

Real-Time Performance via User Interfaces to Musical Structures

Stephen Travis Pope¹

Abstract

This informal and subjective presentation will introduce and compare several software systems written by myself and others for computer music composition and performance based on higher-level abstractions of musical data structures. I will then evaluate a few of the issues in real-time interaction with structural descriptions of musical data.

The premise is that very interesting live-performance software environments could be based in existing technology for structural music description, but that much of the current real-time performance-oriented software for music is rather limited in that it supports only very low-level notions of musical structures. The examples will demonstrate various systems for graphical interaction with procedural, knowledge-based, hierarchical and/or stochastic music description systems that could be used for live performance.

Introduction

This paper discusses various types of software man-machine interfaces to middle- and high-level musical structures in terms of their applicability to live performance. My contention is that there are many good ideas of graphical structure-editing-based interfaces already in the literature that could well be used for controlling performance at levels higher than those addressed by most current systems. The software systems described here can be divided into two classes: those that are primarily designed for use by composers; and novel systems for graphical interaction with structured data.

The paper opens with several comments about formalisms for composition, and how these might be relevant to live performance. This section is followed by a collection of annotated examples of software man-machine interfaces for composition or other tasks, whereby the pertinence of each to live performance is discussed.

Formalisms for Composition

A number of compositional formalisms have been developed through the ages for various types of music. The relationship between musical form and structure and compositional methods is taught in depth in music academies as part of our composition or performance training.

In many musics, the role of text is central in the development of simple musical structures; they often mirror the text's structure, as in the case of many songs or liturgical musics. The central role of diatonic harmony as a structure-giving element in the western music of the last 400 years is also obvious. In the early 20th century, diatonic harmony was replaced with 12-tone formalisms without a parallel advancement in the structural

1. Author's current address: P. O. Box 9496, Berkeley, CA 94709 USA, email stp@CNMAT,Berkeley.edu

aspects. Alban Berg, for example, wrote elaborate sonata-allegro forms in 12-tone technique.

When looking at formalisms for composition as algorithms in the computer science sense, one should cite Donald Knuth's definition of algorithm in terms of finiteness, definiteness, input, output, and effectiveness (Knuth 1973). According to this definition, Guido d'Arezzo's hand and Phillippe de Vitry's use of isorhythms both count as compositional algorithms. Historically, composers have always used the highest technology available to them in composition (although I know of composers who would argue with this statement), so it can be informative to investigate the impact of 20th century mathematics and information theory on this field. Computer software has been applied to music composition since the early days of the availability computers. The first attempts (in the 1950's) to use software for composition fell basically into two areas: implementing complex stochastic methods; and realizing strict serialism (Hiller 1970).

More recently, contemporary mathematical and software techniques have lead to a wide range of attempts—with generally aesthetically-questionable results—into the areas of fractal techniques based on self-similarity over many orders of magnitude, procedural techniques based on the composition-as-programming paradigm (Loy 1989), the use of generative grammars for form generation (Roads 1978), and more advanced artificial intelligence software techniques for knowledge-based composition. The literature in this field is rich.

The relevance of this to live performance is that many software implementations of such formalisms can be executed in real time by contemporary PC- or workstation-class computers. The more complex issue is how to construct man-machine interfaces that facilitate interaction with the structures that are relevant to the chosen formalism or algorithm in a live performance situation. The examples discussed below are all well-known and documented software systems; the newest of them (T-R Trees) is two years old. I believe that any of them could be used to great effect in real-time performance situations where the abstractions they (re)present can be mapped onto the piece's compositional structure.

Software Interfaces for Composers

In the first group of examples below, several software systems for music processing are described and illustrated that take novel approaches to the processes of music composition or performance, or present novel music description languages. The second example section presents several related technologies that the author believes might profitably flow into software tools for the performance of music. It is interesting to note that (though not intended as a selection criterium), all but one of these examples (ARA) are written in the Smalltalk-80 programming system, and make use of the system's object-oriented design, and rapid prototyping and incremental refinement features, often presenting them to the end user in an extensible package.

Novel Music Processing Systems

Sound Kit

Mark Lentczner's *Sound Kit* (Lentczner 1985) is a novel example of the category of samples signal editing tools. In *Sound Kit*, sampled sounds may be edited using the menu of operations visible in the view to the lower-right of Figure 1. The difference between this and other sample editors is that the system maintains an operation tree (visible in the upper-left view in Figure 2), which can be used to describe sounds in terms of Smalltalk-80 language messages. In the example, one can see the results of several copy/cut/paste operations in terms of the Smalltalk-80 messages *copyFrom:to:* and *pasteAt:*. This is a simple technique for capturing the worker's process and presenting it to him or her in a form that may be useful for evaluation of larger-scale patterns. This is also a possible format for describing and interacting with interesting real-time transformations of sampled sound (i.e., the kind that are often undertaken in performances using IRCAM's 4X signal processor). The operation trees allow a fair level of abstraction in the representation and manipulation of sample processing scripts, and thus they could effectively be used for interaction with a signal processor that is processing live sound.

Figure 1: *Sound Kit* sound view and menu (in front) and operation tree (behind).

Kyma

The *Kyma* system, developed by Carla Scaletti (Scaletti 1989), and now commercialized by Symbolic Sound Corporation, is a music composition package that is linked to a real-time digital signal processor (named for some reason the Capybara after a large and ugly rodent). In *Kyma*, no differentiation is made between sounds, scores and compositions; structures are built as hierarchies or concatenations of sub-sounds. The hierarchies can be described by directed acyclic graphs (DAGs) where the leaves of the graphs represent components sounds and the arcs can be marked to denote alterations of the sub-sound, as shown in Figure 2. Thus *Kyma* offers the *musique concrète* abstraction of composition as sound manipulation within a system that scales well over many orders of magnitude (i.e., up to very large scale compositions). This is one of the few current music software systems that breaks away from the paradigms of CMN editors or tape-recorder-oriented sample editors. *Kyma* also allows the specification of new sounds using refinement of existing sounds, so that scores can include comprehensive sound inheritance hierarchies. This system has already been used in live performance by its creators and has aptly demonstrated its effectiveness under these circumstances.

Figure 2: *Kyma* sound view showing a DAG for three plucks with delays and transpositions.

ARA

My 1984 *ARA* system used knowledge-based software techniques to present a composer's user interface based on the paradigm of developing a composition-specific vocabulary to describe each work. When using *ARA*, a composer starts by defining a set of basic melodic and rhythmical materials (visible in the *load_set* expressions in the *motive area* at

the top of the window in Figure 3), and then describes these materials using freely-chosen adjectives (such as those visible in the *load_attribute* expressions in Figure 3). Larger structures are built by moving into the higher-level areas (such as the *voice area* shown in the middle of Figure 3) while keeping some set of attributes marked as “active.” The active attributes are shown as a stack in the menu that is visible to the right of the main view in Figure 3, where something “round, brown, dark” is being set to follow the “consonant, rolling, harder” section. In this way, the composer characterizes his materials at each level of hierarchy using a vocabulary of his or her own design, and the system uses this information to select and manipulate the musical materials. The image of using this type of system in performance entices me. I’d love to be able to play an instrument while instructing the accompaniment system to use “small pointy blue” timbres and mix them with my instrument into a “smooth and dark” textured mix.

Figure 3: The ARA user interface, showing the *motive* and *voice* areas opened and the *gen*, *mix* and *play* areas collapsed.

DoubleTalk

DoubleTalk (Pope 1986) is a system whereby logic-marked Petri nets (called predicate-transition or PrT diagrams), are used for composition. The network can be thought of as a finite state automaton or transition diagram whereby a logic description language (a Prolog variant) is used to describe the conditions under which network transitions will fire, and to describe the types of tokens that flow within the network. Figure 4 shows a *DoubleTalk* net editor (with several of the system menus visible), a net marking editor (with which tokens may be placed in the net), and a form type definition editor (with which one defined token types). The system fell in the no-man’s-land somewhere between composition and performance systems, since low-level networks can be isomorphic to note-by-note scores whose performance can be influenced in real-time, whereas high-level nets can be thought of as abstract machines for semi- (or wholly-) deterministic compositional structures. In terms of the real-time use of such a system, one can simply view it as a vastly more powerful and scalable version of the now popular graphical data-flow editors such as Miller Puckette’s MAX.

Figure 4: *DoubleTalk*’s net, marking and form type editors showing a section of the score of my work *Requiem Aeternam Dona Eis*.

EventGenerators in the MODE

The Musical Object Development Environment (*MODE*) (Pope 1991) has several components that relate to compositional algorithms and tools. (Pope 1989b) describes the system’s framework of “EventGenerators” (EGens)—objects that embody the form of some “middle-level” musical structure such as a chord, cluster or ostinato. EGens can be used to build systems that support a methodology of “composition by refinement” (Pope 1989a) whereby the composer begins to describe a composition in rough terms in terms of general-purpose EGens, and then refines the description or behaviors of these objects to more exactly specify his or her wishes. To my knowledge, there is very little work being

undertaken at present aimed as bridging the gap between notes and higher-level musical structures. I used the EGen system in a live MIDI experiment called *Day*, samples of which I played to accompany a talk about EGen at the 1989 ICMC. In *Day* there were some sections where the user interacted with autonomously-running processes via EGen editors, and others where EGen polled MIDI input and responded to it in various manners.

Figure 5: MODE EventGenerator example showing a code fragment that defines a “DynamicPodCloud” object and the resulting event list displayed in Hauer-Steffens (piano-roll-like) notation.

T-R Trees in the MODE

Another component of the MODE is a collection of tools based on Fred Lerdahl’s “generative theory of tonal music” (Lerdahl and Jackendoof 1983). In this package, patterns of tension and relaxation (or decreasing and increasing stability) in musical motives are analyzed into hierarchical structures according to a set of well-formedness and preference rules defined by Lerdahl and Jackendoof. The similarities between these “tension-relaxation” trees and the prosodic stress trees used by linguists to represent the inflection of spoken utterance can be used to manipulate spoken text and musical materials using the same paradigms. The novelty of the T-R trees system stems from the fact that the hierarchies described in the generative theory (as well as prosody), bridge the gap between expressive and structural hierarchies in music and text—something that is visibly missing in most software tools for composers. The use of the T-R Trees system with speech processing cannot currently be used in real time (owing to its use of the phase vocoder for pitch, duration, and spectral processing), but the possibilities for using the system to generate or modify MIDI streams or commands to simpler signal processing systems are numerous and worthy of further investigation.

Figure 6: A simple T-R Tree editor example showing the prosody of a specific way of reading the word *dunkelkammergespräche*.

Harmonic Analysis Tool

Paul Alderman’s *Harmonic Analysis Tool* (HAT) is a system that analyzes musical chords using traditional rule-based expert system techniques. It is useful in that it can be presented with arbitrary pitch sets and attempts to analyze them using the rules of diatonic harmony. The analysis proceeds from chord analysis to key analysis (if more than one chord is present), using rules about cadences and modulation. HAT’s uniqueness lies in its ability to analyze a given (possibly unstructured) input in terms of a give rule set. Its current rule-base only addresses diatonic harmony, but could be augmented or replaced with a different rule-base relatively easily. For live performance, this is simply an example of a system for somewhat higher-level musical feature extraction, the results of which could be used in any number of ways.

Figure 7: The Harmonic Analysis Tool showing the analysis of a ninth chord; the system's rule base only included chords up to the seventh, so it analyzes this as a Major seventh chord.

Examples of Related Technologies

Alternate Reality Kit

The *Alternate Reality Kit* (ARK) developed by Randy Smith at the Xerox Palo Alto Research Center (PARC) (Smith 1986; 1987), is another example of visual programming whereby the system presents the user with a simulated reality within which he or she can program the laws of interaction of objects. In the example shown in Figure 8 one can see several operation switches, the warehouse of all objects, and the copy-object button (with the Xerox logo, of course). The example shows two pens (the circles to the right of the drawing tables in the upper portion of Figure 8), that have been enrolled in the spring force and the Newtonian laws of motion. They therefore oscillate around each other in the manner of binary stars. The rectangular view in the center-top of the figure is a piece of virtual paper that I "threw" under the pens, creating the sinusoid-like drawings on it. ARK has been used to build simulations ranging in size up to comprehensive models of the interaction of elementary particles in bubble chambers. It is of interest here because it removes users entirely from the domain of computer programming and lets them remain in their own domain of expertise, without limiting their ability to extend the system or to build new models. I like the idea of building virtual worlds that consist of interacting objects for modeling the processes within a musical piece, and then interacting with them in live performance based on an animated alternate reality.

Figure 8: Alternate Reality Kit example; the hand icon is the user's mouse.

ThinkerToy

The *ThinkerToy* system developed by Steven Gutfreund (Gutfreund 1987) is a graphical environment for modeling decision support problems. It uses iconic presentations of possibly complex mathematical operations in the form of "ManipIcons," which have both graphical and semantical properties. ThinkerToy applications are generally configured by system experts for use by domain experts. The system expert analyzes the domain and designs a set of appropriate ManipIcons; the domain expert uses this palette of operations, and may extend it within the graphical manipulation paradigm. Examples of ThinkerToy in action are shown in Figure 9, which shows user interfaces constructed for statistical data manipulation. Figure 9a shows the iconic buttons for several operations that are defined for array-type data. Figure 9b shows a more verbose version of several operations whereby the Smalltalk-80 language messages are visible. In Figure 9c one sees a control panel built for a statistician or scientist who is evaluating data gathered from the Pioneer satellite on its fly-by of the planet Saturn. To the left of and below the main graph view, one can see the icons for various operations on the data—the three large buttons in the lower-left corner, for example, represent three methods of grouping the data. For musical applications, the idea of a domain-specific, extensible set of default operations inte-

grated in a graphical data manipulation environment seems quite compelling. (Gutfreund 1987) contains several of other ThinkerToy examples that make good food for thought in light of possible musical applications. The applicability of this type of system to live performance processing of event-oriented or signal-oriented data is obvious.

Figure 9a: ThinkerToy control board for numerical arrays, showing several of the possible operations in their iconic (ManipIcon) form.

Figure 9b: User interface components for concrete accessory tools that operate on arrays showing the Smalltalk-80 messages they send to their operands.

Figure 9c: A ThinkerToy user interface for data analysis showing a data graph and ManipIcons for several types of operations.

Chinese Temple Design Tool Kit

Ranjit Makkuni's *Chinese Temple Design Toolkit* (CTDTK), also developed at Xerox PARC, is an example of a software package for the capture and aid in the process of refinement of a design. The process of design of façades for chinese temples is modeled as consisting of the phases of definition of a vocabulary of basic elements, design of a compositional topology, and integration of a finished design. The system's various editors use phase-specific input techniques for mapping gestures or graphical configuration onto the properties of temple elements of configurations. Examples of this are the vocabulary editor shown in Figure 10a, where a mouse gesture (shown in the left part of the view) is mapped onto the spacing and orientation of tiles. Once the vocabulary is defined, a topology editor (shown in Figure 10b), may be used to build a larger structure from a collection of basic elements. Any number of transformations are possible in this editor, such as mappings based on mirror symmetry. Each element of a design has a history, and these can be used to annotate the process of refinement via *threads*—lines between elements, topologies and designs that show the heritage of designs within a process. Scene editors are used to represent the overall process of design, as illustrated in Figure 10c where the weight and shape of line connections represent the strength of relationships and design scenes are constructed to capture higher-level aspects of the process. The obvious relevance to composers would be in the fact that CTDTK aids in the introspection and process management operations of creative activities and allows the designer to develop the so-called "libraries of process." This type of system seems extremely useful for groups such as our local *Tonus Finalis* ensemble, which practices structured improvisation. Having the computer act as an "actively-learning listener" during several rehearsals of a performance and then allowing the performers (or a third party) to interact with models of their processes or their interaction during a performance could lead to very interesting results.

Figure 10a: A CTDTK Vocabulary editor whereby the mapping of a gesture onto the spacing of tiles can be designed.

Figure 10b: A CTDTK topology editor showing the palette of vocabulary elements at the bottom and editor where the user places them at the top.

Figure 10c: A scene editor that represents the design process. The paths connecting the various editors represent types and strengths of relationships between items.

Issues

The examples above are intended to demonstrate the powerful abstractions that have been developed for representing and manipulating middle- and high-level musical constructs. I believe that most of them could effectively be applied in demanding real-time environments to produce flexible, powerful and abstract performance instruments. The primary difficulty I see in this actually taking place are the poor support for multi-level hardware and software architectures for performance instruments, and the poor software development environments for real-time software applications.

The first issue, that of system architecture, can be related to two primary problems: system complexity and cost; and data interchange protocols. Many manufacturers (and even many users), still believe that it would be prohibitively expensive to allocate separate processors to critical real-time and user interface tasks. The recent multi-DSP systems developed for NeXT machines at IRCAM and by Ariel_Corp. point in a new direction in this area.

The harder problem is that of standardized communication protocols for the exchange of complex data between processors. The two extremes of MIDI and sample streams simply do not suffice if we are to develop better performance instruments.

Given adequate solutions to these two problems, I believe it will soon be unacceptable to offer performance tools based on the current, very-low-level representations based on events and signals.

Conclusions

This short, informal essay intended to raise several questions and present several rather unorthodox opinions of mine on issues related to computer music systems for live performance. I hope to have demonstrated the power of higher-level abstractions in music representation, presentation and manipulation in terms of a collection of software user interfaces that could be used today to control performance instruments.

References

- Gutfreund, S. H. 1987. "ManiplIcons in ThinkerToy." *Proceedings of the 1987 ACM Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA)*. pp. 307-317.
- Hiller, L. 1970. "Music Composed with Computers." in H. B. Lincoln, ed. *The Computer and Music*. Ithaca, New York: Cornell University Press.
- Lentczner, M. 1985. "Sound Kit: A Sound Manipulator." *Proceedings of the International Computer Music Conference*. San Francisco: Computer Music Association.
- Lerdahl, F. and R. Jackendoof. 1983. *A Generative Theory of Tonal Music*. Cambridge: MIT Press.
- Loy, D. G. 1989. "Composing with Computers—A Survey of Some Compositional Formalisms and Music Programming Languages." in M. Mathews and J. Pierce, eds.

- Current Directions in Computer Music Research*. MIT Press.
- Makkuni, R. 1986. "Representing the Process of Composing Chinese Temples." *Design Computing* 1(3): 216-235.
- Makkuni, R. 1987. "Gestural Representation of the Process of Composing Chinese Temples." *IEEE Computer Graphics and Applications*. 7(12): 45-61.
- Pope, S. T. 1986. "Music Notation and the Representation of Musical Structure and Knowledge" *Perspectives of New Music* 24(2):156-189.
- Pope, Stephen T. 1989a. "Composition by Refinement." *Proceedings of the AIMI Cagliari Computer Music Conference*. Venice, AIMI.
- Pope, Stephen Travis. 1989b. "Modeling Musical Structures as EventGenerators" *Proceedings of the International Computer Music Conference*. San Francisco: Computer Music Association.
- Pope, S. T. 1991. "Introduction to MODE: The Musical Object Development Environment." in S. T. Pope, ed. *The Well-Tempered Object: Musical Applications of Object-Oriented Software Technology*. Cambridge: MIT Press.
- Puckette, M. 1991. "Combining Event and Signal Processing in the MAX Graphical Programming Environment." *Computer Music Journal* 15(3).
- Roads, C. 1978. *Composing Grammars*. (Monograph) San Francisco: Computer Music Association.
- Scaletti, C. 1989. "The Kyma/Platypus Computer Music Workstation." *Computer Music Journal* 13(2): 23-38. also in S. T. Pope, ed. *The Well-Tempered Object: Musical Applications of Object-Oriented Software Technology*. Cambridge: MIT Press.
- Scaletti, C. 1991. "A Kyma Update." in S. T. Pope, ed. *The Well-Tempered Object: Musical Applications of Object-Oriented Software Technology*. Cambridge: MIT Press.
- Smith, R. B. 1986. "The Alternate Reality Kit: An Animated Environment for Creating Interactive Simulations." *Proceedings of the IEEE Workshop in Visual Languages*. pp. 99-106.
- Smith, R. B. 1987. "Experiences with the Alternate Reality Kit: An Examples of the Tension between Literalism and Magic." *IEEE Computer Graphics and Applications*. 7(9): 42-50.

Figure Credits

- Figure 1: (Lentczner 1985) Copyright Mark Lentczner; used by Permission.
- Figure 2: (Scaletti 1991) Copyright MIT Press; used by Permission.
- Figure 4: (Pope 1986)
- Figure 5: (Pope 1989b)
- Figure 9a, b: (Gutfreund 1987) Copyright Association for Computing Machinery, used by Permission
- Figure 10a, b, c: (Makkuni 1986) Copyright John Wiley & Sons; used by Permission.