Current Research in Computer-Generated Music

The six overviews that follow reflect varied ongoing research. Reporting from such diverse locales as Singapore, Europe, and the US, the authors explore the spheres of computer-aided composition, synthesis of musical scores, computer simulation, and composing by musical analog.

Algorithms for Musical Composition: A Question of Granularity

Stephen W. Smoliar
Institute of Systems Science, National University of Singapore
Heng Mui Keng Terrace, Kent Ridge, Singapore 0511

Computer programs that "compose" music have been around almost as long as the computers that run them. Initial approaches selected notes for a composition using a random process with constraints to eliminate or modify undesirable choices. The computer composition "Iliac Suite for String Quartet" by Lejaren Hiller and Leonard Isaacson, demonstrates the range of these approaches.

Experiments with Markov processes were probably the most appealing and the most frustrating applications of computer composition in the "Iliac Suite." The appeal lay in the belief that knowledge of past events could serve as a valuable predictor for subsequent events. The frustration lay in the discovery that there was either too much or too little of this knowledge. If the number of past events considered was too small, the result sounded as random as if the notes had simply been selected from an arbitrary set of weighted probabilities without any prior knowledge. If the number was too large, what was "predicted" was nothing more than a replication of the original data from which the probabilities were computed. There seemed to be no middle ground between the absolute predictability of duplication and the absolute unpredictability of random noise.

The quest for an appropriate middle ground led to an investigation of new mathematical models as alternatives to Markov processes. The spectral densities of a wide variety of physical quantities tend to vary as the inverse of the frequency. \(1/f\). Voss and Clarke demonstrated such a correlation using the audio signal of Bach's first Brandenburg Concerto and then detected the same correlation in signals from classical music radio stations.\(^1\) Voss and Clarke conducted similar studies of the music of other composers, including Scott Joplin and Milton Babbitt. Observing this distribution in many different sources of music, they reasoned that they could synthesize new music by a random process based on the same distribution. Figure 1 shows an example.

Eric Iverson has explored another approach.\(^2\) He applies principles of autocatalysis, a chemical process that enables the reproduction of molecular strings, to the synthesis of "strings" of
music—either pitches or intervals. Figure 2 shows an example of his results.

The variety in approaches to algorithmic composition is impressive, but the results are disappointing. Do these results arise from a critical commonality in these many experiments? I believe the commonality is the assumption that musical composition requires decisions about the placement of notes.

This assumption is natural enough. It has its roots in a pedagogic tradition in which the composition student is concerned with manipulating notes on music paper. However, music need not be notated to be composed. A composer can compose while playing an instrument. This is generally known as improvisation. Anyone who has heard good jazz improvisation can appreciate how unlikely it is that musicians who improvise read from some notation they write in their head. Thus, if we want a foundation for algorithmic composition, the materials provided by the pedagogy of musical composition may be far less valuable than those of the actual practice of making music. In this article I argue that making music is concerned with a higher level of granularity than that of the notes on music paper.

Raising the level of granularity. Musical composition is not necessarily a matter of putting notes together in the right way. Long before there was jazz, people were making music by working with a higher level of granularity. Mozart's Dice Composer, for example, generated (to use the terminology of computational linguistics) random sentences by selecting productions from a simple context-free grammar. Each of the terminal symbols in this case was an entire measure of keyboard music. Each measure of the score, in turn, corresponded to 11 productions, each of which filled in that measure with one of those terminals. The bulk of Mozart's work with the Dice Composer was to make sure that the terminals chosen for any given measure were interchangeable. Thus, by raising the level of granularity to entire measures, Mozart could use chance techniques to produce acceptable music in the classical genre in a manner never duplicated by any algorithmic technique concerned with selecting individual notes.

If raising the level of granularity increases the power of random processes applied to musical composition, perhaps we are thinking about the wrong things when we focus our attention on the selection of individual notes. Work in artificial intelligence shows that such low level decisions may actually be subordinate to a model-based control structure. The underlying premise of such control is that the knowledge sources used to make decisions include actual examples of how problems have been resolved: These examples are models. In musical composition, such models include specific compositions or excerpts from compositions. David Cope is a good example of a working composer who has managed to put a model-based approach into practice. (See Cope's article beginning on page 22 of this issue.)

An alternative agenda. When we talk informally about musical composition, we rarely talk about individual notes. Composition is a matter of working with "musical ideas." We use this phrase intuitively, without pinning down just what it denotes. Nevertheless, following the lead of model-based reasoning, we can assume that musical ideas include our memories of past listening experiences. To some extent all composers pick up materials from past experiences and find new ways to assemble them. What do such intuitions tell us about
algorithmic composition? The primary lesson is that our agenda for algorithmic composition is off the mark. If we are determined to seek out algorithmic rules, then our search should be directed by two questions:

- How do we identify units of material of the appropriate granularity?
- Given a collection of those units, how do we properly assemble them?

What made Mozart's Dice Composer a clever piece of work was his recognition that he could think about simple dances (following the style of the dance movements of Bach's keyboard suites) in terms of individual measures. He could then ask how any one measure could be varied so that it would always "fit" in the context of preceding and succeeding measures. He gave these variations enough flexibility that any alternative for the first measure could be followed by any alternative for the second measure, and so on for the duration of the entire composition.

Perhaps Mozart's exercise was a parody of the rather routine approach to composition he observed in others around him. For Mozart, the model became music only after he suitably perturbed it, but he could not address questions of perturbation until he had a model in place. The minuet movements that Mozart wrote for his more memorable orchestral and chamber music could not readily conform to this dice model, since the model had no good way of implementing material that returns in a similar, but not identical, form.

The ordinary versus the extraordinary. The moral of the Dice Composer story is that identifying units of appropriate granularity and assembling them properly is no easy matter. The basis for "automating" dice-composed binary forms was Mozart's recognition that it was a highly routine activity. However, many who pursue algorithmic composition do so out of "genius envy." They are more interested in the extraordinary than the ordinary, and fail to recognize that they cannot have the extraordinary without first having the ordinary as a point of departure. Any music powerful enough to compel us to sit still and listen — even one of Bach's two-part inventions or Domenico Scarlatti's many sonatas — is probably too sophisticated for the analysis required in algorithmic composition. Finding much past evidence of the ordinary in these extraordinary pieces may be difficult simply because the ordinary rarely survives the eroding forces of history.

However, the ordinary is very much with us in the present. We tend to ignore the routine work of composers obliged to provide themes and jingles for commercial applications. These composers must create on demand, probably to the extent that none of their tasks receive extensive cognitive effort. Here is where we may find artisans with a clear understanding of the units of material with which they work. We should look where the ordinary is practiced to find a foundation for studies of the extraordinary.

References


Stephen W. Smolar is investigating several aspects of music cognition and perception in addition to his artificial intelligence research at the National University of Singapore.

Scoresynth: A System for the Synthesis of Music Scores Based on Petri Nets and a Music Algebra

Goffredo Haus and Alberto Sametti
Laboratorio di Informatica Musicale, Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano, via Moretto da Brescia, 9, I-20133 Milano, Italy

Music structures can be described, processed, and synthesized using a more flexible kind of representation than the staff. In this article, we show how. In fact, we have chosen and organized symbols depending on instrumental needs within common music notations. The level of representation within scores is more detailed than that of music composition and more abstract than that of timbre modeling.

The new kind of representation we propose makes up a conceptual music framework with as many different levels of abstraction as the musician needs. It allows us to explicitly describe and process what we call music objects (both traditional and nontraditional music objects).

A music object means anything that could have a musical meaning and that we think as an entity — either simple or complex, abstract or detailed — with a name and some relationship with other music objects. We can describe music objects at various abstraction levels within a hierarchical context — for example, the structural level, the score level, and the timbre level.

Common music notation is characterized by many different languages (one or more languages for each level of representation). Our research is devoted to the definition of just one language, suitable for every level of representation.
To identify the most suitable description tool, we sought a formal tool that
- requires few symbols,
- has a graphical form of notation,
- allows hierarchical description,
- allows the description of music objects processing,
- allows time description,
- allows deterministic/nondeterministic models,
- allows macro definitions for common structures, and
- shows the musician the score synthesized as the model executes.

Score modeling. We have been using Petri nets (PNs) as the basic tool for music description and processing since 1980. In our research, we have developed certain programs to edit and execute PNs for both music and general-purpose applications. The most recent program we developed is called Scoresynth. In it, we use PNs to describe, process, and synthesize music scores.

While we assign music objects (MOs) to place nodes for describing music information, we assign causal relationships and transformation rules to PN structures and PN parameters (marking, numeric labels, algorithms, etc.).

MOs can be processed by modifying the superimposition and juxtaposition laws within PN structure. On the other hand, PN parameters allow us to create instances of MOs that modify themselves according to the behavior of PN models during execution. Furthermore, we can represent the same musical information by PNs at a lower or higher level of abstraction by using suitable alternative modeling approaches.

Indeed, a PN model with a particular initial state (that is, initial marking) can represent a family of scores; a particular execution of the model synthesizes a specific score. By modifying the initial marking of the model, we can change the family of scores that can be produced by model executions. A special situation exists when we have a fully deterministic model. In such cases, we can produce one only score (given a particular initial marking).

The Scoresynth system is made up of the editor and the executer of the models.

The editor allows the description of hierarchical, timed, concurrent, deterministic/nondeterministic, and weighted PN models. Both places and transitions are used as the "morphism" nodes to implement hierarchy. Common PN structures can be described as macros. An interactive graphic user interface is provided to build PNs by an iconic representation.

The executer performs the models, synthesizes MIDI (Musical Instrument Digital Interface) scores, and shows a graphic map of the MOs as they are generated by the model execution (see Figure 1). Looking at the map, the musician can interact with the model, stop the execution as soon as a bug is found in the model, and proceed to the editor environment to make modifications. This can be done simply by executing the model and looking at a graphic window that shows — on a time-MIDI_channel plane — the map of MOs as they are appended in the score and any information requested by the musician about the genesis of MOs.

Music objects. As stated previously, the meaning of MOs is strictly related to place nodes. To us, an MO encompasses not only any sequence of notes but also something more general that is not exclusively linked to listening and not connected to the idea of process because it could last zero seconds. Obviously, the definition of an MO is partly affected by the music code standard we use (MIDI) and must be kept within the bounds of this standard (see Figure 2 as an MO example). All MIDI messages, even if not directly concerned with a note, can be put in an MO.

MOs can be coded according to three syntaxes:
- a coding language that is specific in Scoresynth,
an alphanumeric MIDI coding language, and
a coding format that implements the Standard MIDI Files 1.0 format (both type 0 and type 1).

The program uses the first format internally. The second format is based on the availability of editing tools (such as common word processors). The third format has been implemented for file import/export. MOs can be defined either by a common text editor or by a MIDI controller (a keyboard, guitar, microphone, etc.). The MOs can be stored in separate files (one file for each object) or within the PN file to which the associated place belongs.

Figure 3a shows the simplest net to perform an MO. We joined the Theme place, which has been previously stored in a file according to any of the three allowed formats, to the Theme MO.

The upper attribute of the place defines the number of tokens available, while the lower attribute specifies the maximum capacity of tokens allowed for that place.

The music algebra. The main idea of Scoresynth is to make it possible to transform both the parameters of sound (pitch, timbre, intensity, and duration) and the order in which notes appear, using algorithms. Algorithms join with transitions. If a transition has an associated algorithm, Scoresynth merges all the MOs in the entrance before making the transition fire. When the transition fires, it takes a token from each input place, applies the transformation to the single MO obtained, puts the token into all the output places, and places the outcoming MO into every output place allowed for MOs. The transformation can only be applied to note events. All the other MIDI commands are filtered and lost.

If Theme stands for the MO on which the algorithm is processed at the input, the algorithm can be applied to the whole Theme or can be limited to one of its subsequences, as designated by the musician. It is also possible to use information from every MO available at that moment within the whole model.

An algorithm can be formed by one or more single operators. Each operator

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alg1:</td>
<td>C: 1, S, [Theme, 1], 1 [...all on MIDI channel 1; use &quot;Theme&quot; MO] M:1, $, 2 [...repeat 2 times]</td>
</tr>
<tr>
<td>Alg2:</td>
<td>C: 1, S, [Theme, 1], 2 [...all on MIDI channel 2; use &quot;Theme&quot; MO] M:1, $, 2 [...repeat 2 times] P[C]: 1, $, ? + 2 [...2 degrees transposition, C major tonality]</td>
</tr>
<tr>
<td>Alg3:</td>
<td>C: 1, S, [Theme, 1], 3 [...all on MIDI channel 3; use &quot;Theme&quot; MO] D: 1, $, ? / 2 [...halve durations] M:1, $, 4 [...repeat 4 times]</td>
</tr>
<tr>
<td>Alg4:</td>
<td>C: 1, S, [Theme, 1], 4 [...all on MIDI channel 4; use &quot;Theme&quot; MO] I: 1, S [...retrograde] M:1, $, 2 [...repeat 2 times]</td>
</tr>
</tbody>
</table>

Figure 6. The parameter list for invoking the Voices macro Petri net.
can affect either the parameters of the notes or their order. Each operator is applied to the outcoming MO from the operator before. Set in its header, the algorithm is applied to the range of notes as each note occurs. The header also defines the kind of parameter to which the operator expression must be applied—pitch, duration, intensity (key velocity), MIDI channel—and the order of the affected notes. Let us examine two examples of such algorithms.

(1) **Specular inversion.** This can be done by means of a reference note, called the mirror note; that is, every processed value of pitch is symmetrical to the mirror note. The following is an operator to invert the pitches with respect to C-sharp 3:

\[ P: 1, $, [\text{Theme}, 1], 2 \times \text{C}\#3 - ? \]

where P stands for pitch; 1, $ stands for "from the first to the last note of the input MO"; [Theme, 1] determines the use of information from the Theme MO (not from the MOs joined to the input places, which are the defaults and need no explicit specification) starting from the first note (1); and 2 * C#3 - ? is the expression that calculates the new pitch values. The ? is a metacharacter that represents the value of the parameter (in this case, the pitch), assigned to the current processed note.

Other algorithm headers affecting note parameters are:

- **C (channel)** = operator on MIDI channels,
- **D (duration)** = operator on duration, and
- **L (loudness)** = operator on intensity (key velocity).

(2) **Inversion of the order (retrogradation).** The inversion operator rewrites the input MOs backwards: I: 1, $, where I stands for invert, and 1, $ stands for "from the first to the last note."

Other algorithm headers affecting the order of notes are:

- **K (kill)** = operator for deleting notes,
- **S (save)** = operator for preserving notes,
- **M (multiply)** = operator for multiplying notes, and
- **R (rotate)** = operator for rotating notes.

Figure 3b shows a very simple net that can transform MOs. Here, we can join any algorithm we wish to the transition Alg, decide if we want to enable the execution of the Theme and/or Theme.mod MOs (setting the specific place attributes), and decide if the MO obtained from the transformation (Theme.mod) should be volatile or stored in a file. In this case, a specular inversion algorithm may be P: 1, $, 2 * C-sharp 3 - ?

**Firing and timing.** PNs are particularly suitable for describing concurrent processes and controlling their synchronization (or lack thereof). The behavior of the net implicitly determines timings. We can synchronize MOs simply by suitably structuring the node connections. The structure of the net determines the firing sequence.

The firing sequence changes with different executions of the net, so there is no correspondence between a net and its firing sequence. We can have many transitions qualified for firing at the same time and not know which one fires first. The net does not describe this kind of information. Every execution of a PN model may give a different firing sequence.

Our approach represents timed MOs by places and their transformations by transitions. The firing of a transition has a null duration. In this way, when a token is put into a place with an associated MO, the token cannot be considered for the firing of transitions connected to the place until the associated MO has ended.

**Hierarchy and macros.** In this section, we introduce two new concepts: hierarchy (refinement) and macro.

A refinement, called a subnet, is a PN that gives a more detailed description of a node (either a place or a transition) of the upper level of abstraction. If we choose to define a subnet associated with a place P, then the daughter net must have two special places: an input place In and an output place Out. Input arcs of P become input arcs of In, and output arcs of P become output arcs of Out. Transitions can be refined in the same way. We can also model recursive nets (that is, nets that contain themselves as refinement nodes). To avoid endless recursive expansions, we must assign a recursion level number to every recursive net.

Furthermore, when we design a net model, we often have to create many similar nets that differ in their place attributes and algorithms but are identical in structure. With a macro net, we can use just one net as the base model and then modify its attributes and/or algorithms as needed. This enables the musician to create personal libraries containing commonly used nets. We can do this by writing a modifier list for every subnet place or subnet transition we want for the net. This modifier list is loaded into memory and actualized before execution of the model.

Figures 4a, 4b, and 5a represent a deterministic model based on the "Are You Sleeping" theme, while Figures 4a, 4b, and 5b represent a nondeterministic version of the same melody. (The alternative depends on Figures 5a and 5b.) Figures 4a and 4b are deterministic if they are considered stand-alone nets.)

The IEEE net is the most abstract net. The Are You Sleeping place is then refined by the homonymous net with Input as input place and Voices as output place. The input arc to the Counter place has weight 3; this means that the firing of the transition will put three tokens in the Counter place. The Rest 4/4 place has a 4/4 rest as an associated MO.

To expand the Voices place with a macro net, we can use one of the two nets in Figure 5. In the first case, we obtain a deterministic execution of the model; in the second case, we obtain a nondeterministic one. Stated another way, we cannot fix the firing sequence, but the firing sequence is determined by Scoresynth with a pseudorandom uniform distribution of the order.

The Input place has no token as initial marking because it will receive tokens from the upper level of the model (in both the Are You Sleeping and Voices nets). Figure 6 shows the parameter list for invoking the Voices macro PN.
(note that we invoke only the actualization of algorithms associated with the four transitions of the macro net here, but we can do the same with respect to all the possible attributes of the PN).

Figure 1 shows the map of the MOs obtained from the IEEE model execution. It contains four different instances of the Theme mod MO belonging to the Voices net put on different MIDI channels. These MOs differ from one another because they are generated by different algorithms. The other MOs of the model are not present in the map (or the output score) because Theme was set as an MO that is not meant to be played and Rest 4/4 produces a delay, since it is a rest MO and not a MIDI event.

Transforming music structures. Our research has shown that music structures can be transformed simply by

• modifying markings, MIDI channel specifications, and capacities of places;
• modifying labels of arcs;
• modifying MOs associated with places;
• modifying algorithms associated with transitions;
• modifying the structure of PNs; and
• executing nondeterministic PN models many times.

While editing PN structures and parameters affects causal/structural relationships within the architecture of scores, editing both MOs and algorithms affects the basic information units and their transformations. We believe that the Scoresynth system provides a powerful means for describing and processing music in a way that is closer to music thinking and perception than common music notation is.

Acknowledgments

We thank Enrico Bianchi, Antonio Camurri, Giovanni Degli Antoni, Giampiero Jacomini, Antonio Rodriguez, and Renato Zaccharia for their contributions to previous research projects and experience related to Petri nets and music. We particularly thank Sergio Scolaro for his relevant contribution to the design and development of some features of Scoresynth Rel. 2.0 (especially the graphic interface and PN macros). Moreover, we especially thank members of the staff of the Laboratorio di Informatica Musicale at the University of Milan for the scientific and technical support they provided during our research.

This research is partially supported by the Italian National Research Council through the Music Topic LRC C4, Grants No. 89.00031.69 and 90.00678.PF69: “Intelligent Music Workstation.” Goal C: Sistemi Avanzati di Produttività Individuale; Subproject 7. Sistemi di Supporto al Lavoro Intellittuale, Finalized Project: Sistemi Informativi e Calcolo Parallelo.”

References


Goffredo Haus, scientific director of the Laboratorio di Informatica Musicale at the University of Milan, does research in the university’s Department of Computer Science and composes music by computer methods at both the score and sound level. He helped found the Italian Association of Musical Informatics.

Alberto Sametti is the software manager of the Intelligent Music Workstation project at the Laboratorio di Informatica Musicale at the University of Milan. He works on Petri net applications to music and is a scientific consultant to Apple Computer on software development and validation.

Neurswing: An Intelligent Workbench for the Investigation of Swing in Jazz*

Denis L. Buggi
Via Casagrande 12
CH 6932 Breganzona, Switzerland

Neurswing is an intelligent system to investigate swing in jazz by simulating the operation of a rhythm section. From an input consisting of the harmonic grid of a standard tune, it constructs a network representing musical data. At runtime the system generates and plays the music of piano, bass, and drums in real time.

The user can control the performing style of the rhythm section by turning "knobs," that is, by setting input parameters of a second, separate and asynchronous, net, which manipulates the probability of some choices. The system can simulate most aspects of a performing jazz rhythm section. It

*This article is an abridged version of previous publications that describe the work performed at the International Computer Science Institute in Berkeley, California, whose sponsorship is gratefully acknowledged.
Figure 1. (a) Structural harmonic grid of Rhythm Changes, the basis of many pieces favored by jazzmen: “Anthropology,” “Dexterity,” and “I Got Rhythm.” (b) The harmonic net constructed from the grid. From top to bottom: measure and beat number, piano note units, bass note units, drum note units, units of the given harmony (squares), and units of harmonic substitutions.

substitutes chords, bass lines, and drum licks and has been used by practicing improvisers as a didactic tool. Furthermore, the rules for substitution as well as the stylistic net are external to the basic system and can be configured and altered at will, allowing the user to treat the system as a workbench for experiments in the synthesis and analysis of swing.

The problem. In jazz the word swing has many meanings. It really means a holistic, “gestaltic” quality of almost extra-musical nature, by virtue of which the jazz message is passed to the listener. Swing is the medium for jazz, as space is the medium for sculpting. This sets jazz apart from classical music, for example, where the piece has a quality independent of the moment in which it is played. In jazz the opposite is true. This is why jazz is improvised by definition. Swing cannot be annotated in a music score; rather, it is captured in a record, which has been the document in jazz since 1917, the accepted date of the first jazz record.

Swing is produced by — though it should not be confused with — certain technical means such as rhythmic patterns, harmonic progressions, melodic accents, and instrument timbres. One of its aspects is possibly the opposition between an implied regularity of patterns, rhythmic or otherwise, and a consistent yet controlled violation of that regularity. This project addresses some of these quantifiable, technical aspects of swing.

The system and its possible uses. Neuriswing attempts to simulate a jazz rhythm section and allows the user to modify its parameters at three levels. At the top are three stylistic knobs:

- **hot-cool** controls the overall impression of drive and push, as defined from the thirties to the fifties, with the “hot” elements of swing in opposition to the “cool” school of the early fifties;

- **dissonant-consonant** controls the overall level of dissonance in the underlying harmony;

- **as-is-ness** amplifies or reduces the level of free substitution versus adherence to the given input — the piece — and is thus a control for improvisation.

At the second level, all rules for harmo-
Figure 2. The architecture of the system: stylistic net above, harmonic net below. The inputs of the stylistic net produce outputs — the hooks — indicated by asterisks and corresponding to those of Figure 3. These outputs affect the probability of possible choices in the harmonic net at runtime as well as tempo increase or decrease and delays of piano and bass in respect to drums.
ny substitution, piano, and drum patterns are in external tables and can be modified at will. At the third level, the presented stylistic network is just an example — an untrained neural net — to demonstrate how style can be modeled. It is external to the system, and some other network could substitute. In this sense, Neurswing is a workbench because it allows us to investigate which parameters have what influence on swing.

Aside from its scientific interest, Neurswing has been used by practicing musicians as a didactic tool, since for most purposes it can substitute for existing records with jazz rhythm sections (for example, Music Minus One and Japanese Karaoke), such records being inflexible and of fixed duration, tempo, and key. Neurswing, on the other hand, is a flexible, programmable system that can play any tune in any key and at any tempo for as long as desired. It can even repeat difficult passages. Neurswing can help aspiring musicians control rhythmic accents and correct typical beginner's errors, such as slowing in difficult passages and accelerating in easy ones. Control of tempo is an essential technique in swing; a metronome may offer some help, but Neurswing is close to the ultimate practicing tool, a real rhythm section embodying harmony, rhythm, and style.

The input and the harmonic net. The harmonic grid (see Figure 1a on page 61) is the input to the system. Each square of the grid corresponds to a measure, or four beats, and contains symbols representing chords that determine the harmonic structure, though not the actual notes. The meaning of these symbols, which are also used in guitar charts, is described in the jazz literature. This section of the system uses a knowledge base to construct a connectionist model of the jazz performance, drawn by the Rochester Connectionist Simulator and shown in Figure 1b. The net units represented by a square contain data of the grid harmony, while the units below show the constructed alternative harmonies. The symbols above the squares represent drum hits, and above these are the units of the bass notes, initialized from the given harmony. The four to seven units just below the measure and beat numbers at the top contain the initial notes of the piano units. The paths in the section representing harmony and the contents of the units of piano, bass, and drum notes change at runtime under control of the stylistic net. The harmonic net operates in synchrony with the beat; activation passes from all units on a beat to those on the next beat and then wraps around.

The system architecture. Figure 2 shows the system architecture. During operation of the underlying harmonic net, there are several points where it is possible to choose among different possibilities — for instance:

- kind of harmony substitution;
- piano rhythmic pattern;
- type of walking bass — up, down, or note from the chord;
- position of piano chord;
- drum pattern; and
- piano or bass appoggiatura.

All these choices, called hooks, are under control of the stylistic net, shown at the top; Figure 3 shows what they are and how they relate.
The system could operate without the stylistic net. This would be equivalent to a setting of 0.5 for all three knobs, and the system would not exhibit any stylistic consistency. Turning a knob activates the stylistic net, which in this example consists of three inputs, 360 units in the middle layer, and 360 outputs. The outputs, or hooks (indicated in Figures 2 and 3 by asterisks), are numbers that change the probability of a given choice. For instance, an increase in the hot/cool knob favors a dense rhythmic piano pattern, more piano appoggiaturas, louder playing of the piano, walking bass up, more bass appoggiaturas, more nervous drum patterns, more drum fill-ins, and more crash cymbal hits. An increase in the dissonance knob favors dissonant harmonic patterns, flatted fifth in dominant seventh chord, inversion for piano chords, and half steps for piano and bass appoggiaturas.

The advantages of modeling stylistic constraints with a neural net are twofold. First, complicated interactions between related aspects, or parameters, can be easily modeled, modified, and tested without changing anything in the controlled program. Second, hierarchies of nets can be conceived to further constrain some aspects of style without affecting the operation of underlying nets.

**Performance.** Neurswing is a hybrid system using a knowledge base to construct a network that contains data for jazz improvisation and performance. It realizes controllable real-time jazz improvisation using neural nets.

Neurswing performs acceptably for tempos between 100 and 250 beats per minute (medium to fast) and optimally for 180 to 700 beats per minute. The time signature is 4/4. Extension to other signatures such as 3/4, 6/8, 5/4, and 6/4 is possible, but has not been realized.

While there are commercial systems that simulate a jazz rhythm section, none are capable of improvising, that is, of elaborating the input with substitutions, changes, and drum accompaniment; they merely replay what the user typed in. Hence, they are not interactive and do not allow control during the performance, as the knobs in Neurswing do.

Computer music projects have rarely been applied to jazz. Perhaps this work will stimulate further investigation into the subtle problems of jazz improvisation and swing.

### References


Denis L. Baggi, a computer scientist and a musician, is guest editor of this special issue. His biography and photo appear on page 9.

---

**HARP: A System for Intelligent Composer's Assistance**

*Antonio Camurri, Corrado Canepa, Marcello Frixione, and Renato Zaccaria*

**DIST — Dept. of Communication, Computer, and System Sciences, University of Genoa, Via Opera Pia 11/A, I-16145 Genoa, Italy.**

Systems for computer-aided composition are generally based on tools and languages designed for low-level manipulation of music scores and composition algorithms. In this article, we introduce HARP (Hybrid Action Representation and Planning), a prototype high-level system for computer-aided composition.

We think of the system as providing intelligent composer assistance. HARP can store and process music and sounds, and carry out plans for manipulating this material according to composers' goals. The system can generate new pieces of music as well as manipulate existent ones on the basis of composer-supplied material. The system also provides capabilities for formal analysis of both music and sounds, so it can extract information useful in subsequent synthesis processes.

HARP represents and manipulates music knowledge using a twofold formalism. An object-oriented concurrent environment, the analogical subsystem, provides procedures to manage the sound itself (samples, codes, and algorithms) and particular analysis processes. The symbolic subsystem, a declarative symbolic environment, stores higher level scores, composition rules, definitions in general, and descriptions of pieces of music. The symbolic subsystem is based on a multiple-inheritance semantic network formalism derived from KL-One. There is a formal link between the two subsystems. For example, if a composer asks the semantic network to generate a particular instance of a music object, the system automatically "fires" the appropriate...
procedures at the analogical level.

Figure 1 shows part of the HARP symbolic knowledge base: Ellipses correspond to concepts in the taxonomy, double arrows are "is-a" links, and arcs with small boxes correspond to roles (slots, relations between pairs of concepts). For example, a music action is-a music fact, and one of its roles is intensity, whose value is amplitude. In other words, the intensity of a music action is an amplitude, as shown in Figure 1a.

**The HARP knowledge base.** We defined two basic music entities in the system: the music action and the compositional action (see Figure 1). Music actions can represent musical material at different levels of abstraction, from the sonological level (low-level descriptions of sounds) to the whole musical form of a piece (for example, fugue or sonata).

Compositional actions are meta-actions; that is, they are the "manipulators" that operate on music actions (both classes and instances) to produce new music actions or to perform analysis tasks useful for subsequent manipulation. Compositional actions let composers manipulate music actions according to their objectives. For example, a composer can operate on a music action at its sonological level for the "tuning" of particular sound features. While planning the overall structure of a part, the composer can see the same music action as a more abstract, high-level symbolic entity and work on its possible relations with other music actions.

The composer can introduce into the system subclasses of music actions and instances. Subclasses are reusable objects, skeletons of music scores, or sound definitions. Instances correspond to complete, individual objects that can be heard reproduced by the HARP sound output channels.

HARP's framework does not refer to a particular musical style or context. Each context requires the creation of suitable definitions in the symbolic part and the related analogical procedures. For example, in the context of Western tonal music, a composer can define the subclass canon with its structure: its component music actions antecedent and consequent, as shown in Figure 1b. Successively, the composer can introduce — or ask the system to generate - a particular canon characterized by a given antecedent and a suitable consequent. Basic features of music actions include

- a temporal connotation (the begin and end roles), allowing the com-
poser to relate music actions to each other in the time domain, and
• a set of relations, such as dynamic evolution, timbral and density content, pitch, metrics, and rhythmic properties.

Assume that a composer wants to specify in a compositional action that a music action (a part of a piece) \( N \) be generated in relation to, say, the evolution of another music action \( M \), expressed simply as a score fragment. An analogical representation of an instance of \( M \) is formed by a score fragment (hooked to the concept \( M \)) and by a set of functions, or methods, hooked to its relations, which extract the related feature from the score. The composer can query the intensity of \( M \), which corresponds to a call to the intensity method of \( M \), whose result is a presentation of the behavior of the dynamics of \( M \), extracted from its score fragment. In this way, the composer carries out a reasoning process, partially in procedural form. No a priori assumption has been superimposed on the leading features of music actions. In classical Western music, the melodic aspect could predominate in the manipulation of music actions, but the system lets the composer modify or redefine their relations and “manipulators” according to his or her needs.

A compositional action is characterized by a procedural description of its behavior and a temporal connotation (with a time axis defined by the \( \text{mbox{msbeg}\hbox{in}} \) and \( \text{mbox{mend}} \) temporal roles, different from the time axis of music actions). As with any other object defined in the system, composers can define both subclasses and instances of compositional actions. Composers can define compound compositional actions in the system as sets of given compositional actions whose execution is ruled by proper constraints on their temporal relations.

**System interface.** HARP lets composers use different visual interfaces to communicate with the system. It provides a visual presentation of the semantic network formalism. Composers interact visually or textually with this part of the system by introducing or editing elements in the net, and specifying to the system queries whose results can be graphically presented. Composers can make queries on classes and subclasses, the definitional part of the network, and on instances, that is, individual music entities.

The system shows subsets of the musical material, allowing different views of the same material. In Figure 1, for example, the query “show all objects from which canon inherits properties” results in a display that colors some network elements gray to give prominence to a path in the taxonomy. Another simple query, “show all about the object music action except temporal relations,” results in a display of the concept music action with all its relations except the begin and end inherited from music fact (see Figure 1a).

**Reasoning mechanisms.** The symbolic subsystem is equivalent to a subset of first-order logic in an artificial intelligence system. The system answers queries using the symbolic subsystem formal deductive apparatus. We have implemented a classifier to organize concepts in the taxonomy.

Music actions have “hooks” to score fragments, and compositional actions have hooks to code “chunks.” This underlying level of procedural definitions constitutes the analogical level of HARP, which is structured into a *sonological component* and a *simulative component*. The simulative component is formed by a set of procedures that acts as a counterpart to the symbolic deductive apparatus and integrates the inference capabilities of the symbolic system. Both the simulative and sonological components are driven by the symbolic level subsystem, because proper activations of a node in the net correspond to calls to its hooked code.

The system also provides mechanisms for visual interaction with the analogical level of music information representation. The mechanism is based on music notation, sound graphic descriptions, and visual metaphors. Here, queries correspond to the activation of code modules, with the passing of proper parameter values. An example is the query on the intensity of the music action \( M \) presented in the previous section. Here is another example: A canon is generated by an imitation process (Figure 1b). A composer can define a particular canon subclass, say *per diminutionem* canon, as generated by a subclass of imitation, say *per diminutionem* imitation. The composer can hook more detailed procedures to the second concept, and one of them can implement a *proper diminution* algorithm. By querying a proper consequent for this canon, the composer can generate an instance of *per diminutionem* canon that starts from the above concepts and an instance of antecedent. The system answers by activating the proper procedure hooked to *per diminutionem* imitation, passing as parameters the antecedent instance. (We produced examples 4 and 5 in the available audio recording by introducing particular canon definitions using suitable subconcepts of canon and imitation. See the “Audio Examples” box on page 54 for information on how to obtain the recording.)

**Force fields — A recorded example.** An important set of analogical descriptions in HARP is based on the metaphor of force fields. These descriptions let the composer think of and perform a set of compositional actions in terms of the intuitive natural dynamics of navigation in attractor fields, instead of using a rule base.

Force fields play a significant role in the generation of the examples on the available recording. The sixth music example is generated starting from a phrase (the incipit), a definition of the sound parameters, and two concurrent force fields operating as follows: The initial incipit (easily recognizable as the beginning phrase in the example) is imitated with continuous timbral variations. At the same time, the aspect of pitch intervals progressively degrades, giving place to the timbral aspect. In HARP terms, there is a force field operating on the timbral space, which performs a continuous timbral disassembling according to its field function. Concurrently, another force field, operating on the pitch axis, gradually superimposes a variation of the initial incipit. The result is that the application...
of this force field (which has a function with local minima at particular pitches) to the incipit causes a gradual pitch stretching, which conduces to the final melodic contour.

The metaphor of force fields is useful for describing continuous changes in music actions. A composer can use several force fields to easily model complex behaviors (such as continuous timbral evolutions), which are very difficult to model as rules. Force fields also give a different viewpoint on music entities and provide powerful manipulation primitives.

With HARP, composers can formulate such complex analogical queries, involving complex activations of interacting procedures, and build high-level music actions.

The other examples. The recording includes two groups of musical examples: a simple improvisation environment in an Afro-American jazz style, and three excerpts from "Anceps Imago," a contemporary piece for two harpsichords and computer-generated music (composed by Corrado Canepa in 1989).

The first three musical examples are simple improvisation fragments. The system's goals were to generate and transform suitable musical phrases (the soloist improvisation) to follow the composer's defined harmonic schemas in an Afro-American jazz style (blues, ballad, and so on). We stored a set of predefined music phrases in the system in terms of assertional constants, or subconcepts of music action. Compositional actions take as inputs subsets of these phrases for developing improvisations.

Examples 4 and 5 are two excerpts from the first part of "Anceps Imago," in a canon form. Example 6 is described above. For the recording, we replaced the two harpsichords with computer music of a similar timbre. We used cmusic as the low-level language to describe the computer-generated scores.

The HARP software prototype is implemented in Prolog and C, running on Apple Macintosh and 386-based machines under Microsoft Windows. HARP has several other features, including the ability to merge different knowledge bases while maintaining global consistency and detecting conflicts. We will implement other knowledge bases in the near future. Problems in the formalism, such as how to store consistent multiple views of music entities, are areas for future research.

Acknowledgments

We thank Salvatore Gaglio, Steve Smollar, and the referees for their useful suggestions and comments on this article, and Arthur Lomas for the improvements he made to our English. Others who helped design and implement HARP include Claudio Massucco, Carlo Innocenti, Armando Allaria, and Maurizio Ciccione.

This work was partially supported by the Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo of the Italian Research Council, under Grant No. 90.00716.PF69, Pos. 115.14231, and by MURST Special Projects (Italian Ministry of University and Research).

References


Corrado Canepa creates electro-acoustic compositions in his electronic music studio and collaborates with the Department of Communication, Computer, and System Sciences at the University of Genoa on the description and processing of sound via software.

Marcello Frixione is a PhD student in the Department of Philosophy at the University of Genoa and collaborates with the Department of Communication, Computer, and System Sciences. He received his Laurea degree in philosophy from the University of Genoa.

Renato Zaccaria is an associate professor of computer science in the Department of Communication, Computer, and System Sciences at the University of Genoa and a research fellow at the Institute of Electrical Engineering. He received his Laurea degree in electronic engineering at the University of Genoa.

Composition Based on Pentatonic Scales: A Computer-Aided Approach

Yap Siong Chua
College of Computer and Information Sciences
University of North Florida
Jacksonville, FL 32216

Pentatonic scales—musical scales of five tones—are used in the music of many cultures. My project uses pentatonic scales to explore some basic techniques of computer-aided composition. This approach can be likened to the study of painting using a limited "palette" and may be of interest to readers with a background in computing and music who want to
experiment with computer-aided composition.

Notation (pitch and duration). The 88 pitches on the piano are usually identified (from lowest to highest) as A0, A#0 (or B0), B0, C1, . . . , C4 (middle C) . . . , C8. The MIDI (Musical Instrument Digital Interface) standard uses integers (0-127) to identify a set of 128 pitches. The pitches A0-C8 are mapped to the MIDI numbers 21-108. While many modern MIDI synthesizers have alternate tunings, we will assume that the default tuning is equal tempered based on 440 Hz for A4 (MIDI No. 69) so that we will have a fixed frame of reference.

Numeric notation for pitches has definite advantages over the conventional names in dealing with computer manipulation of pitch data. The relationship between frequency $F$ (in hertz) and MIDI note number $M$ is given by the formula

$$F = 27.5 \times 2^{M-61}$$

where $S$ is the 12th root of 2 (frequency ratio of a semitone or half step) and 27.5 Hz is the frequency associated with A0. In other tuning systems, musicologists divide the semitone into 100 equal (in terms of frequency ratio) intervals called cents, which can be easily incorporated into the numeric MIDI notation by adding two decimal places. For example, a pitch halfway between C4 (60) and C#4 (61) is said to be 50 cents above C4 and is given a MIDI number of 60.50. The equation above still applies for nonintegral $M$. In addition to the computational flexibility it offers, the MIDI note-numbering system facilitates the translation of musical data into standard MIDI song files for playing with MIDI synthesizers.

We use the number 1 to represent the relative duration of a quarter note. The exact duration is a function of the tempo — for example, the number of quarter notes per minute. Other durations follow naturally: 0.5 for an eighth note, 2 for a half note, and so forth. Rests are notated as negative durations; for example, −1 for a quarter rest, −1.5 for a dotted quarter rest.

Examples of pentatonic music and scales. The examples in Figure 1 are excerpts of pentatonic music from many cultures of the world. Two pentatonic scales are represented in these examples, namely [C4, D4, E4, G4, A4, C5] (variably transposed) and [C4, E4, F4, A4, B4, C5]. These are not the only pentatonic scales, however. Other examples of pentatonic scales include [C4, D4, E4, G4, A4, C5], [C4, D4, E4, F4, A4, C5], and [60, 62, 64, 67, 69, 72]. (At the end of a scale the first note is usually repeated an octave higher.) The last scale, given in numeric MIDI notation, is called the equidistant pentatonic, an idealization of a popular Indonesian scale known as the slendro.

Basic composition. The project illustrates some basic techniques for making music under the guidance of a composer and with assistance from a computer, using only a small set of pitches for simplicity. A musical composition and an English composition share some structural similarities. The hierarchy of an essay or a book (alphabet, words, sentences, paragraphs, chapters, and volume) is analogous to that of a musical composition, which starts with an alphabet of pitches and duration values. These are combined to form phrases (motives and themes). Development and variation of these basic phrases give rise to other phrases. A sequence of related phrases forms a movement, and one or more movements constitute a composition. Additionally, an alphabet of chords permits the formation of chord progressions, the harmonic foundation of a composition.

Elementary computer-aided composition techniques include random sieves, biased choices, and Markov processes. For simplicity, we concentrate on the generation of musical phrases with emphasis on the two fundamental attributes — rhythm and pitch — using an approach that can be characterized as a guided random process based on distributions imposed by the composer.

Rhythm. A rhythm vector is defined as a sequence of real numbers representing relative note and rest durations. For example, the rhythm vector (commas are optional) for the first few bars of "Old MacDonald Had A Farm" is $1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1$. Other considerations such as grace notes, n-tuplets, staccato, and fermata can be handled similarly using the numeric approach.

A rhythm vector can be specified as shown above or with the help of a computer program equipped with a pseudorandom number generator. The composer specifies a biased distribution (or, simply, distribution) of duration values to reflect his or her preferences for the part of the composition being created. This distribution is expressed in the form of a sequence of ordered pairs $(k, D_i)$, $i = 1, 2, . . . , n$, where $k$ is a positive integer giving the count of the corresponding duration value $D_i$. The probability of duration $D_i$ being selected is equal to its count divided by the sum of all $k_i$, $i = 1, 2, . . . , n$. For example, $Rdist_1 = [(2.25) (7.5) (4.1) (2.2) (1.4)]$ specifies a rhythm distribution that can be regarded as an "urn" containing 15 total notes, seven eighth notes, four quarter notes, two half notes, and one whole note. Items placed in a rhythm urn are not limited to duration values of single notes. Rests and rhythm phrases (short rhythm vectors) are also permitted. For example, $Rdist_2 = [(4.1) (2.33 .33 .34)] (2.0) (5.0) (2.1) (5.1.5 .5)]$ is a rhythm distribution in which the probability of choosing a triplet (.33 .33 .34) as a unit is 2 out of 18, or 1/9. A function $RV$ assists in generating a rhythm vector, given a rhythm distribution ($Rdist$) and the total desired duration ($TD$) of the phrase.

Pitch. A sequence of pitches represented in numeric MIDI notation is called a pitch vector. Each element of a pitch vector can be specified explicitly, or a distribution approach can be used to generate pitch vectors containing composer-imposed biases. Consider a given pitch set (a sequence of MIDI numbers in increasing order) and a distribution of skips (defined as relative movements within the pitch set). For example, $Pset_1 = [60 62 64 67 69 72]$ specifies a pitch set corresponding to one octave of a pentatonic scale. $Pset_2 =
Figure 1. Excerpts of pentatonic music from many cultures.
and the rhythm vector it is to match. Although skip sizes are controlled by the skip distribution, interval sizes (generally not the same as skip sizes) can be checked and modified as desired.

Development. A phrase used as a theme can be stated, restated, manipulated, and otherwise modified. Composers constantly create new (sometimes inexplicable) ways to develop and manipulate themes. A few possibilities are

- Reverse the order of elements in a pitch vector, a rhythm vector, or both (retrograde motion).
- Invert the intervals or skips of a pitch vector (inversion).
- Transpose the pitch vector.
- Retain the pitch vector and change the rhythm vector, or vice versa.
- Take portions of the theme and develop each portion individually.

A composition is formed through the creative sequencing of phrases ac-
cording to some compositional form. A computer can be used to experiment with ideas that may be too cumbersome to handle manually.

A short composition. A short composition based on the scale (F3, G3, A3, C4, D4, F4) was generated using the techniques described in this article. Biases in both the rhythm and skip patterns are evident upon examining the score (see Figure 2). It may also be evident that the parts borrowed rhythm and pitch skip patterns from one another. I wrote a program to generate and manipulate the phrases. The pitch and rhythm data thus produced were translated into a standard MIDI song file which was imported into music publishing software (Escort and Score) to produce the score. The score accurately reflects the pitch and rhythm produced by the program without additional manual intervention. At this point, the composer can revise parts (or all) of the composition. Other elements of the composition, such as dynamics and timbres, can be added as well.

Opportunities. The declining cost of computers and synthesizers has made computer-aided composition accessible to anyone with programming skills and musical training. While off-the-shelf products help the computer musician play and print music, to effectively experiment with innovative computer-aided compositional techniques, the computer musician must be very closely involved in translating these techniques into algorithms. The possibilities are endless, even if we restrict ourselves to generating and manipulating musical phrases.

For readers who find this introductory article and its illustrative techniques of interest, I highly recommend the survey by Loy. It is a good starting point for those wishing to learn more about computer composition.

References


Yap Siong Chua is an associate professor of computer and information sciences at the University of North Florida. His research interests include computer graphics, image processing, computer-aided geometric design, and computer music.
Composing by Musical Analog: A Look at Planetary Orbits

Robert Keefe
School of Music, Ithaca College
Ithaca, NY 14850

From the time of Pythagoras, the term number made audible has come to mean the relationship between the pure world of number and the sounding world of music. Today, a new numerology is generating musical compositions using models from probability statistics, fractal geometry, and other nonlinear and chaotic systems. This new way of making numbers audible is now referred to by some composers as a mapping technique or mapping.

Mapping pairs the parameters of an object or event to the parameters of a musical composition, thus creating a musical analog. The object or event can occur in nature or be man-made and will be represented by a series of numbers, letters, or symbols. Compositions based on such musical analogs are found in Heitor Villa-Lobos' New York Skyline Melody (1940) and Montanhas do Brasil (1944), which used a scaled-down version of the 1940’s New York skyline and a mountain range in Brazil, respectively; John Cage’s Atlas Eclipticalis (1961-62) and Luigi Dallapiccola’s Sicut umbra... (1970), both of which employed star maps and star constellations; Charles Dodge’s Earth’s Magnetic Field (1970), where numbers representing the earth’s magnetic field created the musical analog; and Larry Austin’s Canadian Coastlines (1981), where the coastlines of Canada were the object. Some composers have even used the Dow Jones Industrial Average and DNA sequences to generate musical analogs.

My interest in numerical, nonmusical structures began in the fall of 1985 after I found a set of books called The Ephemerides Tables. These volumes contain celestial latitudinal and longitudinal coordinates for the sun, the moon, and naked-eye planets as seen from the biblical city of Babylon, the site of one of the first observatories. The coordinates in the tables are generated from a series of complex calculations. The formulas themselves may be traced back to the works of the astronomer Johannes Kepler (1571-1630).

The orderliness and motion of the universe has always intrigued me, and I felt that this order could be transferred to a musical composition. I had been writing computer-assisted composition programs since 1982 with some very satisfying musical results. Hence, I decided to combine my programming skills with my interest in the planets and create a modern-day music of the spheres.

My celestial coordinates are made audible through an interactive Pascal computer program that generates celestial coordinates for all the planets and the earth’s moon as seen from any point on earth for any given time frame. The program allows me to map these coordinates onto the musical parameters of pitch, dynamics, rhythm, and duration of a composition in a MIDI (Musical Instrument Digital Interface) environment, producing a composition that is a pure musical analog by choosing only a few musical parameters creating a composition that is a partial musical analog.

Compositional environment. Three factors influenced my use of The Ephemerides Tables in a compositional environment:

(1) Each planetary orbit is unique unto itself, similar to itself, and similar to other planetary orbits because they all adhere to Kepler’s three laws of planetary motion. When a planetary orbit is mapped onto selected parameters of a composition, common musical elements among single orbits, adjacent orbits, and nonadjacent orbits might be readily heard. These characteristics lend themselves quite readily to hierarchical relationships and also exhibit a $1/f$ distribution (see Voss and Clarke for an explanation of $1/f$ distribution).

(2) Since each planet has its own orbital speed, any difference between adjacent latitudinal readings will be greater with faster moving planets and lesser with slower moving planets. This characteristic is used to generate the duration and rhythmic activity of a composition. Conversely, the faster the orbit of a planet, the slower the rhythmic activity, and the longer the duration of a pitch.

(3) There is a discernible difference in latitude among the planets. Mercury’s swing in latitude for any given year might be between $-4.91$ degrees and $+3.36$ degrees. Saturn’s swing in latitude for the same year might be between $-1.08$ degrees and $-0.48$ degrees. Due to these latitudinal variations, each planet develops its own melodic profile when mapped onto a synthesizer keyboard. Mercury and Venus generate disjunct melodies while Mars, Jupiter, and Saturn generate conjunct melodies. The melodies generated from Uranus, Neptune, and Pluto center on only a few pitches within a limited range (do, re, mi, fa, sol, for example).

Once the celestial data is generated, I map the latitudinal coordinates onto a pitch letter name, an accompanying accidental (a note foreign to a key indicated by a signature), and an octave displacement (range on the keyboard). The system I devised performs three operations on each latitudinal reading. For example, if the latitude coordinate is 0.25, the following sequence of events occurs:

(1) The rightmost digit “5” from 0.25 maps onto a pitch letter name from A to G. Since our number system is base 10 and there are seven pitch letter names (not including accidentals), some pitch
names must map onto more than one digit. This scheme is user-variable and a typical mapping might be

\[
\begin{align*}
A &= 0 \text{ or } 1 \\
B &= 2 \text{ or } 3 \\
C &= 4 \text{ or } 5 \\
D &= 6 \\
E &= 7 \\
F &= 8 \\
G &= 9
\end{align*}
\]

The 100th's place of the latitude coordinate also acts as an on/off switch or volume control, and a digit may generate a rest instead of a note. The composer can also introduce more rests if desired. Introducing rests allows the generated musical phrases to be phrases instead of streams of notes. Additionally, the longitude of a planet might also generate loudness or dynamics. If a planet reaches 360 degrees in longitude, the loudness is 100 percent (a dynamic level of \textit{fff}, or \textit{fortissimo}); if the planet is at 180 degrees longitude, the loudness would be 50 percent (a dynamic level of \textit{mf}, or \textit{mezzo-forte}), and so forth.

(2) The tenth's place of the latitude number ("2" of 0.25) maps onto an accidental (also composer variable) that is immediately linked to the pitch letter name. I use only three types of accidentals (sharps, flats, and naturals, leaving out double-sharps and double-flats). However, I double up one mapping scheme by

\[
\begin{align*}
\text{Sharp} &= 0, 1, 2, \text{ or } 3 \\
\text{Flat} &= 4, 5, \text{ or } 6 \\
\text{Natural} &= 7, 8, \text{ or } 9
\end{align*}
\]

(3) Finally, the entire latitude number maps onto an octave displacement (vis-à-vis a place on the synthesizer keyboard). The wider the swing in latitude, the greater the range covered on the musical keyboard. If the latitude readings between 0.00 and +1.00 arc assigned to the range between middle C and C an octave higher, the reading 0.25 would fall within the third octave. The entire mapping process would yield the pitch C#3.

In an earlier algorithm, I reversed the roles of the numbers. The tenth's place acted as the pitch generator with very unsatisfying musical results. The musical output centered on only a few pitches within a limited range and with many chromatic inflections. However, this might prove beneficial to someone using a microtonally based tuning scheme.

**Durational schemes.** As the pitch algorithm is running, a durational algorithm maps latitudinal or longitudinal coordinates onto composer-defined durations. The composer has three types of durational schemes from which to choose:

(1) Taking the absolute value of adjacent latitudes or longitudes as determined by the formula \(\text{abs(lat2-lat1)}\) or \(\text{abs(lat2/lat1)}\) and using that as a durational value in seconds or fractions of a second.

(2) Using a coordinate that falls between a specified range and mapping the value onto a designated rhythmic value. For example, latitudes that fall between +1.00 degrees and +1.99 degrees may be assigned to a quarter-note value. Latitudes between +2.00 and +2.99 may be given a half-note value, and so forth.

(3) Superimposing a constant rhythmic value over all pitches such as all 16th-notes, all eighth-notes, or other rhythmic values.

Figure 1 shows "Venus" (from my
composition *The Earth is the Circle Which is the Measure of All* (1988) as a partial musical analog because it only uses the pitch and volume portion of the algorithm. (Figure 1 shows only the first 10 seconds of the work.) A constant durational value of 16th-notes is superimposed over all computer-generated pitches. The upper part of Figure 1 shows the latitude of Venus’ orbit from July 19, 1595, to October 1, 1596, as seen from Graz, Austria. The lower part of Figure 1 displays the musical analog of the planetary orbit. “Venus” can be heard in its entirety on the cassette tape and compact disk available in conjunction with this issue (see the order blank on p. 9).

Figure 2, an excerpt from my composition *Three Movements for Imaginary Dancer*, movement I: “De Stella Nova” (1991), is a polyphonic version of the planetary analog. In this excerpt, all eight planets (as well as the earth’s moon) are made audible as they are panned across the stereo listening field from left to right. Mercury to Pluto, resulting in a modern-day *music of the spheres* composition. Figure 2 is an example of a pure musical analog, where all musical parameters are generated by the planetary data. (The tape and CD contain an excerpt from *Three Movements for Imaginary Dancer.*)

**Decisions on options.** Creating a musical analog from planetary orbits requires a sensible understanding of the simple physical properties of our solar system and incorporating them into the compositional process. The more I know about the celestial model, the better equipped I am in making a musical analog that reflects the structure of the model. I then have the option of deciding whether to make a pure musical
nologic where all of the musical parameters are generated by the computer program (as in “De Stella Nova”) or a partial musical analog where I take artistic control over the computer music output (as in “Venus”).

This is by far the most difficult decision to make when dealing with computer-generated or computer-assisted music and one that I face each time I compose. I cannot codify my intuitive process in a few paragraphs, but I can say that my compositional creativity is influenced by all the different types of music I have listened to and/or performed over my lifetime.

The fine line between adjustment and acceptance of computer output is constantly being redrawn with each new composition. However, my intuitive process seems to maintain a series of subconscious checks and balances over each composition, whether it be a computer-generated tape piece or a work for acoustical instruments. Nonetheless, even my intuitive process may be altered with a new stimulus or input. One of the more fascinating aspects of computer-assisted composition is that sometimes the computer gives you unexpectedly pleasant musical results that might not have been otherwise realized.

References


Robert Keefe, who teaches music theory at Ithaca College, has been using the computer as an additional compositional resource since 1982, and his compositional output has been solely directed at creating musical analogs since 1986. He earned MM and PhD degrees at the University of North Texas.

Robert Keefe

NANYANG TECHNOLOGICAL UNIVERSITY

SCHOOL OF APPLIED SCIENCE

In July 1991, the Nanyang Technological Institute (NTI) will be reconstituted and renamed the Nanyang Technological University (NTU). It will conduct a wide range of courses at degree and postgraduate levels. The School of Applied Science currently offers undergraduate degree courses in COMPUTER TECHNOLOGY and MATERIALS ENGINEERING. Applications are invited from suitably qualified persons to fill teaching positions in the following areas.

COMPUTER TECHNOLOGY

1. Microprocessor-Based System Design
2. Computer Architectures
3. Advanced Microprocessors
4. Computer Communications
5. CAD/Computer Graphics
6. Real-Time Systems
7. Computer Peripherals
8. Control & Instrumentation
9. Image Processing/Computer Vision
10. Artificial Intelligence
11. VLSI System Design
12. Robotics
13. Distributed Computing
14. Compiler Design
15. Fault-Tolerant Computing
16. Performance Prediction & Analysis

Preference will be given to applicants with expertise in both hardware and software.

MATERIALS ENGINEERING

1. Metal Casting
2. Powder Metallurgy
3. Welding
4. Electronic and Magnetic Materials
5. Surface Treatment
6. Corrosion
7. Failure of Materials
8. Materials Selection and Design
9. Modern Analytical Examination of Materials
10. Metal Working and Forming
11. Heat Treatment
12. Polymer Science and Processing
13. Composite Materials
14. Ceramics
15. NDT Techniques
16. All aspects of Physical Metallurgy

Qualifications

Applicants should possess a good Honours degree and a relevant higher degree. Preference will be given to those with teaching/research or industrial experience. Appointment may be offered.

Gross annual emoluments range as follows:

<table>
<thead>
<tr>
<th>Grade</th>
<th>Lecturer</th>
<th>Senior Lecturer</th>
<th>Associate Professor</th>
<th>Professor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gross annual emoluments range as follows:</td>
<td>$53,160 - $64,200</td>
<td>$58,680 - $610,310</td>
<td>$88,660 - $122,870</td>
<td>$108,670 - $146,970</td>
</tr>
</tbody>
</table>

In addition to the above, the Institute adopts the Government’s practice in the payment of a variable bonus, the quantum of which is tied to national economic performance and has, in past years, ranged from 1 to 2½ months of December salary. The commencing salary will depend on the candidates’ qualifications, experience, and the level of appointment offered.

Leave and medical benefits will be provided. Depending on the type of contract offered, other benefits include: provident fund benefits or an end-of-contract gratuity of 25% of the staff member’s last drawn monthly salary for each completed month of service, a settling-in allowance, subsidised housing, education allowance up to a maximum of S$30,000 per annum, passage assistance, baggage allowance and car loan.

The Institute encourages its staff to undertake outside consulting work of a specialist nature. They are permitted to earn and retain such consultation fees up to a maximum of 60% of their gross annual emoluments in any one calendar year.

The Institute has a modern campus with up-to-date facilities for teaching and research as well as residential and recreational facilities for staff and students. It has one of the largest high-speed campus wide network of computing facilities for supporting teaching, research and office automation in this region. Over a hundred engineering workstations and a thousand PCs are supported on NTlnet which runs into all academic staff offices and laboratories. Large computer and file servers available on NTlnet include: three large VAX 8820s, one VAX 8630, one VAX 8200, five VAX 3500 servers and ten MicroVAX II servers. In addition dedicated CAD/CAM facilities are available in the various Schools. The Institute also provides its users with a communication link via NTlnet to a supercomputer installed in the Singapore Science Park.

Further information on the above may be communicated to the Institute through BITNET to: TFWANG@NTIVAX

Candidates wishing to be considered should write to:

THE DIRECTOR
PERSONNEL DEPARTMENT
NANYANG TECHNOLOGICAL INSTITUTE
NANYANG AVENUE, SINGAPORE 2263

giving their curriculum vitae and the names and addresses of referees.

July 1991