particular, dynamic programming is used to select melodic lines under cost constraints of register, harmonic template, existing voices and voice leading heuristics.

This paper is the first technical write-up of Infno's mechanisms, and also reports evaluation by online questionaires and a seminar listening survey. Source code and musical examples are available at the author's web site.

1. GENERATIVE POPULAR MUSIC

Infno is a generative music program written to demonstrate on-demand creation of convincing musical pieces which engage popular electronic styles and employ live synthesis and effects. Research goals include substantial variation of output between each run, and an investigation of the close inter-relation of parts alongside arbitrary part creation order.

Whilst the *Illiac Suite* is the more famous, a highly relevant early algorithmic composition is *Push Button Bertha* from 1956 (Datatron program designed by Douglas Bolitho and Martin Klein, with the output melody line set to lyrics by Jack Owens [1]). Despite a failure in the pop charts (perhaps linked to only receiving one radio play), the engagement with popular music is itself of note, and it provides an inspiration to this project. Another precedent from the algorithmic composition literature is Ames and Domino's *Cybernetic Composer* [2], which tackled MIDI data generation in four styles (standard jazz, latin jazz, rock and ragtime) as a demonstration program for Kurzweil synthesisers.

An early example where live synthesis is built into the program is provided by the various releases of Arguru and WakaX's *Saiko* (2000), a goa trance simulator. It is not fully autonomous, requiring manual control of many parameters, and does not deals with any longer term form than a single (at most) 64 step pattern. In 2002, the MadWaves company released a generative music player as part of a portable MP3 player, the *MadPlayer*. Tackling a number of styles using six instrumental parts, this proprietary device creates music on demand and allows the realtime substitution of parts, and the export of favourites; it can operate autonomously, with additional options for user tweaking. The device uses sample sets for synthesis and an internal MIDI representation. A more recent venture is Mungo Enterprise's *Infinite Horizon* (2006) a hardware box which can generate techno tracks constructed of standard voicings, with manual user control of form.

My own first (MIDI based) techno generation program was written in 1997 in York as an exercise in C programming; I have pursued such research extensively over the last decade. Algorithmic composition experiments with UK garage and jungle styles (amongst many others) have been previously described [5], and I might mention a contribution to John Eacott's *Morpheus* CDROM of 2001, and the various manifestations of BBCut. [1]

My most recent project attempts to up the scale of automatic music engaging with contemporary styles. It is both a compositional response to, and an investigation of, new technological directions, as well as a necessarily musicological study through heuristic modelling. The genre labels 'synth pop' and 'techno' are used to indicate a general stylistic space of action, and to side step the dance music culture's genre fixation. Synth pop is treated since pursuing techno alone would leave the author open to accusations of modelling too simple a style (even though I would argue that many authors substantially underestimate the many facets of musical complexity across EDM). Live synthesis is used for all voices, and the largest samples are at the level of 100-250msec sources for sample playback synthesis, primarily for percussive events. The paper will describe the main aspects of the program's implementation, alongside evaluation based on internet surveying and seminar groups.

2. OVERVIEW OF INFNO

Describing every detail of Infno would be exhausting; ultimately the program is its own best description, but I will

[1] A chapter on this work appears as part of my 2006 PhD thesis.

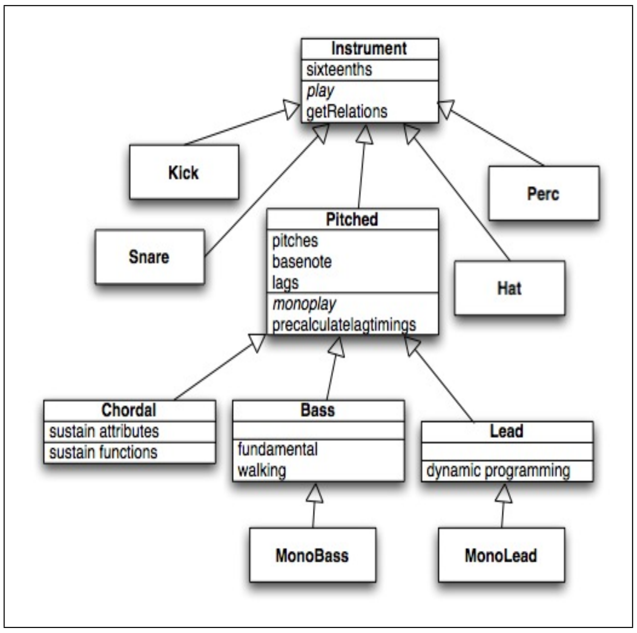

Figure 1. Instrument class and subclasses

provide a guide to notable aspects of the design and technological novelties.

A note should be made concerning the balance of machine learning and expert system in this project. Whilst a database of chord sequences has been developed for one of the harmonic models, Infno uses a vast number of manually specified rules (heuristics). In terms of Pearce et al. [7] it is primarily an 'algorithmic composition' and not empirical musicology. Although it engages with contemporary styles, the impossibility of portraying them without also influencing their development through technology is acknowledged in advance.

2.1. Object oriented design

Unified Modeling Language class diagrams are presented in figures 1 and 2, highlighting the main inheritance, associative and container [2] relations of the Infno class hierarchy. Many attributes and operations have been missed out for reasons of space, and to focus on essential design decisions.

The abstract **Instrument** superclass represents a particular compositional voice, of which there are nine available by default (kick, snare, hat, perc(ussion), bass, leads 1 to 3, and chord). Subclasses implement the **play** method polymorphically to playback n elements of a 64 step sequence (by default, representing **sixteenths** over four 4/4 bars). This innate quantisation is appropriate to the stylistic domain, and whilst timing can be adapted by groove and fill parameters, provides a basic framework for generation and playback. Monophonic/glissando playback capability is provided in the **Pitched** class as a common ancestor of **MonoBass** and **MonoLead**; the **lags** attribute and **precalculatelagtimings** operation allow the generation, in advance of playback, of necessary additional scheduling parameters for a monophonic sequence. The **Chordal** class has instance variables for sustain parameters, and various specialist material generation instance methods are noted for basslines and the melody generation algorithm based on dynamic programming to be discussed below. Indeed, each class has its own methods for the generation of appropriate material.

[2] marked as 1 to many (*) in UML

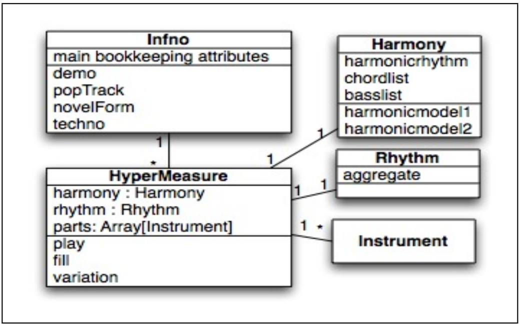

Figure 2. Infno main classes

Each voice can support or oppose other previously created parts, such that the order of creation of the parts impacts substantially on the algorithmic composition. The chief innovation is the **getRelations** method, which, given a specified list of preferences of the form [voice, chance of opposition] runs through that list to find the first existing voice which will influence material generation for the new voice to be created. Support or opposition of voices is primarily of rhythmic material but becomes pitch based for the lead parts in particular, through the dynamic programming cost algorithm detailed below.

The second UML diagram in figure 2 shows the main classes in Infno; **Infno** itself is the central singleton class whose single instance represents the generative program. The various methods from demo to techno will be described below and enable particular forms for generative pieces created on demand. Both **Harmony** and **Rhythm** are abstract template classes which provide chord sequences and aggregate probabilistic rhythmic templates as starting points for material generation within a given HyperMeasure, from which the individual **Instrument** subclasses are guided in creating parts; **Harmony** will be more closely studied in the next section. Each **HyperMeasure** is generated as a four measure phrase, though they may be played back over a smaller number of beats and under various fill and variation operations which subtly or strongly alter existing material.

2.2. Harmony generation

Two main methods for generating harmonic templates have been designed. Both create a list of chords (for the chordal part and as dyamic programming constraints) and bass root notes (for the bassline generation) with an accompanying harmonic rhythm (allowing for example a three beat then one beat pair of chords in a bar rather than a single full measure chord).

The first method draws on a database of typical pop chord sequences, described either via a diatonic representation of (duration, degree) for the quick specification of major/minor chord pop sequences or a chromatic system of (duration, root chroma, chord type) for more involved sequences. There is a chance of variations on the database sequences by rotation and permutation, root alteration and transposition.

The second method is a rather more complicated heuristic system. Chord qualities are altered and extended by sevenths, ninths, elevenths, by suspensions and other harmonic devices. Chord substitutions and root inversions are available, as well as the Prokofievan device of an entirely nonstandard bass root note.

Whilst C major is the home key, there are chances for both global and local (hypermeasure) key changes. An instance of the **Harmony** class holds the data after generation and provides various helper methods for creating possible matching scales and querying harmonic data at any given point in a hypermeasure.

2.3. Dynamic programming algorithm for melodic lines

The most technical part of the Infno program is a dynamic programming algorithm for melodic line generation which takes into account existing parts, harmonic constraints and note transitions. In order to explore different possible melodic algorithms, the costs associated with the dynamic programming are altered (within controlled bounds) for each run, following certain heuristics, which allows for different characters for each run.

The full dynamic programming algorithm would take much longer than this paper to reproduce, so only important facets are discussed here. For each required note at a given step of the sequence, costs are determined for every chromatic tone over a two octave range (24 possibilities) and with transition cost from the last 24. Contributions to the calculated cost value of each option are detailed in Table 1. The numerical ranges give a rough idea of the relative values costs can take on, but the actual distributions guiding the selection of values for each dynamic programming run are more complicated and locked to SuperCollider code, so are not further detailed here. The curious reader can examine the source code method **InfnoLead:dynamicprogramming1** for further information.

The **Harmony** class provides functions to find the best fitting scales at a given moment in time, taking into account local chord choices. Diatonic notes are then determined from these scale patterns. Deliberate noise can be added both at the individual transition and best path selection stages.

A final cost for each possible transition from note i to note j at a given time in the hypermeasure is established based on:

$$\text{cost}(i,j) = \alpha{*}\text{transition}(i,j) + \beta{*}\text{contour}(j) + \gamma{*}\text{penalty}(j) \quad (1)$$

Cost	Details of numerical cost
repetition	accumulates based on number of repetitions (1-10 per prior repetition)
tritone	transition penalty for a specific interval 0-20
transition cost	absolute distance from previous note * (0-2)
contour cost	absolute distance from ideal contour for part
chromatic tone	8-25
diatonic tone	4-10
chord tone	based on metrical position: half note: 1-4 quarter note: 0.0-2.5 other: 0.0-0.5
voice interaction	penalty for being on another voice's notes: 0-10

Table 1. Table of dynamic programming costs

where the weighting constants α,β and γ are themselves determined for each run (in the range 0.0 to 1.0). Each summand is determined from the various cost factors illustrated in the table as split up by horizontal lines. The contour cost refers to separation from an ideal contour determined prior to the run of the dynamic programming algorithm.

In practice, the best path (which will directly yield the final melody) is determined by full dynamic programming with forward and backward (Viterbi) stages. However, for speed of calculation, an option is available to use a *greedy* online version [8] which will simply take the best pitch at each stage during the initial forward calculation.

2.4. Infpop and Infno: Synthpop and techno generation modes

Two main playing modes are provided, built out of the musical machinery described in previous sections. Each run of the program in one of these modes selects a new appropriate tempo, groove, some **InfnoMix** instances which determine which synthesis methods and effects are employed from a database for each part, and a set of HyperMeasures as musical materials.

Synthpop mode generates popular song forms, based on the probabilistic juxtaposition of intro, riff, verse, bridge, chorus and middle8 sections following various heuristics, varying for each section the number of repeats and instrumentation, and allowing certain types of fill (both eighth note permutations and rolls/stutters). The *techno* playing mode generates tracks over a range of tempi, implicating various subgenres. [3] The formal construction follows the layer-based work of techno, where gross changes are rarer, whilst instrumentation (the current combination of parts) is often varied one part at a time (perhaps by adding one extra element every four measures). Whilst there are a few other playing modes (a demo mode, and a novelForm mode which explore non-standard popular song structures) these two are the most developed and are the subject of the evaluation.

[3] Electronic dance music categories are notoriously fickle, often socially and commercially constructed, and no further analysis along these lines is attempted here except to observe that allusions to certain styles naturally fall out of tempo and groove choices in combination with instrumentation and materials.

3. EVALUATION BY PEER FEEDBACK

Feedback on the Infno project has been solicited from a number of sources, from workshop and seminar presentations, to an official source code release and accompanying user survey. [4] The majority of data were qualitative, though rating scales were used for a number of questionaire responses, and a 'spot the difference' listener survey was carried out at a seminar where 24 respondents tried to distinguish Infno's output from manually composed output, as well as to rate individual tracks. Any choice of testing procedure can be problematised for the subjective and social domain of music, so the primary aim of evaluation was to gather feedback as guidance for future iterations of the software. Certainly, 'musical output Turing tests' are problematically disconnected from interactive conversation [3], and Turing will not be invoked further here.

Key questions were to assess the musical quality or appropriateness of synthesised outputs, to have users consider the variation in output allowed by this generative musical artefact, and to informally probe John Eacott's 'earworm humming test' for generative music, of the memorability of output constructions [6].

As much as anything else, the seminar listening task revealed much about the individual subjectivities of the participants. Six thirty-second extracts from techno tracks were played; subjects were instructed to rate each track on a scale from 1-7, and to mark each as either human-composed (at a standard sequencer) or automatically generated by Infno. They were further instructed that three of each type were present, though in fact there were two human and four Infno generated tracks, the rationale of this deception being to find the Infno track considered 'most human'. No statistically significant results (both correlation and t-test) were found to distinguish perception of human and computer generation, nor quality (as some subjects wrote in the comments boxes on their experimental forms, 'I found it very difficult to discern any difference!' or 'no correlation between quality and hand/auto generation'). There was no correlation either between the attribution of direct manual human authorship and quality, demonstrating a lack of bias against machine composition per se. Electronic music raises the spectre of algorithms up front, as noted by a number of participants, who discussed arpeggiators in sequencers, but perhaps overestimated the automation used by dance music producers already. However flawed the experimental setting, the listener survey became a useful basis for discussion, and participants provided much constructive criticism on the musical quality of outputs and the status of the generative music artefact.

Infno's source code had also been released, with a questionaire included in the download. Users were encouraged to fill in the questionaire as they used the program for the first time. Aside from musical background, subjects gave qualitative and quantitative responses on issues of musical quality and variation of output, and memorability. Variation was consistently rated more highly than quality. The questionaire collection is ongoing, and may be built into a standalone release at a future point. Indeed, third parties have become involved in the testing cycle; Chris Jeffs was kind enough to build an Infno DJing application with built in track rater!

There is no space to go further into individual feedback in depth here, however, conference presentation would give an opportunity to expand upon this evaluation.

[4] For background information on evaluation methodology and internet surveying in particular, see, for example, Cohen et al. [4]

4. CONCLUSIONS

Potential applications for Infno range from installation and fixed composition outcomes, through the automatic generation of material for beat tracking experiments, to musicological engagement with contemporary musical styles and even new generative performance practices such as generative karaoke. Indeed, it is in the latter capacity that a prototype of Infno appeared at last year's SuperCollider symposium, with Takeko Akamatsu daring to improvise a new song to a backing track she could not possibly have heard before, using the lyrics from Push Button Bertha. Further work in that direction might necessitate incorporation of various (existing, third party) automatic lyric/poem generation programs, and even the composition of vocal lines themselves, perhaps with guide synthesis from Vocaloid or similar programs.

Future quantitative evaluation may be carried out, thought such work is dependent on 'freezing' the version of the program temporarily for the purposes of review. It is tempting to employ secret shopper techniques from posting generated tracks anonymously or under psuedonyms or under submission to demo tape criticism.

In the mean time this project stands as an exhortation to take seriously standalone generative music systems of more ambitious scope, as potentially strong exemplars of the computer music field's progress and capability.

5. REFERENCES

[1] C. Ames. Automated composition in retrospect: 1956-1986. *Leonardo*, 20(2):169–185, 1987.

[2] C. Ames and M. Domino. Cybernetic composer: an overview. In *Understanding Music with AI: Perspectives on Music Cognition*, pages 186–205. The AAAI Press/ MIT Press, 1992.

[3] C. Ariza. The interrogator as critic: The questionable relevance of Turing tests and aesthetic tests in the evaluation of generative music systems. *Computer Music Journal*, under review, 2008.

[4] L. Cohen, L. Manion, and K. Morrison. *Research Methods in Education (6th edition).* Routledge, New York, 2007.

[5] N. Collins. Algorithmic composition methods for breakbeat science. In *Proceedings of Music Without Walls*, De Montfort University, Leicester, June 2001.

[6] J. Eacott. *Contents May Vary: The Play and Behaviour of Generative Music Artefacts.* PhD thesis, University of Westminster, 2006.

[7] M. Pearce, D. Meredith, and G. Wiggins. Motivations and methodologies for automation of the compositional process. *Musicae Scientiae*, 6(2), 2002.

[8] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach (Second Edition).* Prentice Hall, Upper Saddle River: NJ, 2003.