Automatic Notation Generators

G. Douglas Barrett
State University of New York
at Buffalo
Department of Music
311 Baird Hall
Buffalo, NY 14260-4700
(617) 372-0737
gbarrett@buffalo.edu

Michael Winter
University of California,
Santa Barbara
Media Arts and Technology
Santa Barbara, CA 93106-6065
(434) 295-4113
mwinter@sonicism.net

Harris Wulfson
Graduate Center of City University of
New York
Department of Music
365 Fifth Ave.
New York, NY 10016
(661) 373-5611
harris@wulfson.com

Abstract
This article presents various custom software tools called Automatic Notation Generators (ANG's) developed by the authors to aid in the creation of algorithmic instrumental compositions. The unique possibilities afforded by ANG software are described, along with relevant examples of their compositional output. These avenues of exploration include: mappings of spectral data directly into notated music, the creation of software transcribers that enable users to generate multiple realizations of algorithmic compositions, and new types of spontaneous performance with live generated screen-based music notation. The authors present their existing software tools along with suggestions for future research and artistic inquiry.

Keywords
Automatic Notation, Algorithmic Music, Real-time Notation Generation, Spectral Analysis, Graphic Notation, Interactive Music, Interactive Composition

1. Introduction to Automatic Notation Generators
Automatic Notation Generators (ANG's) are computer applications that generate notated music based on mappings of data onto specific notation systems. The authors intend to show that ANG's can enable new modes of compositional thinking and facilitate experimentation with notation systems.

The authors began work on notation generators independently until creating a forum dedicated to the topic of ANG's. Since then, the development of more specialized ANG's has continued, culminating in evolving software systems and the creation of The Society for Automatic Music Notators (SAMN), which convened in New York City for the first time in the fall of 2006.

To illustrate the functionality of ANG's, recent software by the authors will be described along with examples of their use. These include ANG's that transcribe spectral data as in the work of G. Douglas Barrett, ANG's for algorithmic compositions by Michael Winter, and Harris Wulfson's live real-time ANG's for screen-based notation. Ultimately, with these examples we hope to illustrate a music making process wherein the notation software is an integral part of the composition.

2. Spectmore: Automatic Notation Generation for Spectral Transcription Processes

2.1 Introduction
This section introduces a newly developed software tool designed to automatically produce instrumental scores based upon spectral analysis and re-synthesis processes using sound-file input. The original software, Spectmore, and its various components will be described and an initial musical work produced using the software will be presented. Overall, this project has drawn significantly upon ideas and insight gained from previous projects (specifically, see Barrett, 2006) and is much indebted to the work of a key set of composers: the large-ensemble works of the Itineraire Spectral composers, along with many works of Peter Ablinger, and the algorithmic music of James Tenney. In addition to providing an overview of the program's general functionality, an attempt will be made to illustrate the value of the ANG in its ability to provide a user with rapid feedback and fast prototyping when composing music using algorithms and spectral analysis.

An interest, which grew from experimental works produced with Spectmore's predecessor, Spectore, was the use of environmental field recordings of extended duration, along with an exploration of their spectral properties. Peter Ablinger's *Quadraturen* series, in part, is similarly concerned with creating instrumental pieces from spectral analyses from a variety of sources, including field recordings. Central to some of these works is the use of polyphonic instrumental textures that can be considered analogous to the original recorded material with respect to certain morphological properties. With these interests in mind, a central goal of the Spectmore project has been to create a working environment in which various configurations of spectral analysis, algorithmic composition, and automatic notation generation could be rapidly and flexibly configured.

2.2 Spectral Orchestration
Spectral orchestration is the term given to the assignment of an instrumental tone with somewhat analogous properties – frequency to pitch, amplitude to dynamic – for each significant partial found in a spectral analysis of a recorded sound file. A partial is defined as significant when it passes tests for its duration, pitch, and amplitude. This process creates a collection of partials that shares structural properties with the original recorded material. Significant partials are assigned to instruments (Figure 2.3) based upon a complex set of time-variant and static criteria (see Section 2.4.2, Statistical Orchestration).

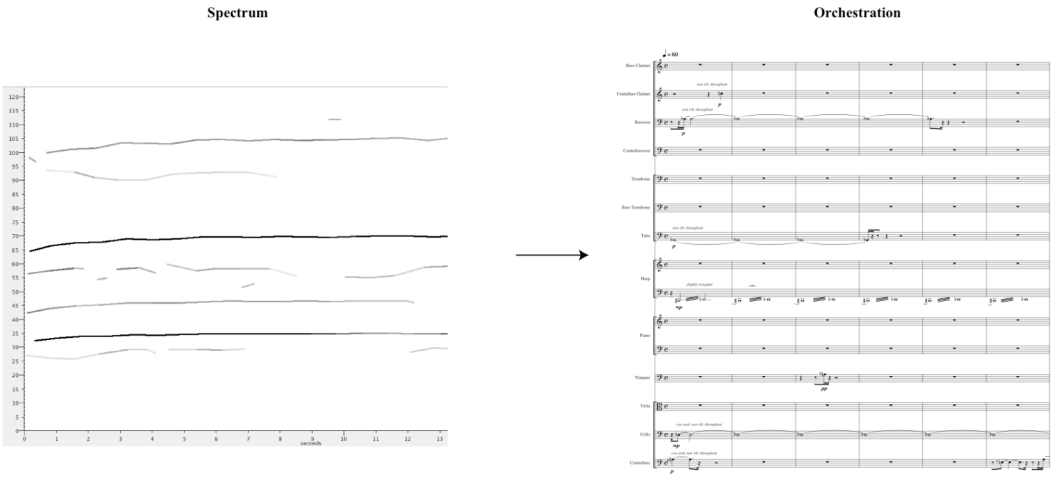

Figure 2.1 Spectral Orchestration.

2.3 Notation

Notation in Spectmore is handled by a specialized Python module (mxml.py), which contains certain musical and notational functionality to facilitate quick mapping schemes of input data. This module, uses as its eventual output format the widely accepted Music XML standard (Recordare, 2003). The module is constructed hierarchically as illustrated in Figure 2.2: a Note belongs to a Measure, which belongs to a Part, which belongs to the entire Orchestra. As differentiated from the Note object, the Plank describes a single tone that may span several measures and is attached directly to the Part object.

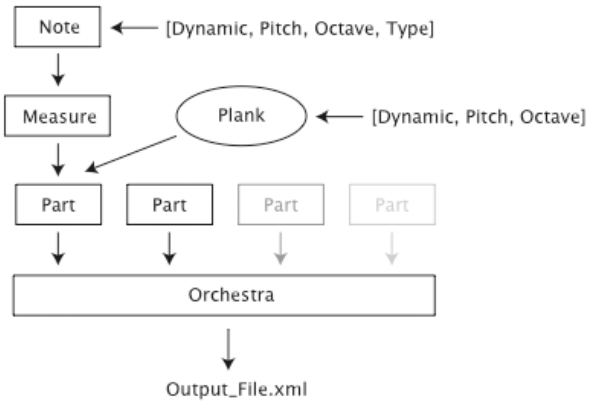

Figure 2.2 Internal structure of Notation Module (mxml.py).

2.4 Functionality

2.4.1 Spectral Analysis

Spectmore uses as its analysis engine Loris, the open-source software package, which implements the reassigned bandwidth-enhanced additive sound model. Loris provides a set of time-variant amplitude and frequency envelopes with initial phase values and noise energy (Fitz, et al, 2003). After obtaining a list of partials from a Loris analysis instance, the Spectmore analysis module performs a thinning function, which removes partials below user-specified amplitude and duration thresholds (Figure 2.3). Then, the changes in frequency throughout the duration of each partial are averaged to provide single frequency values. A similar averaging process is performed for amplitude. This collection of partials, to be referred to as a *spectral separation*, is then arranged in order of decreasing amplitude throughout time. This ordering gives more prominent partials priority when the spectral separation is handed to the Statistical Orchestration module.

2.4.2 Statistical Orchestration

For each partial contained within a spectral separation, an instrument is chosen from a previously defined collection, contained within the Orchestra object, based upon a system of statistical weightings (Figure 2.4). These weightings are specified by user-defined criteria such as fixed instrumental properties and time-variant states of the instrumental texture. Each of the following static weightings relate to instrumental properties and can be more or less emphasized by a user-defined factor (0-9): the *choir weight*, a weighting based simply upon the choir to which the instrument belongs (i.e. strings, woodwinds, brass, percussion), and the *range weight*, the partial's relative distance from a 'sweet-spot' chosen for each instrument (default, the middle of the instrument's range). Time-variant weightings include functions relating to voice-leading and the control of overall density. A distance in semi-tones from the last note of each instrument determines the voice-leading weight while the duration from each instrument's last note is used to control density.

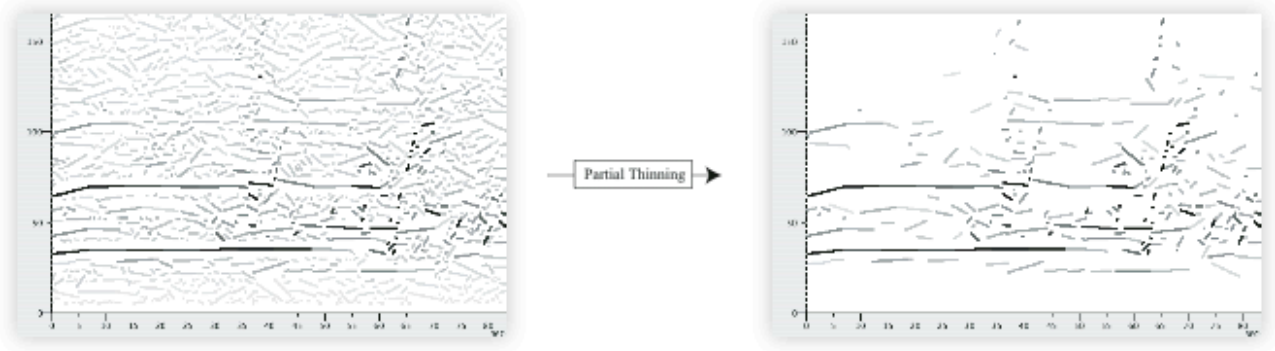

Figure 2.3 Illustration of Partial Thinning.

2.5 Music

Derivation V. for the S.E.M Ensemble is the first piece created using Spectmore and received its first performance on February 15, 2007 in Brooklyn, New York. The piece uses as its source material a recording of a busy street corner, Hollywood and Vine, located in Hollywood, California. The piece lasts approximately eleven minutes and consists of an instrumental texture, which moves from extremely sparse to moderately dense. Due to a concentration in low-frequency spectral energy, an ensemble of instruments with strong lower registers was chosen. While an adjustment could have easily "corrected" or equalized the spectral distribution, this unusual concentration of low-registral material seemed to bear an interesting relationship to similar observations made concerning the city soundscape (most notably, see *Increased Bass Response in Music and the Soundscape,* Schafer, 1977).

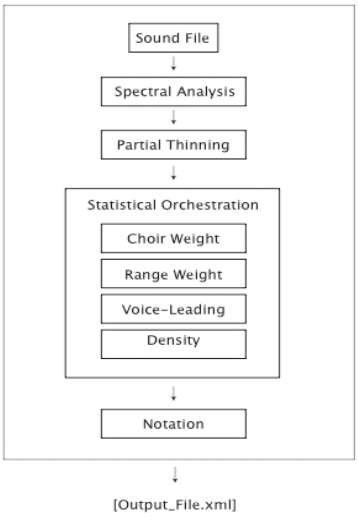

Figure 2.4 Spectmore: General Program Overview.

2.6 Conclusion

Spectmore provides a robust interface for algorithmic spectral composition. This ANG facilitates the kind of rapid prototyping and flexibility integral to working with experimental configurations of algorithmic music, transcription, and spectral analysis. Two areas that Spectmore leaves open for further development are user-interface design and general interactivity. The following sections address these issues among others.

3. ANG's for Algorithmic Compositions

3.1 Introduction

One exciting characteristic of ANG's is the ability to conceive of musical works as abstract structures with multiple realizations. Each ANG presented in this section allows users to create multiple realizations of one piece that is generated by an algorithm with stochastic and user defined-variables. The software also illustrates the kind of rapid prototyping employed in Spectmore with the added functionality of a graphical user interface. The corresponding works for these ANG's, *nothing… I* and *sort 1*, are compositions in which the structure and form of the piece are constant, but variable parameters such as instrumentation and tone information (pitch, duration, and amplitude) only become fixed when a realization of the score is generated for a particular performance. The compositional methods in these pieces draw from the considerable amount of work that has been done in algorithmic composition within the past few decades (Polansky, 1996; Ames, 2005 and 2006; Xenakis, 1971) and recent developments in ANG technologies. Nick Didkovsky's JMSL and JScore create an elegant link between an algorithm and its subsequent transcription into music notation (Didkovsky, 1997 – 2007).

To realize scores for *nothing… I* and *sort 1*, a modular framework has been implemented in Java (with some objects from the JMSL API). Each of the ANG's for these pieces share this framework (Figure 3.1) and share similar user-interfaces. Significantly, each also creates output using its own unique notation system. The ability to define individualized graphic environments for each ANG has afforded the composer the opportunity to create a notation system best suited for each piece.

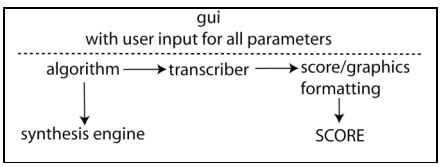

Figure 3.1 ANG Modular Paradigm.

3.2 *nothing… I*

3.2.1 *The Algorithm*

nothing… I is stochastically generated by an algorithm that defines limits for tone parameters such as pitch, amplitude, and duration. Temporal densities vary throughout the piece based on register. Figure 3.2 is a visualization that represents the changes in temporal density. Within the graph, the *x*-axis represents time and the *y*-axis represents pitch. Darker areas indicate louder and shorter tones while lighter areas indicate softer, longer tones. Completely white areas not bounded by lines indicate no sound at all. Proceeding from time-zero to the end of the piece, pitches are chosen randomly from an available pitch-range that changes over time. Then a third value is calculated based on the current time and the chosen pitch. From this third value, center points are derived for the range of possible durations and amplitudes. Then, time (the *x*-value) is incremented by the duration and the process is repeated.

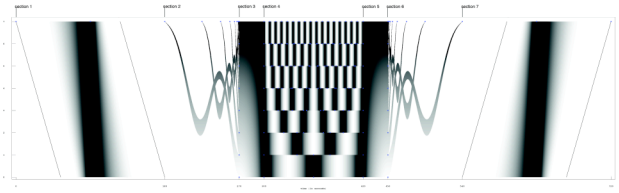

Figure 3.2 *nothing… I* Graph.

3.2.2 *Orchestration*

As in Spectmore, tones are assigned to instruments by statistical means. Since instrumentation is variable, orchestration is determined by factors of similarity and dissimilarity (Tenney, 1961) in timbre between groups of instruments that are user-defined as opposed to composer prescribed. In *nothing… I*, these groups are entered by the user in the instrument frame as "type" along with other attributes such as playable range (Figure 3.3). For example, type 1, type 2, and type 3 may be strings/brass, winds/brass, and strings/voices, respectively. The program first checks to see if any instruments of the preferred type are available, i.e. whether a given pitch is within the instrument's playable range and whether the instrument is already sounding a tone. If no instrument of the preferred type is found, the orchestration module checks the rest of the instruments in the user-defined ensemble. If no instruments are available, the tone is discarded.

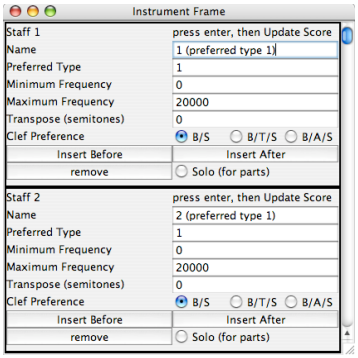

Figure 3.3 *nothing… I* Instrument Frame.

3.2.3 *Transcription and Notation*

After the tone information is generated and assigned to different instruments, there is a transcription module that renders the data into notation. The unique characteristics of the notation for *nothing… I* include proportional notation, cent-deviations above the notated pitches, and amplitude contours of tones represented in beams. In order to preserve the proportional notation, auxiliary information that does not fit between notes is placed above the note to which it applies. A fragment of the score is shown in Figure 3.4.

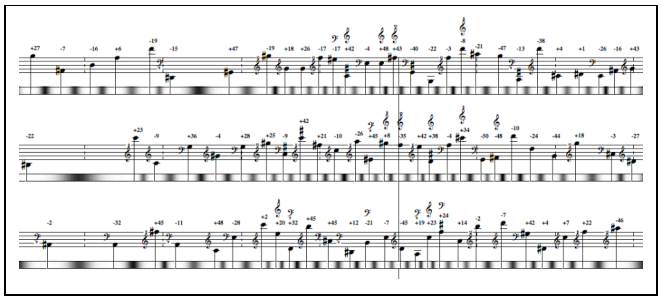

Figure 3.4 *nothing… I* Score Fragment.

3.3 *sort 1*

3.3.1 *The Process*

In *sort 1*, a set of pitches is iterated through several times. In the first iteration, the entire set is out of order. In each successive iteration, the set is rearranged; by the final iteration, the set is in descending order of pitch. The rearrangements are actually worked out in reverse by starting the set in descending order and moving the pitches to random positions until the set is completely scrambled. This is done such that the number of iterations is equal to the size of the set divided by the number of pitches to be moved per iteration.

3.3.2 *More on User-Interfaces*

The user can define any set, enter it in the Scale Frame (Figure 3.5), and then assign scale indices to different instruments. The piece can be rendered multiple times and each rendering can be formatted, saved, and printed (Figure 3.6). The program also includes a playback engine that allows the user to play a synthesized realization of the piece. This playback can be used as an accompaniment to acoustic instruments or just for audition.

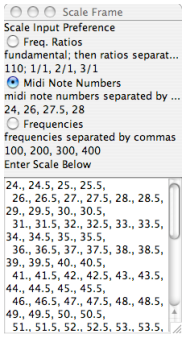

Figure 3.5 *sort 1* Scale Frame.

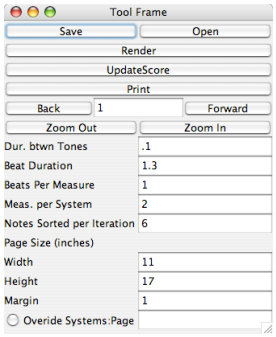

Figure 3.6 *sort 1* Tool Frame.

3.4 Projected Development Ideas

With each new ANG, additions are being developed. For example, in the ANG for *Rise I* from *4 Ascents for James Tenney,* the user can select notes on the screen and change their properties in order to format the score. This piece also uses a new, unconventional notation system (Figure 3.7).

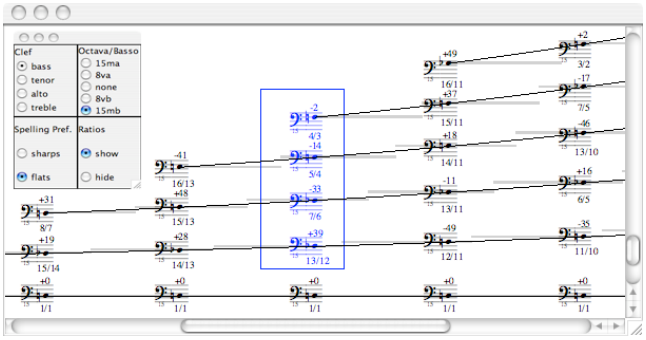

Figure 3.7 *Rise I* Score Frame and Note Properties Interface.

The development of a different ANG for every piece can be extremely time consuming. Hopefully, this work will lead to an ANG development kit that combines features already existing in separate environments. Some of these features may include:

- An algorithmic programming environment (such as JMSL).
- A transcriber that supports traditional notation as well as newer notational devices such as proportional notation schemes.
- A robust graphics environment so that the transcriber outputs a user-defined notation system.
- A synthesis engine for playback.
- A GUI development package.
- A means to bundle the generated software.

Such a multifaceted environment should allow composers to more easily create ANG's such as the ones for *nothing… I* and *sort 1*. They would provide more efficient feedback on compositional ideas and afford composers more time to generate pieces instead of writing computer code and engraving.

The ANG's presented in this section are critical to the compositions by allowing multiple realizations to be rendered. The interactivity takes place before a performance resulting in a fixed score (or several different fixed scores if the piece is rendered many times). The following section addresses an ANG that facilitates dynamic, mutable realizations of a piece by streaming notation in real time.

4. LiveScore: Real Time Generated Music Notation

4.1 Introduction

Real time algorithmic processes have been used in electro-acoustic music for some time now, and composers have a host of tools (notably Cycling '74's Max/MSP and the open source SuperCollider and PureData environments) available for realizing those processes in sound. The LiveScore project began by posing the question: could live generated music be performed by human musicians on acoustic instruments? This idea raised a host of other questions: How would musical information be conveyed to the musicians? Should the musicians be synchronized with each other, and how would that be accomplished? The resulting LiveScore piece provides a working example of live generated instrumental music. This section will outline the LiveScore system and touch briefly on findings from the first performances.

4.2 Technical Information

The LiveScore software consists of a client program for notation display and a separate server program that generates the musical material. The client is written in Objective-C and uses the Cocoa frameworks for Macintosh OS X. The server is written in the SuperCollider language. The client and server machines are networked wirelessly and communicate with each other using the Open Sound Control protocol developed at the University of California at Berkeley.

The client and server function together as a notation broadcast system. Music information is created on the server, orchestrated on the fly, and parts are displayed note by note on the performers' laptop screens. Each performer has her own client machine from which she reads the resulting notation.

4.3 Note Streams

A proportional notation system was chosen, in which horizontal space represents time. This approach is used for several reasons. First, it allows the full range of time to be notated. One of the unique attributes of the system is that it enables multiple musicians to read proportional notation in a coordinated fashion, which is something that is difficult or impossible to achieve with a static printed score. In addition, the proportional notation allows the server to treat notes as events in time without having to map them into meters with measures and beats. The client can display a stream of notes immediately as they are generated on the server. A standard 5-line staff is used, showing as many systems as will fit on the client screen. Each note is given a trailing beam that indicates the note's duration. For extremely short durations, the stem and beam are omitted. In addition to note information, the client can also display dynamics and arbitrary text instructions.

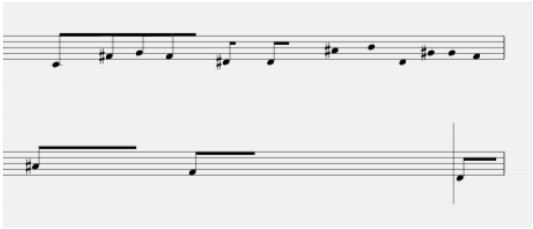

Figure 4.1 Fragment from LiveScore.

Any process may be used to generate the musical material. The real-time nature of the system allows for the incorporation of live data feeds such as environmental data, sensor input, or other kinds of controls.

4.4 Synchronization

Coordination between the musicians is accomplished by means of a conductor bar superimposed over the staff and displayed as a vertical line that moves from left to right, indicating the current time in relation to the part. Therefore, when the bar passes over a notehead, the performer is to start playing the indicated pitch and

sustain it until the conductor bar clears the end of the duration beam. The presence of the conductor bar creates a performance situation somewhat akin to certain pieces by John Cage in which musicians coordinate using stopwatches rather than watching a conductor or each other for cues. Musicians operate as independent entities, but their individual contributions are coordinated by the algorithm. This technique also enables a completely new kind of performance in which musicians are able to play complex rhythms together without the need for any sense of pulse or meter, though both may be implicitly present.

The conductor bar trails the notes by some amount of time specified for the piece, so the musicians can look ahead in the part and see what is coming up. The length of this lag determines the extent to which the process is "real-time." A short lag results in a performance that follows the note stream very closely and in practice, three to five seconds was adequate to enable the musicians to sight read their parts.

Figure 4.2 LiveScore at the Calarts Integrated Media Show, May 2006.

4.5 Performance Setting

The first performance of LiveScore was developed as an audience participatory performance at the Machine Project gallery in Los Angeles as part of the *You, Too, Can Play Difficult Music* series. The audience was invited to play with the knobs on a MIDI controller, while a quartet of musicians performed the resulting screen-based notation. Since this was a gallery and not a stage, the audience participants were able to walk around the space, view the notation, and "play" the MIDI knob controller. The situation created an informal atmosphere that encouraged exploration.

The actual musical content was generated by a simple stochastic algorithm whose bounds were determined by the knob positions. The server kept track of a few chance determined pitch collections, which were then selected at random and voiced for the ensemble, taking into account each instrument's range. The knobs were given whimsical labels, intended to encourage experimentation. These included "sparseness" – the amount of time allotted for a particular pitch set to sound, "pitchiness" – the size of the pitch collections, "stasis" – the number of available pitch collections, and "togetherness" – the size of the time window in which attacks could occur, and so on.

The resulting collaboration between audience, composer, machine, and musicians yielded some intriguing results. Although the process was simple, it afforded a rich variety of textures. Some of the participants who had a turn at the knobs were able to infuse the musical fabric with a sense of personal style. The performers remarked afterwards that that was a key point of interest in playing the piece.

4.6 Directions

The LiveScore system need not be limited to this one piece, and hopefully it will be applied to other generative processes and other performance goals. In addition, the scope of screen-based notation is potentially very large, and some of the concepts from LiveScore – the networked environment, the conductor function, the controller input – could be adapted to many kinds of notation and many styles of music.

5. Automatic Notation Generators: Conclusion

The preceding examples have shown various implementations of ANG's that use different input and incorporate different kinds of music notation including standard notation with MusicXML, unconventional composer-designed notations, and real time screen-based notation. We believe that they point to a use of notation software as an integral part of a composition. ANG's can drastically accelerate the iterative process of auditioning musical and notational ideas. They enable composers to create pieces with infinite possible realizations and foster the notion that creating music is an experimental process.

6. References

[1] Ablinger, Peter (2006) *Peter Ablinger – Quadraturen* URL: http://ablinger.mur.at/docu11.html

[2] Ames, Charles. "Thresholds of Confidence: An Analysis of Statistical Methods for Composition, Part 1: Theory." *Leonardo Music Journal*, Vol 5. (1995): 33-38.

[3] Ames, Charles. "Thresholds of Confidence: An Analysis of Statistical Methods for Composition, Part 2: Applications." *Leonardo Music Journal*, Vol 6. (1996): 21-26.

[4] Barrett, G. Douglas. (2006) *Spectore*. URL: http://synthia.caset.buffalo.edu/~gbarrett/spectore.pdf

[5] Fitz, K., Haken, L., Lefvert, S., Champion and O'Donnell, M. (2003). "Cell-utes and Flutter-Tongued Cats: Sound Morphing Using Loris and the Reassigned Bandwidth-Enhanced Model." *Computer Music Journal* 27(4): 44-65.

[6] JMSL and JScore. Nick Didkovsky (1997 – 2007). URL: http://www.algomusic.com/jmsl

[7] Polansky, Larry. "Morphological Metrics." *Journal for New Music Research*, Vol. 25 (1996): 289-368.

[8] Recordare, 2003, *MusicXML Definition*, Recordare LLC, California, USA. URL: http://www.musicxml.com/about.html

[9] Schafer, R. Murray. *The Tuning of the World*. New York: Alfred A. Knopf, 1977.

[10] Tenney, James. *Meta + Hodos*. Lebanon, NH: Frog Peak Music, 1961.

[11] Winter, Michael. *nothing… I*, *sort 1*, and all referenced work (2006). URL: http://www.sonicism.net/selected_works.html

[12] Xenakis, Iannis. *Formalized Music*. Hillsdale, NY: Pendragon Press, 1971.