UNIVERSITY OF CALIFORNIA, SAN DIEGO

On the Formalization of Expression in Music Performed by Computers

A thesis submitted in partial satisfaction of the
requirements for the degree of Master of Arts
in Music

by

Michael Pelz-Sherman

Committee in charge:

Professor F. Richard Moore, Chair
Professor Gerald. J. Balzano
Professor George E. Lewis

1992

Table of Contents

ABSTRACT OF THE THESIS

On the Formalization of Expression
in Music Performed by computers

by

Michael Pelz-Sherman
Master of Arts in Music

University of California, San Diego, 1992
Professor F. Richard Moore, Chair

This thesis investigates the role and nature of expression in computer music. It is argued herein that at least one function of musical expression is to help communicate the structure of the music to the listener. In order to create meaningful expressive deviations around categorically perceived note values, a performer invokes a rich structural description tied to a set of methods for creating sonic variation. Several synthesis techniques for creating subtle variations are described, as are some important contributions to the description of musical structure.

Several formalisms for generating expressive performances by computer have been developed and tested using analysis-by-synthesis techniques. Prototype computer programs or "microworlds" were constructed using Opcode Inc.'s *MAX*™ and Apple Computer's *HyperCard*™, two commercially available software toolkits for the Macintosh computer. While the formalisms were successful in creating the sense of a believable "presence" behind the performance, the performances were frequently of less-than-expert quality. Listening examples and program listings are provided, by means of which the reader may personally evaluate the formalisms. It is hoped that this work will

provide designers of “intelligent” tools for computer music with some insights into the often overly-mystified relationship between musical structure and expressive performance.

1. Introduction

This thesis investigates the role and nature of expression in computer music and proposes some formalisms which composers might employ in computer music systems to emulate some common aspects of expressive musical behavior, which are seen to be important channels through which musical information is conveyed. To adequately address this problem requires both theoretical insight and empirical observation; therefore, I will first cover the relevant background issues and theoretical contributions. Next, I will describe my experiences in applying these ideas to the creation of computer programs that generate expressive behavior according to a "virtual performer" model. Finally, I will discuss the results of my work, and point to some of the larger problems raised by it that merit further study.

The expressive components of musical performance I am interested in synthesizing are structure-based deviations from "categorically correct" values of time, pitch, amplitude and timbre. The concept of "categorical correctness" is based on the Gestalt theory of psychology (Koffka, 1935), which describes how the human perceptual system sometimes attempts to parse stimuli into categories. Within these categories, "certain members are considered normative, unique, self-consistent, simple, typical, or the best exemplars of the domain (sometimes called 'prototypes'). From a psychological point of view, the existence of singular, central, or prototypic elements within categories is thought to reflect a drive toward maximizing efficiency of coding or minimizing the complexity of cognitive objects" (Krumhansl 1990). In musical performance, it is the performer's categorical understanding of the notes in the score that makes expressive deviation possible; conversely, the fact that the deviations center around prototype values allows listeners to perceive the categories.

The paper will focus on aspects of expression generally associated with melodic music; that is to say, music having one principal voice in which pitch is the most important form-bearing element. This is the arena of musical discourse in which the nature of expression is best understood. Pitch-based music representation systems, e.g. Common Practice Notation (hereafter CPN) and standard MIDI files, are well known and easily implemented in computer programs. Nonetheless, the thesis will show that results of this research are applicable in a wide variety of musical contexts, including those in which pitch choice is itself an expressive element, or in which pitch plays a relatively minor structural role.

The effort to endow computers with "human" qualities and capabilities (e.g.: natural language, interface "agents", speech recognition and synthesis, gestural input

devices, graphic and multi-dimensional interfaces, etc.) is rooted in the belief that in doing so, programmers can increase the effectiveness and efficiency with which people can utilize complex software tools to communicate their ideas (Bauersfeld, Bennett, and Lynch, 1992). By programming computers to assume a degree of control over certain expressive aspects of performance, I aim to accomplish a similar increase in effectiveness and efficiency of communication in computer music.

1.1. Orientation

One of the great advantages of computers for musical composition is their ability to generate arbitrarily complex musical signals without contracting the services of large numbers of assisting musicians. Over the past decade, the computer has become an increasingly ubiquitous partner in the performance of nearly all forms of music. However, since computers, by definition, require completely deterministic instructions in order to do anything at all, formal languages[1] are required for the specification of their musical *behavior* at the computational level. *Formalisms*, the conceptual building blocks of formal languages, are the link between human thought and mechanical computation. Adopting the appropriate formalisms for the task at hand is therefore of paramount importance. An existing example of a formalism in computer music is the *unit generator*[2], which has served as a general and powerful conceptual tool in the design of languages for sound synthesis.

The formalisms presented here are my contributions toward the development of formal languages for computer performance of music. The ultimate goal of such an endeavor is the construction of a musical interface agent, a *virtual performer* which, like a human performer, is capable of multiple expressive realizations of a musical score. In the virtual performer (VP) model (see figure 1), the computer emulates aspects of the behavior of human performers in response to various kinds of musical input. Figure 1 depicts a scenario in which a computer explicitly takes on the role traditionally assigned to the performer in musical activity. Note that the picture can include an arbitrary number of human or cybernetic performers playing together and influencing one another's output.

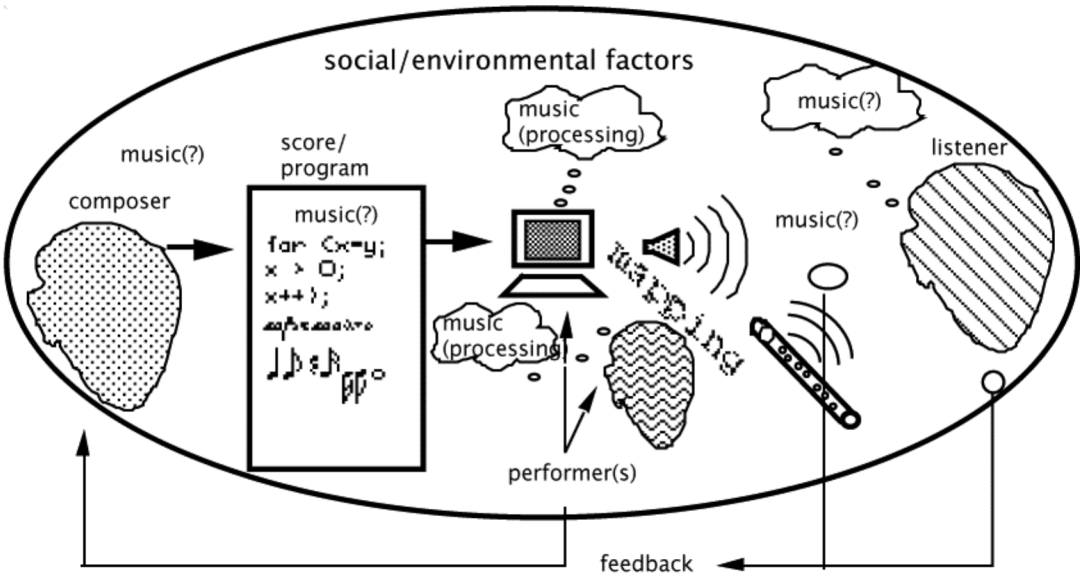
---

[1]In computer programming, a formal language defines a set of symbols - the alphabet - and a set of strings of those symbols. The significance of formal languages in programming is that a given program may be considered as a string of a formal language (i.e. the programming language). The program is produced by writing expressions according to the syntax of the language; this is equivalent to using productions of a corresponding grammar. Programs written in a formal language can be checked by a translator to see if they are strings of the language or not. (Longley and Shain, 1989)

[2]An explanation of the unit generator concept can be found in Dodge and Jerse (1985), pp. 63-65.

At the present stage of development, the formalisms presented here do not attempt to completely represent the expertise of professional musicians, but merely reflect certain aspects of the performance process.

fig. 1: The "virtual performer" model of computer music



The process depicted in figure 1 may be described as follows:

1. The composer arranges and expresses ideas outside of real-time in the form of a score based on a categorical representation of sonic events. The score contains the results of the composer's creative activity in the form of a set of mutually understood instructions for performance.
2. The VP analyzes and *interprets* the score, *adding expression* to the literal instructions represented by the notation. This interpretation may also be influenced by a human performer interacting in real-time with the virtual performer.
3. Using instruments designed or chosen by the composer, the performer(s) realize an interpretation of the score, producing sound waves which are transmitted to the perceptual apparatus of the listener(s).

Figure 1 presents the virtual performer metaphor as a variation of the traditional method of music-making. Both traditional and computational models of music performance share a common information flow, which can be described as a three-stage process: *input*, *analysis*, and *mapping*. In terms of systems design, these stages involve: 1) establishing the categorical *input* parameters available to the composer, 2) establishing

analysis procedures or programs which produce control values, and 3) *mapping* the values returned by the analysis procedure to parameters of tone production.

A simplified example will clarify this process: Suppose we wish to define a function *f(n)* which will produce a deviation from metrically "correct" inter-onset time $\Delta t(n)$. Let us posit that Δt is scaled according to
$\Delta t = \Delta t + Y\Delta t$, where Y is a scaling factor in the range [-1,1] derived from score analysis. Next, we take n to be a note event taken from a CPN score with pitch ***c'*** and dynamic ***ff***. To obtain an appropriate value for Y, we first analyze the context of n and find that

-it marks the beginning of a new phrase group;

-it is preceded by the pitch ***F*** below middle c;

-the dynamic of the previous section was ***p***.

We feed this information to a programmed mapping procedure *f(n)* which draws upon the following processing rules:

-If the note begins a new phrase group, add .1 to Y.

-Add .05 to Y for every semitone in the interval between its pitch and the previous pitch minus one octave.

(19 - 12 = 7 semitones x .5 = .35)

-For an increase in loudness larger than 2 dynamic levels, add .05 per level.

(There are 3 dynamic levels between ***p*** and ***ff***, so this yields an additional .05.)

The sum of these values gives us a total of .5 for Y. We may then map Y to the inter-onset interval according to the formula above.

In the above example, n represents a well-defined, symbolic, categorical input; in this case, a pitch, with an associated dynamic and position relative to phrase structure. The function *f()* represents a process used to obtain Y, which was subsequently mapped to a parameter affecting the expressive realization of n.

Successful implementation of the VP model requires extensive exploration of such formalizations of expressive processes in music. Use of the computer as a tool for this work is an outgrowth of recent intersections between the fields of cognitive science, computer science, and music. Pylyshyn (1984) explains the link between computer programming and cognitive theory construction:

> Digital computers have dramatically altered the criteria by which the success of theories of cognition is judged. In addition to requiring that the theory capture generalizations about behavior, the notion of SUFFICIENCY now permeates theory construction in cognitive science. The term SUFFICIENCY refers to the theory's ability to explain how actual instances of behaviors are generated. Further, the explanation of the generation of such behaviors must be constructive, that is, effectively computable or realizable by a formal mechanism, for example, a Turing machine.

In my explorations, I have adopted what has been described by Honing (1992) as a "microworld" approach. In this approach,

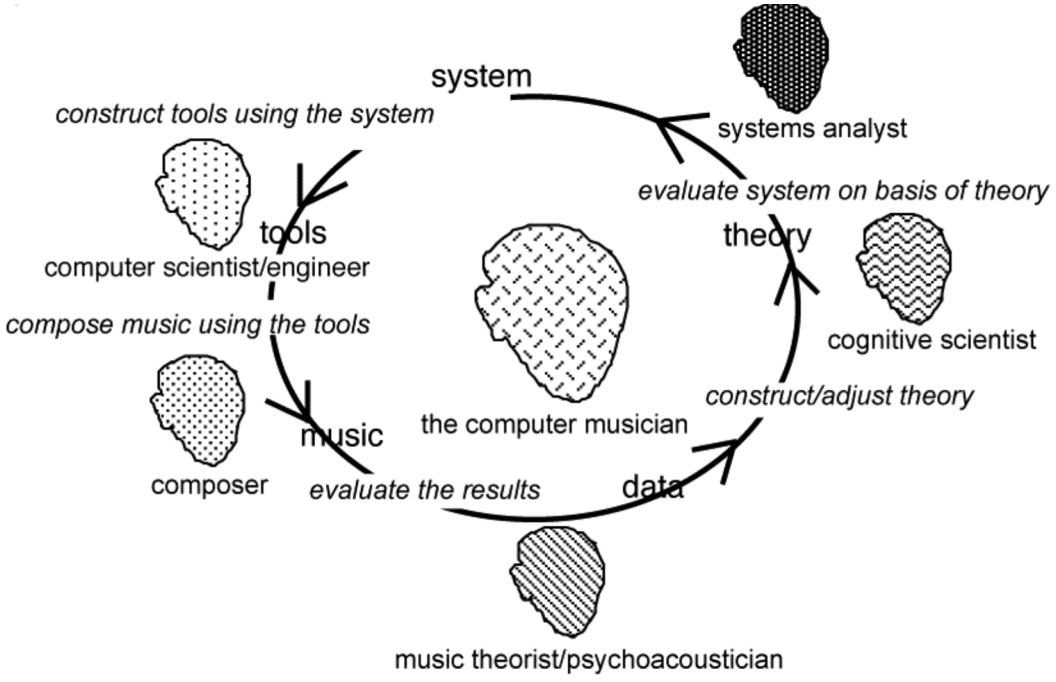
> ...one concentrates on the construction of a small and closed set of procedures and data structures. In an exploratory microworld it is easy to experiment with ideas, vague as they are, to gain more insight into the problem to be modeled. [Constructing] theories in computational form allows for tests on completeness and internal consistency.

Furthermore:

> ...Experimenting with the resulting ad hoc formalization or program may bring out further insights, providing a real understanding, that, in turn, possibly provides for a new formalization, and a new theory. In making problems concrete, deciding what is essential and what isn't, and moving knowledge and understanding from being implicit (e.g. in the control structure) to being explicit (e.g. as data structures), problems become objects, objects of thought, that facilitate thinking about them - just as the turtle gave children "an object to think with" (Papert, 1980), helping them to understand more about geometry.

I have come to see this "exploratory programming" as a cyclical process which involves all of the sub-disciplines of computer music outlined in Moore (1990). This process is represented in figure 2 below. In the diagram, the various roles played by the computer musician are indicated outside the circle in small type, the activities carried out in the various stages of the cycle are indicated in larger type, and the output at each step in the process is indicated in bold type. These roles are typically carried out by separate individuals working in teams (as is the practice at institutions like IRCAM and MIT), as it is increasingly difficult for a single individual to master all of them. The most notable addition to Moore's disciplines is that of the systems analyst; this is an extremely critical stage where the environments and protocols for creating computer music tools are established. An example of the output of such activity is the MIDI communications standard, the widespread ramifications of which are well-known. The importance of the long-term impact of system analysis and design cannot be understated. This paper is intended, in fact, as a sort of pre-analysis for the specification of a system for generating expressive musical performances by computer, the ideal capabilities of which may very well exceed the limitations of real-time music performance systems to date.

fig. 2: the cycle of computer music research



2. Background and Scope

Due to the enormous amounts of data involved, the computer musician's task of generating and managing *sonic variation*, which is a fundamental principle of beauty in music (Seashore 1947), is enormously difficult. Moore (1990) states this problem with particular clarity:

> A fundamental problem in most types of electronic music - especially computer music - is that the machine producing the sound actually does what it is told to do, no less and *no more*. Because our sensory mechanisms are so acutely sensitive to changes in sensation, they are also acutely sensitive to lacks of change. One of the most striking characteristics of any unwavering sound is therefore its synthetic quality. A fundamental challenge of computer music is to impart a perceptible infrastructure to each and every sound by some means that will prevent it from sounding like distilled water tastes. [p. 279, emphasis added]

This very problem is the theme of Robert Erickson's Sound Structure in Music, in which Erickson writes:

> These small differences in the time course of a sound are extremely important for music. As long as composers dealt only with acoustical instruments and human performers there was no difficulty in introducing small differences - the nuance level has always been an important domain of performers. In electronic, tape, and computer music the microlevel either must be composed or decisions must be made about the level of detail worth bothering with. [p.60, emphasis added]

In the acoustical context of human performance, microlevel sonic nuances arise from a complex synergism of the following factors:

1. the interaction of the human motor system with the physical characteristics of the instrument and the acoustical characteristics of the environment;
2. the overall style or mood represented by the piece;
3. the social, traditional and environmental circumstances surrounding the creation, performance, and sound production of music;
4. the performer's execution of expression markings indicated in the score (i.e.: accents, articulations, crescendi/diminuendi, verbal suggestions such as *cantabile*, *agitato*, etc.)
5. the performer's interpretation and communication of the structure of the work being performed.

These factors interact to create the aesthetically pleasing effect known as *expression*. Expression belongs to a class of phenomena that base their very existence on a special kind of interaction of their component parts. Such phenomena are often assumed to be impossible to comprehend; they are called "holistic" or Gestalt phenomena, labels Minsky (1985) deplores as creating a false sense of understanding. Minsky's work

attempts to show how different aspects of mental phenomena are organized into "societies"; it is this organization which gives human behavior its unique qualities. To explain this concept, Minsky uses the analogy of a box made up of flat, individual boards. We must agree that no single board, in itself, possesses the quality of *containment*. In fact, it requires at least six boards "interacting" in a particular manner to create a structure that can claim this quality. In this sense it is impossible to discuss any of the above expressive factors independently. Out of respect for the limitations of tractability and scope, however, this paper will concentrate primarily on the fifth factor: the conscious communication of musical structure through expressive performance, in hope of gaining a better understanding of how *its* sub-components are organized.

2.1. Detecting Expressivity: A "Musical Turing Test"

Expression in music has been the object of much mystification[3]. The Harvard Dictionary of Music claims "Expression in music may be said to be that part of music which can not be indicated by notes, or, in its highest manifestations, by any sign or symbol whatever" (my underscore). Murray Periah, in his liner notes for the last piano recording of Vladimir Horowitz, writes about the great pianist's mastery of the "singing" line:

> Here [Horowitz's] love of singing and opera enabled him to reveal the true shape and character of these beautiful melodies. Perhaps through listening to great singers, perhaps just instinctively, he was able to master the very elusive art of *bel canto* - the laws of which, according to the great Viennese theorist Heinrich Schenker, 'can neither be taught nor transcribed'. They come from an inner understanding of the embellishments and diminutions that give the music life. (my emphasis)

`The skeptical view put forward in these extracts is not at all uncommon among musicians, computer programmers, and people in general. The unwritten rules which guide performers in their expressive decision-making processes are indeed complex and resistant to formalization. To propose to codify any aspect of expressive behavior in computer language is, for many, tantamount to an act of hubris.

Brenda Laurel has written critically about the archetypal, cross-cultural taboo against humans mimicking God's unique ability to create life. Laurel (1990) has

---

[3] This is relatively more true of Western music since the Romantic era; performance practice in the Baroque period, for example, was much more formalistic in its methods of expression. Many musical traditions, in fact, are notated through tablature describing the precise method of excitation of the instrument. The zither music of ancient China, for example, utilized about two hundred such indications. (Erickson, 1975 p. 108)

identified four sources of fear which constrain our vision of the role of computers in art and in our lives:

> The first is that, if we attempt to create computational "entities" that are modeled after sentient beings, we will succeed only in constructing crude and lifeless representations ... it is the fear of failure.
>
> The second fearsome vision is that we will succeed, like Dr. Frankenstein, in creating "life" that grossly amplifies our own shortcomings and flaws - and that suffers and makes us suffer in return. This is the fear of our human fallibility.
>
> ...The third fear is that if we succeed in transforming computers into augmentation devices as empowering and indispensable as the opposing thumb, we will transform ourselves into something "less" than humans - slaves to a cybernetic symbiosis. It is the fear of losing our identity.
>
> The final fearsome vision is that there really *is* a genie in the bottle. When some threshold of knowledge and capability is reached, a new kind of sentience will emerge. ... This is the fear of alien life. [p. 482-3]

One way to address the fear of artificial life infiltrating computer applications is by trying to better understand and appreciate our own cognitive processes. One assumption I made when beginning this research, for example, was that human beings are somehow "programmed" to be capable of perceiving expressive behavior solely on the basis of subtle transformations of timing. To verify this hypothesis, I carried out some informal experiments which confirm that sensitive listeners are indeed able to distinguish among overly-literal, "deadpan" performances (no deviation from the written score), inaccurate performances (too much deviation in the wrong places), and those generally classified as being "musically expressive". Skilled musicians have little difficulty distinguishing between a rhythmically quantized version of a skillful musical performance (in which all note onsets have been forced to conform to a precise timing grid) and the original, un-quantized version. Further, a version that has been produced by adding random deviations (or errors) in timing from the quantized value are distinguishable from both the original performance and the quantized version[4]. This

---

[4]These ongoing experiments augment the work of Johan Sundberg (and others) in this area. Complete results will be presented in a separate paper. Skilled musical subjects (graduate music students) were asked to identify sequences as being rhythmically quantized, randomized, or original versions of human performance. For a stimulus, I chose a performance of Prelude No. 1 of J.S. Bach (from the "Well-Tempered Clavier", Book 1), as this piece contains no tempo changes which are difficult to quantize objectively. The use of a MIDI-controlled synthetic harpsichord "patch" in capturing and presenting both computer and human performances eliminated such factors as "stage presence", dynamics, and timbral quality; the study measures response to expressive timing only. The random sequence was created by altering each of the inter-onset intervals of the quantized sequence according to a random distribution above or below the nominal value. This distribution was statistically identical to that of the original human performance, but the metrical placement of the deviations was chosen randomly. Most subjects were able to correctly identify the sequences after two or three trials. Although the subjects had far more trouble distinguishing between the random sequence and the human performance than

"musical Turing test" encourages speculation as to whether there might exist some way of describing a computable process by which scores may be converted into musically expressive performances.

2.2. Comprehension, Expectation and Meaning in Music

In addition to positing the perceptibility of algorithms for generating expressive behavior, the musical Turing test shows that there is much more involved in simulating expressive performance than simply producing random variations at the sonic microlevel. While many algorithms have been developed for sound synthesis, signal processing, and composition which can amplify to a very large degree the timbral and organizational resources available to computer musicians (Moore 1990, Loy 1989), the mere unmediated employment of these techniques does little to determine the "meaningfulness" of the musical output. Many algorithms as commonly used in computer music, in fact, contribute to the "cognitive opacity" of the music rather than to its comprehensibility. This is due to the difficulty of constructing appropriate analytical procedures for contemporary musical structures and implementing the results as constraints on synthesis algorithms. When expressive decisions are based on pseudo-random or stochastic processes, it becomes less likely that the listener will be able to assign a precise mental representation to what is perceived (Lerdahl 1988); for it is the expressive component in music through which the performer's comprehension of structure is communicated. Expressive deviations from categorical values disambiguate between multivalent possibilities of structural perception, and since it is the listener's ability to recall that structure which makes expectation possible, expressive elements in music performance are therefore an important channel *through which meaning in music is transmitted*.

The psychological evidence supporting the perception of deviation from (or frustration of) expectation as a generator of meaning in music is extensive. Most of our knowledge about the principle of expectation in music concerns how norms are established in the mind of the listener. For example, Fraisse (1982) has established through many years of research that our ability to perceive the rhythmic structure of sonic events, so fundamental to all music, depends on our ability to predict future events on the basis of what is perceived earlier in a sequence. Fraisse adopts Plato's definition of rhythm as "the order in movement", which he and countless other researchers in the past

---

identifying the quantized version, most subjects remarked that they were able to hear some expressive characteristic in the human performance, such as ritards at phrase endings.

century have explored through various psychological experiments. Fraisse concluded from his research that

> A musical composition is a synthesis of very different stimuli that are perceptually unified much as forms and colors are unified in a painting. We distinguish melody, harmony, timbre, and a rhythmic organization consisting of the succession of rhythmic patterns, at the same time identical to themselves and also varying continuously. The unity assures the characteristic of anticipation, which seems to us to be fundamental. (p.171)

A group of German researchers in the 1920's known as the Gestalt psychologists (Koffka, 1962) proposed that the human mind groups both aural and visual stimuli according to the following principles:

1) Proximity - Nearer elements are grouped together.
2) Similarity - Configurations are formed out of like elements.
3) Good Continuation - successive elements which follow each other in a given direction are perceived together.
4) Common Fate - simultaneous elements which move in the same direction are perceived together.

A study by Deutsch (1982) reveals some connections between these Gestalt principles and music performance. Experiments carried out in this study show a strong connection between subjects' ability to correctly recall sequences of tones and the way the tones were structured and performed. Examples were composed which were structured in groups of 3 or 4. Pauses were inserted to temporally segment the tones into groups of various sizes. A marked improvement in recall (a ratio of nearly 2:1) was shown for sequences in which the structural groupings matched those of the performance.

More recently, the Gestalt principles have been extended into the study of what Bregman and McAdams (1985) have called "auditory streams". A *stream* is "a psychological organization that mentally represents a sequence of acoustic events and displays a certain internal consistency, or continuity, that allows that sequence to be interpreted as a whole." The thrust of the investigation of auditory streams is to identify the principles by which the mind organizes musical stimuli, and to examine how these organizations are affected by such factors as tempo, pitch, loudness, and so on. The implications of this research are clear: any music which violates or ignores the Gestalt principles runs the risk of failing to establish perceptible norms, without which communication by means of expressive deviation is impossible.

2.3. Communication Theory and Universality

Since it is the aim of my research to "filter out" expressive factors of style, tradition, and social context, I have searched for theories of expression that transcend stylistic boundaries. Naturally, there are limits to the universality of any theory of music. Nevertheless, it can be said with some certainty that most musical activity involves a special kind of *communication* between composer, performer, and listener. In the 1940's, mathematicians developed the field of communication theory, which states that the amount of information, or entropy, contained in any message varies inversely with the sum of the logarithms of the probabilities of its component symbols (Shannon and Weaver, 1949). Several music theorists have turned to information theory as a basis for making style-independent descriptions of musical structure (Hiller and Bean, 1961; Tenney, 1988; Todd, 1989).

To apply communication theory to music, we must first accept Leonard B. Meyer's (1959) dictum that

> Musical events take place in a world of stylistic probability. If we hear only a single tone, a great number of different tones could follow it with equal probability. If a sequence of two tones is heard the number of probable consequent tones is somewhat reduced - how much depends upon the tones chosen and the stylistic context - and hence the probability of the remaining alternatives is somewhat increased. As more tones are added and consequently more relationships between tones established, the probabilities of a particular goal become increased.

As stated in Todd (1989):

> A theory of expression must describe the process of communication - the message to be communicated, the signal encoding the message, and also the recovery of the message from the signal. On the other hand, a performer playing "with expression" will invariably employ the use of various devices, depending on the instrument and style, which include variations in tempo (rubato), loudness (dynamics), timbre (tone quality) and note offset time (articulation). We may regard these variables as *signals* between performer and listener and the variables themselves as *channels*. (p. 405)

Once having accepted the above, we can utilize Meyer's definition of expressive deviations as "disturbances in the goal-oriented tendencies of a musical impulse". It is these deviations that, Meyer claims, give music *value*. In his famous treatise "Emotion and Meaning in Music", Meyer points out several examples of music from various eras and cultures employing this principle of deviation in the composition and performance of music (pp. 197-255). In Meyer's analyses, communication theory acts like a filter that is relatively insensitive to stylistic differences. This thesis will therefore adopt communication theory as a framework for analyzing relationships between musical structure and performance.

My search for universal aspects of performance practice has been impeded by the fact that in late twentieth-century Western Art music culture, the freedom to choose one's method of establishing musical norms is the right and responsibility of each individual musician, and may vary significantly even among different pieces by the same composer, or among different performances of the same piece by the same performer. This situation has come about largely through a process of assimilation through constant employment of what were formerly considered deviations into the category of accepted norms. Thus "the nature of esthetic communication tends to make for the eventual destruction of any given style" (Meyer, 1956, p. 65).

The transience of musical style has been further exacerbated by the advent of the electronic medium. Electronic music techniques have expanded the sonic resources of composers into vast and relatively uncharted areas, while simultaneously removing the constraints of performance practice. In addition, the widespread availability of low-cost, high-quality sound recording and playback equipment (compact disks, digital audio tape, etc.) has quickened the pace of musical consumption to an unprecedented rate. Jon Appleton, writing in 1968, commented that

> ...when the musical experience is taken *in toto*, we discover that the information content in electronic music experiences is far greater than that for, say, the performance of a new string quartet. The listener must not only familiarize himself with a new sonic vocabulary, he is required to forego the usual cues which the conventional media of expression supply. (p.107)

Appleton's remarks give further support to the notion that the "new sonic vocabulary" would not be so difficult to assimilate if some of the constraints of performance practice were reinstated. Thus, in order to help listeners handle the enormous amounts of musical information generated by the electronic medium, we should consider reinvesting electronic music with some of the cues provided by human performers.

2.4. Studies of Music Performance

Psychologists have long been interested in studying musical performance in order to observe the cognitive processes at work. A number of important scientific studies of expressive performance have been carried out, e.g.: Seashore (1947), Shaffer (1981), Clynes (1983), Longuet-Higgins (1983), Gabrielsson(1974), Sundberg (1991), Todd (1989), Clarke (1985), and Palmer (1989). These studies are relevant to the creation of formalisms for expression in computer-performed music because our perceptual mechanisms are attuned to human performance characteristics. The vast majority of such

studies examine the mapping or "output" stage of the expressive performance process. Output (as opposed to *input* or *processing*) is certainly the easiest place to begin, because the execution of a musical performance produces observable phenomena which can be measured, thanks to advances in technology, with ever-increasing accuracy[5].

Of primary relevance to my present work has been the efforts of these researchers to measure deviations from notated values in performance, and to relate these deviations back to musical structure. Measurements of performance deviations are useful in themselves for the ideas they provide regarding the methods of expressive output. Theories of musical expression, especially of the generative variety, are even more valuable for their contributions to the growing knowledge base of performance rules and processes. Deductive studies such as those undertaken by Seashore (1947) and Gabrielsson (1974) which attempt to generate hypotheses based on actual performance measurements have proven to be less informative to my work than those which take an adductive, analysis-by-synthesis approach (Gabrielsson, 1985). F. Richard Moore (personal communication) has referred to the results of physical measurements of human performance as "Medusa" data: there are simply too many possibilities to be considered for a strictly deductive process to be effective. Adductive studies begin by making an assumption (based on the opinions of experts) and then devise methods for implementing their assumptions in code so that they may evaluate the result. Approaching the problem in this way greatly reduces the amount of information to be sifted through in search of some kind of correlation between input (score) and output (performance).

The concentration in this thesis on structural factors of expression requires me to focus on deviations that contribute in some way to the cohesion or segregation of groups, and to filter out deviations that are not related to structure. In most empirical performance studies, structural and non-structural aspects are "convolved" in a complex and subtle way. Confusion arises because it is often impossible to ascertain whether a particular expressive deviation was the result of premeditated planning, spontaneous will, unconscious habit, reflex, or simple random error. Studies of accuracy in timing by skilled musicians (Sternberg, Knoll, and Zukofsky, 1982) indicate that in clinical,

---

[5]Two examples of such technological advances are the use of MIDI-equipped piano mechanisms which allow the timing and velocity of piano performances to be precisely measured and recorded, and phase-vocoder analysis (Gordon and Strawn, 1985), which can provide extremely accurate time-varying frequency and amplitude data on any digitally sampled sound source. Of course, one should not overlook the fact that we already possess the most sophisticated and appropriate tools for measuring music - our own ears (Balzano, 1987)! Listening to sounds greatly expanded in time with the phase-vocoder can often be very informative(listening example 4).

intentionally "non-musical" tests, even the most highly regarded professional musicians make significant errors (20-50%) in both performance and judgement of small beat fractions less than one-fourth of a second in length. These errors do not vary with beat placement, nor do they improve with increased auditory feedback to the performer; however, there was some dependency established on both the absolute duration of the judged interval and its duration relative to the beat interval. This "background noise" in musical timing can be likened to the (apparently random) frequency "jitter" - usually between ±4 cents from the fundamental - found to exist to some extent in all human performances of acoustic instruments of continuously variable pitch (Dixon-Ward, 1988). Despite their importance in the simulation of "natural" sounds, perceptually random processes such as frequency and timing jitter can be easily added into any computer-based performance system using pseudo-random algorithms, and therefore will not be discussed further here.

Performance studies reveal a basic polarization of opinion over the epistemology of expression. At one extreme, expression in music is seen to be almost entirely a result of the conscious formulation of performance plans (Sloboda, 1982)[6]. At the other, expression is seen as a product of unconscious, emotional and physical responses aquired through conditioning, heredity and evolution (Seashore, 1947). Undoubtedly, both opinions contain a degree of truth. Conscious aspects of expressive performance may generally consist of high-level decisions made by the performer regarding the interpretation of musical structure (e.g.: "I'm going to play this section *rubato* because the harmonies are more intense.") This type of decision requires a relatively large time window of musical analysis and deals with complex qualities like *rubato* and "more intense". Unconscious aspects will generally operate at lower levels and are not as dependent on analysis; for example, some of the more physical aspects may be incorporated into the design of the instrument itself (e.g.: the tone gets louder and brighter and the envelopes move faster as pitch increases.) Expressive factors related to physical motion, for example, may be implemented within a two-note window of analysis according to simple rules (e.g.: 'performers tend to shorten the durations of notes initiating a leap to an extent which depends on the leap interval'.)

---

[6]The term "conscious" is not to be confused with the term "intentional" as used in Clarke (1985) and Shaffer (1976), which simply refers to the input to an executive motor program and carries no associations of consciousness, will, or deliberation.

Although conscious and unconscious performance practices may reflect very different facets of musical behavior, they both contribute to the communication of structure. A computer model which aspires to realize fully the communicative role of the performance process must therefore formalize elements of both the conscious and the unconscious aspects of expressive musical performance. In fact, much of the efforts of advanced musical pedagogy are devoted to bringing the student's unconscious performance deviations under conscious control. The technical difficulty, in both cybernetic and human performance, lies in resolving conflicts between conscious and unconscious processes. (A classic example of the conflict between conscious and unconscious processes is the tendency for less skilled musicians to unconsciously play louder when the music gets faster, regardless of whether a *crescendo* is called for by the score.)

For the designer of a cybernetic performance system, the division of performance processes into "conscious" and "unconscious" groups is both artificial and counter-productive. The problem of process conflict, however, remains. The resolution of conflicting processes seems to have two possible solutions: either the the interactions between the various processes must be accounted for in the system's formalisms themselves, or the system can take a "divide and conquer" approach, defining many performance rules which act independently, the strength of which can be adjusted externally as needed. The contrast between the former, more unified approach and the latter are exemplified by the two computer performance systems I shall discuss in section 2.6.

2.5. Techniques of Sonic Variation

In order to make the *channels* of expressive communication in computer music more explicit, I have attempted to categorically describe some of the technical means by which expressive nuances may be synthesized by a computer. For convenience, have divided these techniques into four areas: timing, pitch, amplitude, and timbre. Such a categorization is bound to be flawed, because these techniques often stretch across parametric boundaries.

The focus here is on those techniques that are easily implemented using existing synthesis languages and programming methods. Implementations of several of these techniques are demonstrated using MIT's *Csound* synthesis language (Vercoe, 1986). Because the human perceptual mechanism seems to respond most favorably to expressive techniques found in human performances, I have indicated, where available, laboratory

studies that confirm the use of these techniques in human performance on acoustic instruments (including the voice). I must point out, however, that it is not my aim here to provide a comprehensive list of synthesis techniques for accurately simulating acoustic instrumental performance. Realism is not the goal here, but rather *plausibility*; in other words, a listener, upon hearing these nuances, should able to construct a mental image of the virtual performer based on a degree of predictability in its responses to input. Since predictability is often enhanced by reference to the familiar, some degree of imitation of human performance is certainly helpful, but not required.

2.5.1. Timing

Since timing is the most easily measured and most general component of expressive performance, it is not surprising that it is the subject of by far the largest body of research. Variations in expressive timing are generally categorized as belonging to, giving rise to, or resulting from metrical or agogic accent, tempo, phrasing, rubato, articulation, chord "voicing" (control over the order of voice entrances), and the style or "feel" of the composition (conventional accent and timing patterns within the measure). Speaking strictly in terms of physical phenomena, however, only two timing variables are actually being affected by the performer: the inter-onset interval (time between event onsets), and the "duty cycle" or percentage of nominal note duration over which a note is actually sounded. These are the "channels" of expressive timing we shall be examining here.

In several performance studies (Palmer 1989, Friberg et al., 1987), rubato patterns (characteristic alterations in tempo) have been shown to be connected to units of hierarchical phrase structure. Todd (1985) proposes that performers "use phrase-end lengthening as a device to reflect some underlying structure abstracted from the musical surface". According to Todd's sources, "variations in relative lengthening, if perceived, contribute to the recovery of syntactic structure by the listener: the greater the lengthening, the more important the syntactic break." Furthermore, he cites three hierarchical levels on which expressive timing is organized:

1. Global - tempo variation over the whole piece
2. Intermediate - phrase level
3. Local - fluctuations at the note-to-note level.

Todd's model determines the amount of tempo fluctuation of each phrase by adding the "embedding depth" of levels 1 and 2. (Note-level fluctuations are not dealt with in the model.) The embedding depth of each phrase determines the amplitude of a

parabolic function which governs the amount of inter-onset deviation from the "flat" timing values given by the score. The function is sampled at each note onset to determine the amount of change in the current inter-onset interval. In its linkage of phrase boundaries to tempo curves, Todd's model is directly and explicitly based on the concept of "prolongation structures" presented in Lerdahl and Jackendoff (1983).

The use of time deformation functions, or tempo curves, to delineate structural boundaries is an extremely important expressive device used by Sundberg, Todd, Lerdahl and Jackendoff, Jaffe, and many others to generate expressive timing. The expressive properties of the tempo curve are noted in Jones (1987):

> Melodic phrases which accelerate and slow have, intrinsic to their structure, correlated tempo changes and together these form dynamic shapes which afford, among other things, expressive communication. There is evidence that some of these temporal features do indeed specify various affective states (Levi, 1982; Clynes, 1977; Clynes & Walker, 1982). Because these shapes are often based on local tempo change, they effectively encourage the entrained listener to shift temporal perspective with the communicated modulation in pace. In this way, I suggest that there is a participatory experience within the listener of the "shaped" affective state that is communicated by the performer. (p.172)

Although most theorists restrict their models to tonal music by limiting their extraction of sectional boundaries and phrase goals solely to harmonic structure, there is little reason to doubt that Jones' "arcs of tonal motion" would apply equally well to music in which structure is primarily determined by other parameters; we must simply enlarge the criteria by which these conditions are established.

Computational methods for controlling the synchronization of multiple event streams developed by Jaffee (1985) allow non-real-time systems to synthesize both "compensated" and "non-compensated" types of rubato, as defined in Wessel, Bristow, and Settel (1987, pp. 108-116). Todd's model allows only for the non-compensated type, where the performer does not adjust for his slowing down by speeding up elsewhere. When synchronization with other instruments is desired, temporal deformations must be compensated. To accomplish this, Jaffe presents computational methods called "time maps" whereby a running onset-time tally is used to allow for compensation of temporal perturbations. Jaffe claims (without specific reference to empirical research) that "this approach is much closer to the way human musicians play. They do not 'accumulate' error. Rather they are continually compensating for their 'mistakes'."

Palmer's study of chord asynchronies in piano performances supports the commonly-held belief that pianists play melodic notes slightly ahead of other notes in order to "bring out" the melody from the general texture. There was also significant

correlation between beat location and mean onset difference between the notated melody and the remaining voices: in general, chords occuring on the downbeat were played much more asynchronously. Assuming the data structures used to represent the score distinguish melody from accompaniment somehow, the forward-shifting of melodic onset-time is a fairly straightforward matter. This forward-shift may also be applied to inner voices of chords which are more melodically "active". The entire process may also be scaled according to beat placement. Listening example 6 contains results from a microworld experiment (see section 4.1) in which these ideas are implemented with some success.

The control of overlap (or *articulation*) is often dealt with in note-based sound synthesis languages (e.g. cmusic or Csound) by multiplying the stated duration of each note by a factor called a "duty cycle" (Moore, 1990, p. 542). Assuming a note list in which the duration of each note is equal to the time until the next note onset (or "inter-onset interval"), duty cycles less than 1.0 will create *staccato* articulation, while duty cycles greater than 1.0 will yield *legato* phrasing. Palmer's investigation of overlaps in musical performances reveals significant correlation between a performer's interpretation of phrase boundaries and the amount of overlap between successive note events. Performers tend to insert a "micropause" at phrase boundaries, a fact exploited by the Sundberg research team in the development of their rule-based performance system.

2.5.2. Pitch

Musical pitch has long been the principal form-bearing element of Western music. This may be due to the fact that pitch makes a good "carrier" for other parameters of expressive control (i.e.: amplitude and timbre). Certainly, pitch is the domain in which Western ears hear most categorically; we routinely parse the frequency continuum into discrete pitches, while categories in other domains (time, amplitude, timbre) remain much "fuzzier". In addition, particularly in equal-tempered tuning systems, pitch and interval classes, collections, and sequences remain perceptually invariant under many different types of transformations. (Balzano, 1980). This has led to the widespread (mis)conception of pitch as the "most important and least expressive" element of music (Mathews, 1991). However, the importance of pitch deviations to expressive performance has been nearly as extensively documented as expressive timing. Expressive pitch modifications of a single musical note include vibrato, trills, pitch bend, and portamento (glide between notes). Expressive aspects of pitch sequences will be discussed under the auspices of intonation.

Vibrato has been studied extensively as a rich source of expressive nuance (Seashore 1947, Sundberg 1982). Since in acoustic instruments it actually affects amplitude and timbre as well as pitch, it almost deserves its own category; I have placed it under pitch merely for convenience. One of the most important perceptual functions of vibrato in sound synthesis is that it helps listeners fuse component tones into a single, unified "voice", thus enabling our perceptual mechanism to isolate single tones out of a dense texture. Because vibrato has such close associations with the human voice, a "human" quality can be imparted to almost any continuous synthetic tone by adjusting the vibrato characteristics to resemble those of human singers. In his composition *Phoné*, John Chowning exploits this fact by adding vibrato to components of bell tones to impart a fused, "singing" quality to them. Another important function of vibrato is to avoid the beats which arise in ensemble situations from deviations around categorically correct fundamental frequencies. As Sundberg notes in his article "Perception of Singing", "the use of vibrato offers the singer more freedom in the choice of fundamental frequencies and this freedom is used for purposes of musical expression" (p. 91).

There are three physical aspects of vibrato within one period of oscillation: the frequency or rate, amplitude or depth, and waveshape of motion around a central value. Of these parameters, rate and depth are the most commonly manipulated features of vibrato. Expressive vibrato is created by varying these parameters over time in a manner consistent with higher levels of structure. For example, performers may increase the vibrato frequency and/or amplitude as musical intensity increases. Increased vibrato amplitude and rate has the effect of making a single voice stand out from the surrounding texture. Vibrato rate and amplitude envelopes thus constitute a powerful and effective aspect of expressive output.

There are several aspects of vibrato, however, which are often overlooked by synthesis instrument designers. For example, in the singing of many folk cultures (e.g. Islamic devotional singing) vibrato is not applied continuously, but rather seems to "toggle" on and off over the course of a held note. A stochastic function can be used to create this effect by setting a threshold beyond which a random signal will toggle the vibrato on or off. In software synthesis, care must be exercised with this technique to avoid clicks caused by the vibrato function suddenly "kicking in" at a high point in its cycle. Listening example 1 contains a demonstration of this technique, using Csound's "port" unit generator to smooth frequency discontinuities. The vibrato "density" is made to vary from minimum to maximum over ten seconds, while the vibrato rate simultaneously increases from 4.5 to 7 Hz.

The transition speed between positive and negative vibrato "peaks" is another under-utilized but important expressive parameter; sharp or discontinuous transitions yield a more "yodel-like" vibrato, while more sinusoidal transitions are more typical of the "bel canto" style Western ears seem to prefer. "Real" vibrato is never perfectly sinusoidal, thus the use of "pure" sine functions as vibrato waveshapes is likely to be detected by the ear. To simulate the "jitter" found in all human performances, instrument designers often add random deviations in amplitude and frequency to the vibrato function. In addition, composers may wish to control the transition characteristics of the waveshape. One way of accomplishing this is by variable interpolation between sine and square-wave functions. This can can be easily accomplished by employing the following equation:

equation 1: Interpolated vibrato

$$x(t) = A((1-K)\sin(t) + K(\text{square}(t))$$

where K is a scaling factor between zero and 1, t is the function table index, and A is the overall waveform amplitude. Increasing values of K produce increasingly square-wave-like functions. Listening example 2 demonstrates the effect of slowly changing the value of K from 0 to 1.

Another type of control infrequently imposed on vibrato functions is to independently vary the extent above and below the fundamental frequency, thus creating an asymmetrical vibrato. One way to accomplish this is to use a conditional operator to test the sign of the vibrato waveform, then multiply the positive and negative halves of the wave by different values. This method, while computationally expensive, has the virtue of being independent of the basic vibrato waveshape. In listening example 3, a Csound instrument was devised in which a single constant (K) between 0 and 1 is used to determine the emphasis on positive vs. negative vibrato deviations. The positive half of the vibrato function is multiplied by K, while the negative half is multiplied by (1-K); thus a value of 1 for K yields an "all-positive" vibrato, while a value of zero yields an "all-negative" vibrato. Values in-between create various gradations of asymmetry.

Although generally considered separate processes governed by different sets of parameters, I have found it useful and natural to consider both continuous and discrete pitch ornamentation as special cases of vibrato. In the case of discrete ornaments (such as trills), the vibrato function becomes discontinuous, having a large (generally one half-

step or more) all-positive or all-negative amplitude, perhaps inverting itself occasionally. Pitch bend ornamentation can be defined as a slow, continuous vibrato function with a period greater than or equal to approximately one-half of the note duration. The fact that human performers seldom use continuous and discrete pitch ornamentation simultaneously (listening example 4) may explain the preference we seem to have for this constraint on pitch variations.

Pitch bends may be distinguished from portamenti in that the latter are simply transition curves from one note's pitch level to the next. The parameters of an ideal musical portamento are its start time relative to the note duration, and the percentage of travel toward the target pitch; this way, the arrival time of the target pitch is not affected by the portamento rate. A Csound synthesis instrument implementing this technique is demonstrated in listening example 5. Note the distinction between this method and MIDI standard portamento, which uses a single global value to represent glide time. Since MIDI systems generally have no way of encoding what pitch the performer intends to play next, the glide cannot begin until the performer has triggered the next note. This will result in a delay before reaching the note, which is often undesirable. A virtual performer can take advantage of contextual knowledge, however, using true portamento rather than pitch "lag" to achieve more carefully controlled note-to-note transitions. Another cybernetic advantage is the ability to perform arbitrarily discontinuous pitch bends; a human performer using a pitch bend wheel or joystick physically cannot do this.

Alternatives to and deviations from the 12-note equal tempered tuning system have been widely explored in computer music, spurred by the computer's ability to easily produce the exact frequencies specified by the composer (e.g.: Blackwood's "Microtonal Etudes", Carlos' "Beauty in the Beast", etc.). These alternatives play a very important role of the expressive resources of computer music for a number of reasons. Composers may be interested in using greater numbers of equal divisions of the octave in order to achieve a higher "melodic resolution"; they may wish to achieve greater "harmonicity" in their music by using tuning systems derived from simple ratios; they may wish to carefully control the frequency of the "beating" caused by the proximity of multiple tones, or they may be searching for a particular "exotic" or ethnic sound.

Several MIDI synthesizers (e.g.: Emu's Proteus, Yamaha's TX802 and SY77) now incorporate the ability to internally define microtonal tunings, and proposals are underway to update the MIDI specification to accommodate both "fixed-gamut" and continuously variable microtonal capabilities (Scholz, 1992). At present, however, most commercial synthesizers are extremely limited in their ability to change tuning

parameters dynamically in relation to other information. This makes rules regarding intonation difficult to implement using current real-time technology. Nevertheless, industrious programmers have managed to devise quite elaborate MIDI-driven tuning systems by allocating pitch bend polyphonically across several MIDI channels. Anders Friden's "Rulle" program (see section 2.6), for example, implements intonation rules using pitch bends which mimic the way human performers adjust their intonation over the course of a note. The rules are based on research findings by Sundberg and his colleagues which indicates that vocal, wind, and string performers tend to favor Pythagorean intonation. The rules are implemented using harmonic information added to the input score. The use of Pythagorean intonation in human performance has been confirmed by laboratory research and discussed in detail in Theodor Podnos' "Intonation for Strings, Winds, and Singers" (1981,). It appears than even in twentieth century "atonal" music, performers make local adjustments which relate to the local tonalities implied by the score; live performances may owe a great deal of their attractiveness and expressivity to this property.

2.5.3. Amplitude

There are three levels of expressive amplitude in music: *dynamics*, *envelope*, and *tremolo* or amplitude vibrato. Dynamics correspond to categorical peak amplitude values applied to single notes or across entire sections. Envelope refers to the overall amplitude shape of a single note event. Tremolo effects are simply the transference of pitch vibrato into the domain of amplitude. The plausibility of synthetic sounds can often be enhanced by introducing a degree of amplitude change connected to the same oscillator driving the pitch vibrato.

The qualities of the amplitude envelope are known to be extremely important for conveying expression. Sundberg has cited a study by Soviet researchers Kotlyar and Morozov (1976) in which it was found that singers were able to convey emotional information primarily on the basis of amplitude curves extracted from their performances and used to modulate an artificial signal deprived of its intonational and spectral characteristics. Clearly, then, amplitude modulation should rank high on the list of expressive techniques.

In sound synthesis, control over the amplitude envelope (or of any other expressive parameter, for that matter) requires both precision and power. Great precision has been made possible by engineering techniques of software synthesis; the challenge

now seems to be in developing interfaces which give us powerful ways of manipulating large numbers of parameters with a small number of commands.

Toward this end, I have found it useful to establish a small set of four-segment amplitude envelope *families*[7], and to formally describe how these are internally scaled. I have developed six such families, based on the six non-redundant permutations of two three-directional line segments (up, down, or horizontal): up-down or "swell", up-horizontal or "ramp", down-up or "Sfp", down-horizontal or "dive", horizontal-down or "tenuto", and horizontal-up or "delay" (figure 3b). Note that these families determine only the "internal" shape of the envelope - the "decay" and "sustain" segments in traditional synthesizer parlance. All envelope attack and release times are scaled independent of family, depending instead on the *articulation type* (either staccato or legato) and scaling of the note event (figure 3a). The diagrams below indicate the unique manner in which each of the families is scaled. The scaling procedures employ certain experimentally chosen constants in order to preserve the group identities of the families and to prevent segments of zero length from creating waveform discontinuities.

Although four segments is not enough to facilitate accurate imitations of acoustic instruments, limiting the number of envelope segments to four offers a good compromise between simplicity of coding and richness of expression. This limitation also admits of a standard format for specifying breakpoint parameters in software synthesis instruments while maintaining compatibility with most MIDI synthesizers, which generally use 4-segment amplitude envelopes. A graphic representation of these envelope families and their scaling procedures is given in figure 3b. For a code listing in the HyperTalk language which demonstrates in detail how these envelope parameters are computed, see appendix B.

In software synthesis one frequently encounters the issue of *duration-relative* vs. *fixed-duration* envelope construction[8]. A duration-relative envelope algorithm may use a percentage of the note duration to calculate the length of envelope segments, in which case the segments will all scale themselves according to the length of the note. While this can be an interesting and even quite expressive effect, it may lead to undesirably long

---

[7]The term envelope "families" was used by Curtis Abbott (1989) to describe his program's approach to the grouping of individual ramp segments. My use of the term is more specifically related to the directional characteristics of the 4-stage envelope shape as a whole.

[8]This issue has generally not been considered relevant to real-time computer music (i.e.: MIDI-based systems) because it is impossible for the computer to predict human-performed note duration in real-time. However, there is no reason to ignore duration as an expressive scaling factor in computer-performed music!

attack time in the case of unusually long durations, for example. Fixed-duration envelopes, on the other hand, move the same rate regardless of the actual note duration, lacking variety. The scaling procedures for the envelope families employ a hybrid approach whereby the envelope parameters to be scaled according to an abstract "articulation value", which may or may not be influenced by duration, frequency, peak amplitude, or any other combination of factors.

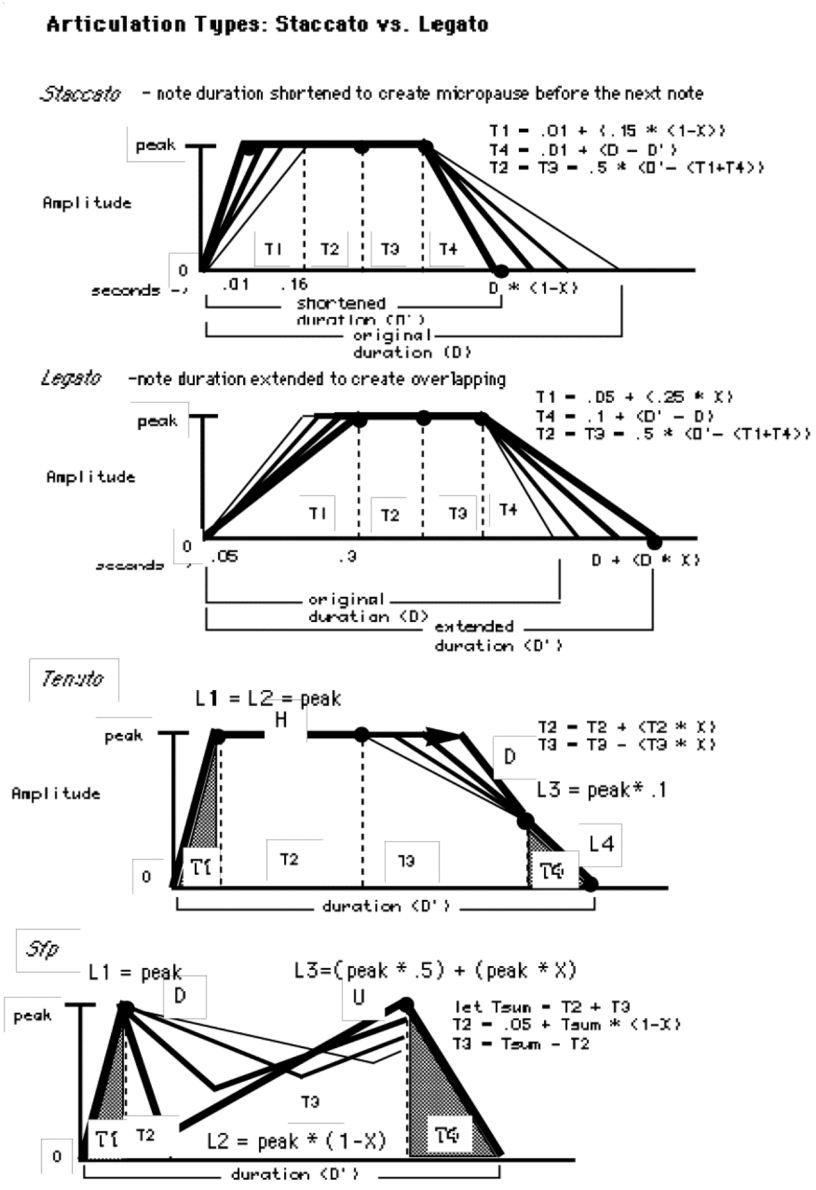


fig. 3a: Scaling of articulation types. All envelope parameters are scaleable by the musical intensity value **X**. Line thickness corresponds to articulation value.

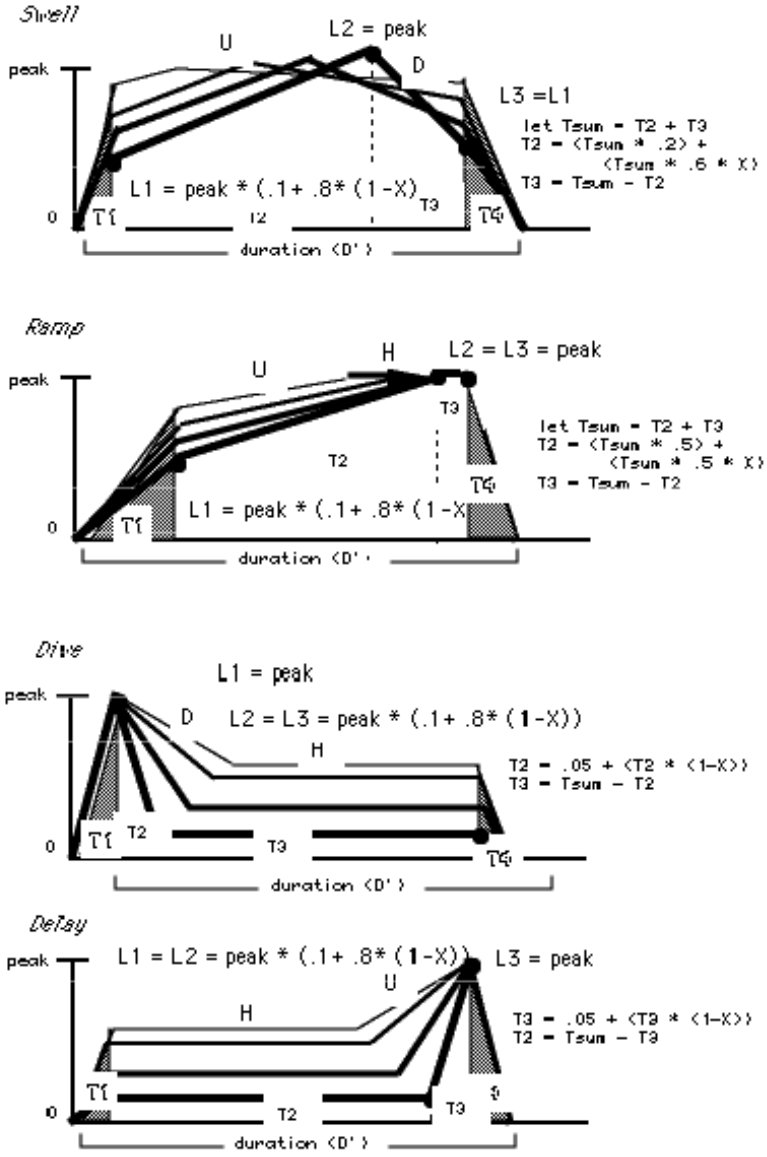


fig. 3b: Envelope families and their associated scaling procedures.

2.5.4. Timbre and Instrument Design

A complete examination of expressive musical performance techniques cannot ignore the contribution of the intrinsic properties of musical instruments, for surely one of the chief factors governing the perception of expressivity in music is the interaction of musical structure with the behavior of the instrument itself. The struggle to overcome the physical constraints of a production system is one of the most important ways in which

human performers generate exciting performances; this phenomenon is often referred to in synthesis literature as "effort" (Bennett & Rodet, 1989; Ryan, 1991). When played skillfully, acoustic instruments communicate information to the listener about their physical structure, which helps the listener get oriented to the music much in the same way that communication of musical structure helps the listener understand and appreciate what they are hearing. Many expressive parameter interactions are a direct function of physical properties of musical instruments, such as the relationship between envelope functions, loudness, and frequency, as in the well-known correspondence between pitch, amplitude, and the "brightness" of instrumental and vocal timbres.

However, it is important to bear in mind that many (if not most) computer musicians prefer to think in terms of abstract "sonic gestures" rather than striving toward imitation of the singing voice or orchestral instruments, particularly as the computer can easily extend any such model beyond the scale of physical human limitations. Even in such extended models, however, the fact remains that in order to maximize the expressive potential of one's instruments, a composer must define some normative relationships between the various parameters of the sonic behavior of the instrument (e.g.: as pitch increases, brightness increases). If no perceptible relationship exists among these parameters, the listener will be presented with fewer opportunities to assemble the sound into meaningful information. This argument is supported by J.K. Randall (1972), who has questioned the validity of the concept of isolated, "inherently interesting" timbres. Randall's argument challenges the frequently encountered approach to computer music which presumes that designing "better" timbres will automatically lead to better music. According to Randall, the scaling of sonic materials in contextually meaningful ways is of far higher importance than their qualities as individual events.

Composers of contemporary instrumental music seldom begin a project before learning the limitations and timbral characteristics of the instruments they are writing for. In computer music, however, we are confronted with the task of establishing these characteristics for ourselves. When a composer writes the name of the instrument in the margin of a score, a great deal of information is instantly established about the nature of the music. First and foremost, we know whether the instrument is monophonic or polyphonic. We also know whether the performer is capable of independently controlling the sounding time of each pitch (as is the case with the organ). We can thus divide all instruments (including those created in software) into three broad categories:

1. Single voiced, or "Melodic"
2. Multiple voices freely ringing (e.g.: a carillon, chimes, etc.), or "Percussive"

3. Multiple voices independently damped, or “Polyphonic”.
(A formal presentation of how these categories affect duration is represented in Appendix A).

Furthermore, instrumental music composers always have a good idea of the timbral resources of instrument, which tend to arrange themselves along various axes. For example, string players can bow further up the neck *(sul tasto)* or closer to the bridge *(sul ponticello)*; they can likewise play with the point of the bow or at the heel. In “Sound Structure in Music”, Robert Erickson presents five acoustic parameters which define the dimensions of timbre, based on a study by J.F. Schouten (1968):

1. The range between tonal and noiselike character.
2. The spectral envelope.
3. The time envelope in terms of rise, duration, and decay.
4. The changes both of spectral envelope (formant-glide) and fundamental frequency (micro-intonation).
5. The prefix, an onset of a sound quite dissimilar to the ensuing lasting vibration.

(Erickson adds the “suffix” to his own list.)

Wessel (1979) proposed a timbre “space” in which the vertical axis is related to the spectral energy of the tone, and the horizontal to the nature of the onset transient, or “prefix”. Risset and Wessel (1982) have proposed the idea of navigating through timbre space as a means of harnessing the power of digital sound synthesis within a coherent syntactic context.

Although a complete examination of timbre space navigation techniques is beyond the scope of this paper, I will describe briefly here an exploratory programming method which has proved useful in thinking about this problem. Using the MAX prototyping environment, I have developed a program which connects the Macintosh’s x-y coordinates to synthesis parameters of a Yamaha TX7 synthesizer via MIDI system-exclusive commands. Offloading the responsibility for envelope generation to the synthesizer makes the computational aspects of this experiment much simpler. The mouse position is marked by an oval which changes its color and shape according to information entered by two foot pedals or other continuous controllers; thus, up to four parameters can be controlled simultaneously with continuous visual feedback. Notes are played by clicking the mouse. Changes in the oval size reflect the overall amplitude and “brightness”; color changes reflect changes in fundamental frequency. Attack characteristics are represented by the verticality/horizontality of the oval shape. This feedback seems to be extremely helpful in orienting oneself in timbre space. Examples of

synthesis parameters I have controlled using this technique are the harmonicity ratio and peak index values of the modulating oscillators, envelope attack and decay times, etc.

While the above may be an interesting technique for sound design, interactive composition and live performance, the goal of these experiments within the context of this thesis has been to help me discover effective mapping procedures which allow the virtual performer to move through timbre space as a function of musical structure.

2.6. Descriptions of Musical Structure

The influence of the structure of a composition to its expressive realization has long been apparent to music theorists. In 1892, the French pedagogue Mathis Lussy wrote an extensive pedagogical treatise entitled "Musical Expression: accents, nuances, and tempo in vocal and instrumental music." Although clearly geared toward the musical conventions of the late 19th century, Lussy's writings foreshadow many of the contributions of contemporary theorists. The role of deviation from expectation, for example, was well understood by Lussy, as evidenced by the following passages:

> Now it is precisely these unexpected, irregular, and, as it were, illogical notes which more especially have the faculty of impressing the feelings. They are the notes that engender *expression*, because they are the elements of stimulus, movement, force, fire, and contrast. The musical sentiment, being accustomed to expect affinity, regularity, and symmetry, is startled and disturbed by these unexpected and foreign notes. They baffle its expectations, disturb calculation, break the thread of customary progression, and impede advance. (p. 8)

> A composition may be compared to a country passed through by a traveler. As long as the road is smooth his pace is regular. If it is interrupted by ditches and banks, rough places and rising-grounds, his walk and his pace will vary. And just as the traveller regulates his pace according to the nature of the ground, the musician will modify his rate of *tempo* according to the ascending or descending structure of the phrases, and the quantity of the harmonic transitions and undulations. (p. 166)

Lussy identifies three types of accents: *metrical accents* which concern the role of strong and weak beats; *rhythmic accents* which concern the performance of rhythmic patterns; and *expressive accents* which concern the performance of "exceptional" individual notes. In the music of Lussy's contemporaries, this was probably a very apt categorization; today, however, the vocabulary of meter and rhythm has been too greatly expanded to allow for such explicit categories.

A fundamental problem, noted in Clarke (1985), in formalizing the relationship between structure and expression is the absence of any generally accepted, well-formed system of musical analysis and structural description. In this section, I will describe three

contemporary theoretical contributions toward formalizing the representation of musical structure. While there are no doubt additional examples in the literature of analytical systems which could offer much to this endeavor, these three systems have been the most frequently referenced among the literature I have reviewed, and have proved to be the most versatile and powerful methods I have encountered. Each of them recognizes the importance of Gestalt grouping principles, and each has contributed ideas to my own work in the explorative creation of computer programs for expressive performance of music.

2.6.1. Tenney's Temporal Gestalt theory

The composer and theorist James Tenney, in 1975, contributed *Meta + Hodos*, an essay on his ideas concerning the application of Gestalt principles to the description of music. This document presents examples of the analysis of contemporary Western music in terms of information theory, and thus constitutes an essential resource for computer programs which attempt to make aesthetic decisions based on Gestalt grouping principles. Tenney posits the existence of *temporal gestalt-units*, or TG's, which are formed "at several different hierarchical levels" according to the principles of cohesion and segregation. The number of levels in a given piece, and the relative durations of the TG's at the adjacent hierarchical levels varies, depending on such things as style, texture, tempo, the duration of the piece, etc. A TG at the lowest (or first) hierarchical level is called an *element*; at the next higher (2nd) a *clang*, and at the 3rd level a *sequence*. Tenney gives his analysis of the nature of the musical parameters and sub-parameters thereof which make up these TG's (pitch, timbre, and musical time); he then goes on to define three distinct perceptual aspects of form (*state, shape, and structure*) and explains how these aspects determine the *content* of TG's at the next higher level, as well as the *context* of TG's at the next lower level. Parameters which vary widely over the course of a TG are called *formative*; those which remain relatively consistent are called *cohesive*.

The definitions of musical units Tenney proposes are important because they provide a general method of parsing music into discrete subsections according to their *entropy*, or variation value according to

$$H = -\Sigma\, p(i) \log_2 p(i) \text{ (bits per message)}$$

where $p(i)$ is the relative frequency of the occurrence of symbol i over the TG being analyzed. Tenney points out that i can represent the TG's lower-level states, shapes, or

structures, as well as the relationships between them. Tenney's theories have been implemented in a computer program which performs automatic analysis of the TG's in a monophonic composition (Tenney and Polansky, 1980). The program's analyses compare quite favorably to those done by human theorists using personal judgement. The fact that Tenney's theory has been successfully implemented in a computer program is encouraging; however, in Lerdahl and Jackendoff (1983), criticisms are raised:

> [Tenney and Polansky's system] does not comfortably account for vague or ambiguous grouping judgements, because of its numerical character, and they note the essential arbitrariness in the choice of [certain numerical values]. And, although aware of the need for global rules such as those of symmetry and parallelism, they do not incorporate these rules into their system. It is our impression that they do not really confront the difficulty of how in principle one balances global against local considerations. (p. 55)

2.6.2. Lerdahl and Jackendoff

Lerdahl and Jackendoff's 1983 treatise, *A Generative Theory of Tonal Music* (GTTM), has become an almost standard reference in contemporary music theory literature. Its thoroughness and clarity are practically without parallel. The authors restrict themselves to describing those aspects of the intuition of music listeners regarding the hierarchical aspects of tonal, homophonic music. The theory breaks musical structure down into four components: *grouping structure* which expresses a hierarchical segmentation of the piece into motives, phrases, and sections; *metrical structure* which expresses the intuition that the events of a piece are related to a regular alternation of strong and weak beats at a number of hierarchical levels; *time-span reductions* which assign to the pitches of a piece a hierarchy of "structural importance" with respect to their position in grouping and metrical structure; and *prolongational reductions*, which assign to the pitches a hierarchy that expresses harmonic and melodic tension and relaxation, continuity, and progression.

Despite the highly formal nature of GTTM, the authors point out two important reasons why it is incapable of providing a computable procedure for music analysis. One is the use of "preference rules", which designate out of the possible structural descriptions those that correspond to experienced listeners' hearings of a particular work. Preference rules do the major work in developing analyses within the theory. The rules are non-numerical, however; they are written in ordinary English without explicitly stating values for assigning preference ratings or segmentation thresholds. For example, Grouping Preference Rule (GPR) 3 states:

> Consider a sequence of four notes $n_1n_2n_3n_4$. All else being equal, the transition n_2-n_3 may be heard as a group boundary if
>
> a. (Register) the transition n_2-n_3 involves a greater intervallic distance than both n_1-n_2 and n_3-n_4. ... (p. 46)

Note that the rule does not indicate <u>how much</u> greater the intervallic distance needs to be to indicate a group boundary, nor does it explicitly state whether or in what manner the strength of the boundary is relative to the interval width. In defense of their position, the authors claim that "the choice of threshold values is to a certain extent arbitrary."

The second respect in which GTTM is not "computable" is related to its failure to characterize what happens when two preference rules come into conflict. The authors claim that "it would be foolish to attempt to quantify local rules of grouping without a far better understanding of how these rules interact with other rules whose effects are in many ways not comparable." (p.55) Clarke (1985) praises the "cognitive honesty and ecological validity" of this position, but also warns that

> ...it is more helpful to have analytical systems that adopt a stronger position and generate more restrictive interpretations. Rather than narrowing the gap between cognitive theories and formal theories, it may actually be more productive to maintain that clear separation and to develop each within its own sphere. Performance analysis, and the interaction between structure and expression in particular, offers an arena in which the clash between the two domains may be fruitfully investigated. (p. 235)

Regarding my own work, I have found that cognitive theories can often be extremely valuable in informing formal theories. Once the basic parameters have been established, a little arbitrariness goes a long way! In fact, it is precisely those parameters that must remain arbitrary in any cognitive theory which indicate the handles of control over the behavior or "personality" of its formal implementation.

2.6.3. Berry's Structural Functions

Noticeably lacking in both Tenny's and Lerdahl and Jackendoff's theories is an attempt to characterize the *function* of musical groups. Wallace Berry's *Structural Functions in Music* (1987) provides the missing link between structural segmentation and classification. Berry's observation:

> In music that is composed (as opposed to music of random operations or random consequences), actions (changes, events) involving various elements (lines of pitch change, tonal and harmonic succession, rhythm and meter, texture, and coloration) are so conceived and controlled that they *function* at hierarchically ordered levels *in processes by which intensities develop and decline*, and by which analogous feeling is induced (p. 4)

…is entirely consistent with Composer Joji Yuasa's comment (personal communication) that "composition is the act of structuring musical intensity." The strength of Berry's position has proved helpful indeed in providing a useful theoretical construct for the development of structural descriptions that may be effectively mapped to expressive output.

Berry posits four functional identities of musical impulses or events: *anticipatory*, *initiative*, *reactive*, and *conclusive*. While these categories undoubtedly represent an undersampling of all possible functional states one might ascribe to music, one is hard-pressed to add to the list without the new addition being seen as either a sub-category of one of the four states, with the possible exception of a "null" category, which would presumably represent an absence of change in intensity. Furthermore, these functions can be applied to the analysis of virtually any piece of music, over any parameter. For these reasons, I chose to adopt Berry's four functional categories in the formalisms described in section 3.1.

2.6.4. Honing and Desain

The Netherlands research team of Henkjan Honing and Peter Desain has recently provided some remarkable insights into the relationship of musical structure and expression. Their work is based on the hypothesis that "the complex expressive timing profiles found in musical performances are more readily explained as the product of a small number of simple rules linked to a relatively complex structure, rather than as the result of a large collection of interacting rules with hardly any structure" (1991, p. 43). For example, one of the most interesting aspects of the time-scaling methods outlined by Honing and Desain is their rejection of the "tempo curve" as a global function for controlling expressive timing. In their words:

> ...a simplistic notion of a tempo curve of a musical performance is a dangerous and harmful theoretical construct. ...In changing the overall tempo of a performance, by manipulating the tempo curve alone, all time intervals of equal length between two notes are scaled in the same way. But some notes may constitute a particular kind of ornamentation, whose duration should be more or less unaffected by tempo. As a result, the timing of the piece becomes unmusical. (1991)

The main thrust of the work of Honing and Desain is thus to develop more complete and accurate computational representations of music. For example, they point out that rests, rather than mere markers of empty space, should be thought to represent

musical events with zero amplitude, as "they are central e.g. in dealing with articulation - a short note followed by a rest behaves differently under transformation than a longer note played in a staccato way" (1991, p. 8). Honing and Desain have defined specific time-scaling procedures for each of four basic "musical objects" or structures, split into two categories: Multilateral objects which contain components of equal importance, and Colateral objects whose components maintain a dependency relation to one another. The two Multilateral structures are called *Successive* and *Parallel*, which basically relate to monodic melody and chords. Their procedures for the scaling of successive events is essentially no different from the "time maps" described in Jaffe (1985), in which the derivative of a tempo curve is taken in order to determine in advance the new time placement of each event. The scaling of parallel events deals with expressive asynchrony. The two Colateral structures are called *Appog*, for "appogiatura", and *Accia* for "acciaccatura". When these events are re-positioned in time, the dependent components may alter their relationship to the main component in an separate manner from the way the entire event was re-positioned.

Such relational categorizations cannot easily be made using a "flat" data representation (such as standard MIDI data), because the necessary conditions for grouping are impossible to infer from the data structure. We could easily distinguish between Successive vs. Parallel structures if we assume that all notes which have start times within a certain threshold distance from each other are to be taken as members of a single Parallel object. Colateral objects, on the other hand, are not so simple. In CPN, we identify such structures by the use of grace notes; the graced note being indicated by a connecting slur. This type of structure contains more information than standard MIDI events can represent. A virtual performance system utilizing MIDI must, therefore, be capable of subsuming the MIDI standard into a more complete contextual framework in which relationships between events and musical structure can be more fully specified. This is precisely the approach Honing and Desain have taken in their LISP-based POCO system, which has recently become available from the authors.

2.6.5. CPN as Input Model

The example of grace notes is just one of the many ways in which CPN serves as a model form of categorical data representation. Western composers and performers developed CPN over many centuries into an efficient, flexible, and powerful method for the encoding, editing, preservation, and transmission of musical ideas. In CPN, high-level discrete representations (as opposed to low-level, continuous data) provide an elegant

way to convey structural information which can be "fleshed out" differently in each successive performance.

CPN and other graphical pitch-time grid (e.g.: "piano roll") representations convey an overview of composed input with great efficiency. This allows the performer to visualize large-scale horizontal and vertical grouping patterns in the piece. For example, on a local level, the beaming of notes indicates to the performer how shorter patterns are to be grouped. Higher-level groupings are indicated by slurs in CPN. Still higher levels of musical organization are indicated by various types of barlines: a double bar, for example, to indicate a major sectional division. Braces and brackets in CPN show how several instruments may be grouped together vertically. These features of CPN are particularly important for their ability to encoding information which may affect the performance without disturbing the metrical placement of the notes. The preservation of metrical information allows the performer to continue using beat location to influence other musical parameters, and may be critical in a multi-instrumental piece if synchronization is desired. The same can be said of articulation marks such as *staccato*: they inform the player about the intended duration of the note without disturbing its metrical value. In addition, the composer may add instructions for realizing expressive details without disturbing the categorical information. It is then up to the performer (cybernetic or human) to processes this data, converting the high-level representation into the appropriate low-level code for sound production.

2.6.6. Performance as Input

Flexible and powerful input descriptions are necessary not only for scores entered by a composer outside of real-time, but also as a basis for real-time interaction between human and cybernetic performers. In real-time situations, however, more of the procedural and grouping information must be extracted by feature analysis, as human players cannot be expected to signal their long-term intentions explicitly during an improvised performance. Furthermore, the incorporation of feedback into the performance process is impossible without some means of capturing and analyzing the output of the system.

Technical methods for obtaining accurate performance data have reached a fairly advanced level. MIDI-equipped instruments (such as the Yamaha Disklavier™ piano and the WX7 wind controller) can provide pitch, velocity, and continuous controller information at a reasonably high bandwidth. Signal processing techniques for real-time frequency, amplitude, and spectral analysis can provide fairly accurate data on pitch,

loudness, timing, and timbre of performed input. These techniques have been implemented both in software and as stand-alone hardware devices (e.g.: the IVL Pitchrider, the IRCAM/Ariel workstation, the CNMAT DSP objects for Opcode's Macintosh/MAX environment, etc.).

Feature analysis of real-time input is important for the automatic extraction of necessary information not supplied by a composer. Examples of useful information might be phrase boundaries, registral contours, rhythmic patterns, etc.. Unfortunately, research in this area is quite uncommon. Perhaps the best example of a real-time analysis procedure for music was presented to me by George Lewis during a seminar on interactive computer music systems at UCSD (Spring 1992). Lewis referred to the procedure as a "leaky integrator". This procedure essentially samples incoming data at a fixed rate and moves partway toward new target points as they arrive. If no new data arrives, the integrator will eventually "catch up" with the last input value. The integrator thus provides two important values: the continuous "average" of the incoming data, and a "jag" factor, which is the difference between the current average and the new input value. High jag values indicate large, sudden changes which can often indicate event or phrase boundaries.

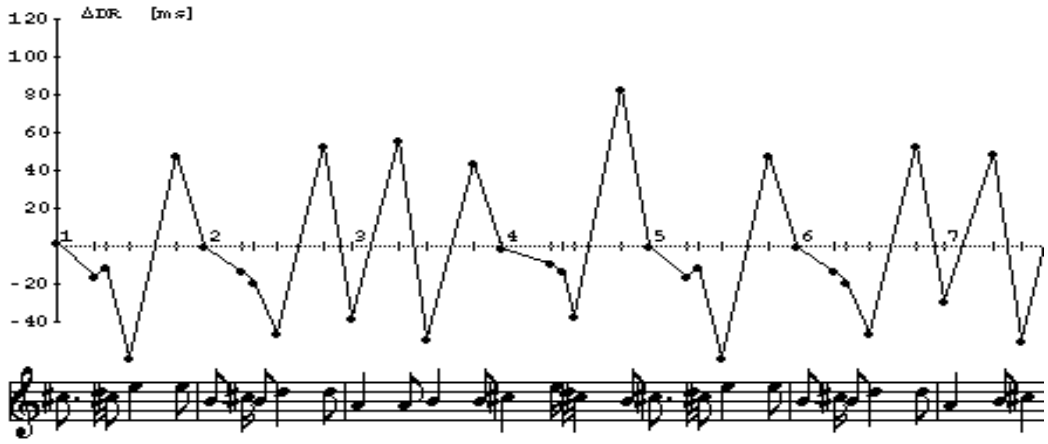
2.7. A Comparison of two Computer-based Music Performance Systems

By far the most audible and visible evidence of activity in expressive performance by computer has been produced by two research teams centered around two very well-known and almost diametrically opposed figures in the field of music performance research: Johan Sundberg and Manfred Clynes. In chosing the methodology of analysis-by-synthesis to test the validity of their ideas, both Sundberg and Clynes have provided researchers with an in-depth look at the clash between cognitive and formalized theory. A comparison of Sundberg's and Clynes' research methods and results shows that although Sundberg's experimental methods are much more scientifically sound, Clynes seems to have had the greater success thus far in obtaining a convergence between performance and synthesis. This leads to the conclusion that if one's goal is obtaining such a convergence, it may be more fruitful to trust one's musical instincts rather than proceeding with caution along the narrow path of systematic reasoning.

2.7.1. The Sundberg Rulle *system*

The system for generating expressive performance (best described in Friberg, Frydén, Bodin, and Sundberg (1991)) represents the most completely described and

easily “testable” set of expressive formalisms I have encountered. The software is available free of charge from the authors and runs on a Macintosh II under Allegro Common Lisp. “Flat” scores in a special text file format (which includes phrase boundary markings and harmonic analysis data) comprise the input to the program. The input values are processed according to a rather large set of rules (approximately twenty) and expressive deviations in timing, amplitude, and pitch are calculated. The program can then play the expressively-altered score directly via MIDI, write the data to a standard MIDI file, and display graphs showing the expressive deviations above the note list in CPN. Below is an example of such output, illustrating changes in inter-onset durations generated by the program. The tune is the well-known Adagio theme from Mozart’s K. 331:



The *Rulle* system is rooted in Sundberg’s interest in the simulation of human speech. Attempts to mechanically synthesize human speech date back as far as 1779 (Flanagan, 1972). Spurred by the potential commercial value of “talking computers”, research in text-to-speech conversion has recently become especially extensive in recent years since the advent of cheap digital signal processing hardware. Taking advantage of the numerous parallels between speech and music performance, Sundberg’s research team has looked to the standard techniques of speech synthesis for examples of the formulation of expressive output “motor commands” (or synthesis parameters) as a function of categorical structure. In Carlson, Friberg, Fryden, Granstrom, and Sundberg (1989), the authors point out that

> Both speech and music are examples of formalized inter-human communication by means of acoustic signals. Both must be devised for the same perceptual and cognitive systems. The limitations and capabilities of these systems must contribute importantly to the development of both speech and music. (p. 403)

To briefly summarize their specific observations:
-Both music performance and speech synthesis systems convert discrete representations into continuous acoustic waveforms.
-The scope of accurate acoustic realization of both texts and scores is constrained within definite limits.
-*Hierarchical structure* is evident from the graphic representations of both music and speech.
-*Final phrase lengthening* is an important component of both. (Analogy is also made here to human locomotion; e.g.: the slowing of a runner coming to a stop.)
-*Category Separation* is used to stress or emphasize important differences. (e.g.: In music, minor seconds are played narrower, and major seconds wider than their nominal value; in speech, contrasts are exaggerated when there is a contextual need to distinguish two similar words, like Emigrant/Immigrant.)
-*Predictability* of notes in music and words in speech plays an important role in the performance and intelligibility of both.

Techniques developed for speech synthesis (Flanagan, 1968) generally revolve around the "production system" model of artificial intelligence (Longley and Shain, 1989): a collection of speech rules form a static database in production format (i.e. IF condition THEN action); the database also holds certain tables (e.g.: formant peak frequencies for vowel production). Text information is stored in a dynamic database and updated during program operation. The production system has been adopted as the basic model for several music performance programs, including those described in Sundberg, Fryberg and Friden (1991), Johnson (1991), and Katayose and Inokuchi (1989).

Attempts to formalize musical knowledge in the form of production rules are certainly nothing new: Fux' *Gradus ad Parnassum* is a well-known example; the treatise by Lussy mentioned previously certainly qualifies as well. The rules developed by Sundberg's team are much fewer, more condensed and much more formal than those found in such pedagogical treatises. This compactness does not come without sacrifice, however. There is, for example, no representation of metrical information in the system. There are also no rules for scaling chord asynchrony, an important component of expression as indicated by Palmer (1989). Furthermore, although the user is allowed to change the rule "amplitudes" individually (by multiplying the output of each rule by a constant *K*), there is no "envelope control" on these amplitudes. In other words, the rules

must be applied with the same force throughout an entire piece. The data structures used as input are much richer, in general on the note-to-note level of musical structure than on higher levels such as the phrase or section. This probably accounts for the fact that the rule effects are rather subtle, as well as that they seem to possess a fairly narrow window of functionality, beyond which the deviations quickly become unmusical.

Despite the authors efforts to describe the functions of the various rules as either *grouping* or *segregating*, Honing (personal communication) has criticized the Rulle system (and expert-systems in general) for the lack of control over how rules interact. Indeed, the assumption that rule effects can be simply added together without regard to context disregards the fact that in speech as well as music performance, rules which operate on lower levels may be overridden when higher-level rules are invoked. This situation comes about when ambiguities at low levels are resolved by higher-level context. In English speech, for example the alphabetical sequence *r-e-c-o-r-d* has two possible pronunciations resulting in different meanings depending on which syllable is stressed. In the sentence

May I borrow that **re**cord*?*

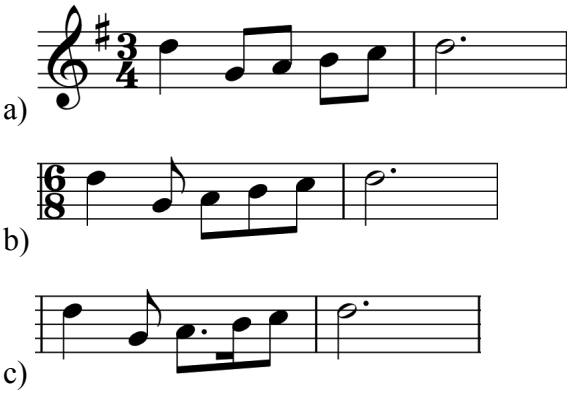
the stress would be on the first syllable, while in the sentence

Did you re**cord** *the concert?*

the stress is placed on the second syllable. As context resolves ambiguities of stress placement in speech, stress ambiguities in music are generally resolved by metrical context (Longuet-Higgins and Lee, 1983) but the Rulle system does not make any use of metrical information. This can be problematic in cases like that presented in figure 4. In example a, the 3/4 meter would lead a performer to stress the G on the second beat more strongly than the A which follows it. The Rulle system, however, would tend to favor a 6/8 interpretation by applying its "melodic charge" rule, which ranks the tones of the chromatic scale according to their "remarkableness", which is basically relative to the distance along the circle of fifths from the tonic. Thus, the second scale degree is assigned a higher charge than the tonic, and would therefore be played louder. The Rulle system would not support a 3/4 interpretation unless the passage were re-written as in example c, where the lengthening of the A and shortening of the B's duration would invoke rule DDC 1B: "the shorter, the softer", as well as the accent rule DDC 2A: "double duration". This type of interaction between performance practice and composition is one of the more interesting side-effects of working with such systems: the composer's knowledge of the behavior of the system helps to guide the composition process.

In addition to the absence of metrical representation, this example illustrates another common criticism of the Rulle system: the lack of control over rule *interactions*. One way to avoid messy situations like this might be to establish methods by which rules are prioritized according to their contribution to the formation or segregation of groups. For instance, in example c, the change of duration the quarter note (D) to the eighth-note run (G-A-B-C) might be considered a more *formative* parameter (as in Tenney) because it represents a Gestalt formed by the ascending line; therefore, the differentiation rules based on duration (DDC) should take priority over those based on scale degree. This type of judgment requires greater contextual analysis than the Rulle system currently provides.

fig. 4. metrical ambiguity



Although some of the rules have been substantiated by experimental evidence, it has been my experience that most rules have a very narrow margin of effectiveness; increasing values of K do not yield increasingly "expressive" performance, but simply awkward-sounding music! This seems to indicate that the production rules may be too narrow in focus and not very well-integrated with analysis (e.g.: phrase marking only introduces pauses of a constant length). The most effectively scaled rules were Melodic and Harmonic Charge, which supports the notion that rules which effect more than one parameter result in a more unified and malleable performance.

The fragility of the *Rulle* system may be inherent in the nature of expert systems. With such a large number of rules (and these are but a fraction of those which could be formalized along similar lines), the behavior of the system becomes impossible to manage. When interactions between the rules are not controlled, one rule may undo what another rule does. The historical failure of production systems to provide flexible and manageable paradigms for simulating musical skills by computer (Rothgeb, 1980) has

lead me to explore more Platonic, unified, hierarchically structured forms of knowledge representation.

2.7.2. Clynes' performance program

Manfred Clynes' work is well known to those studying the problem of expressive performance by computer. Based at the Research Center of the New South Wales State Conservatorium of Music in Sydney, Australia, Clynes has published numerous articles and books which describe his nearly three decades of research in this area. The following description and evaluation is based on my reading of these, and in particular, on the listening examples provided in Clynes (1983) and (1987).

Clynes' approach to musicality is based on Platonic notions of ideal forms of motion toward which great artists presumably aspire. This approach often manifests itself in theoretical statements which have earned Clynes a reputation as somewhat of an eccentric. For example, Clynes claims to have isolated a small set of "essentic curves" from experiments using touch-sensitive pads to measure subjects' physical responses when asked to express certain emotions (Clynes, 1977). These shapes are said to represent human emotional states such as love, hate, anger, sexual desire, etc.. Furthermore, Clynes firmly believes that there exists, for every "great" composer, a single ideal *pulse* (i.e.: a timing/amplitude vector prescribing a certain amount of deviation for each beat) which defines his essential character. In his numerous publications, Clynes shows examples of his own research verifying his theories.

Unfortunately, the inability of independent researchers to satisfactorily reproduce Clynes' experimental results on essentic curves has ostracized him from most of the intellectual community. The overabundant self-referentiality of Clynes' writings does tend to cast great suspicion on the validity of his work. Furthermore, Clynes focus is limited to Western music of the classical period; he justifies the lack of universality in his writings by claiming that Western classical music uniquely carries the quality of the "inner pulse" - a claim which any ethnomusicologist would violently dispute. More importantly, a careful psycho-acoustical study by Bruno Repp (1969) shows a distinct lack of correlation between preference ratings of pulses for various composers.

What Clynes' critics generally fail to realize is the advantage gained by avoiding the pitfalls of the rule-based approach. Rather than relying on an patchwork representation of musical knowledge (in the form of production rules) to produce his desired effects, Clynes has constructed formalisms which directly produce the expressive changes he seeks to introduce. Clynes' belief that his curves represent emotional states

may rest on shaky theoretical foundations, but one cannot deny upon hearing the results of his experiments that his formalisms have substantial expressive power.

Clynes most important work is probably in the area of expressive amplitude. He has defined "predictive amplitude functions" (based on statistical "Beta functions") which are shaped according to melodic structure. The basic half-sine function is modified so that

1. Its peak is shifted forward in proportion to the slope of the pitch curve to the next tone if the next tone is higher in pitch.

2. Its peak is shifted backward in proportion to the slope of the pitch curve to the next tone if the next tone is lower in pitch. In support of this idea, Clynes writes:

> One may attempt to explain this function by considering the sense of muscular effort. In order for a note to ascend in pitch the vocal chords have to be tensed more; in order to throw an object high, the arm is prepared muscularly. In the present note, we sense the preparation for what is to follow; the leaning forward signifies an increased tension as the next note approaches. Conversely, if the next note drops, there is a relaxation beforehand; the present note leans backwards, diminishing in loudness sooner. Thus its function relates the melody to organic bodily functioning, to the way we sense and organize effort. (1984, p. 227)

Although Clynes' Beta functions yield a fairly limited vocabulary of envelope shapes, they provide a formal example of an expressive function simultaneously driven by musical structure and reflective of qualities of physical movement. As Sundberg himself has pointed out, "[t]here seems reason to believe that vocal behavior under the influence of emotions is the mere translation into the acoustical domain of general patterns of body movements". (1982)

In spite of all his detractors, Clynes' performance program in action (Clynes: 1983, 1987 - listening examples) does (to my ear) an excellent job of synthesizing a limited set of expressive musical qualities. Clynes' use of metrically-based timing/amplitude patterns is, after all, quite appropriate within the stylistic context of Western classical music. This may be an indication of the general importance of style as an input to performance systems, if not a vindication of the pulse matrix as a theoretical construct. Repp's study did in fact show that pulse-modified performances are preferred to the un-modified version, although this shows only that any expressive variation is better than none at all. A more thorough test would have compared Clynes' synthetic performances against performances using random variations in timing and amplitude, so as to eliminate this conclusion. In my opinion, however, the success of Clynes' performance system stems from the unified sound which results from the application of a

few simple theoretical constructs on many different levels. In concordance with the Repp study, my own listening experiences convince me more of the general saliency of the metrically-related timing and amplitude deviations than of the applicability of specific values to specific composers' music.

3. Some Formalisms for Generating and Manipulating Expression

The previous section defined the (rather enormous) scope of the problem and noted some contributions of previous research. In this section I will describe some formalisms I have employed in my explorations of expression. Through my work with exploratory microworlds, they are being constantly updated and revised. The following section thus represents a "snapshot" of both the theoretical underpinnings and the results of my programming explorations.

Thinking back to the three steps of expressive performance posited in the introduction, we can see how the application of these formalisms follows these steps. The *input* to the procedures consists of the list of values of one or more parameters of the score; the parameters used as input I shall call *dominant* parameters. The goal of the formalisms presented here is to append a *deviation parameter* to each note in the score by means of *analysis* of the input. The analysis procedures involved need not be strictly formal (e.g. the placement and categorization of phrase boundaries), but ideally they should involve a minimal amount of effort on the part of the composer. The idea is to empower the user of these formalisms so that meaningful changes in the expressive output can be made without excessive data entry or editing. A deviation parameter is a floating-point number between -1.0 and 1.0 representing the amount of expressive "intensity" returned by the analysis procedure. This deviation value can then be used to alter the values of any chosen parameters in performance according to an arbitrary *mapping* procedure. The parameters of tone production altered by the deviation values I shall call *subordinate* parameters. Generally speaking, dominant parameters are those written down by the composer, and subordinate parameters are handled by the performer and/or built into the instrument itself. The classifications of dominant and subordinate parameters are not mutually exclusive, however; these are merely useful terms in describing the interaction of structure and performance. The main practical difference between the two is in the way they are affected by the deviation values: subordinate parameters must have a pre-defined default value corresponding to a norm or median around which to deviate, whereas dominant parameters always deviate around their own nominal or notated value.

It is important to clarify the distinction between the types of formalisms presented here and those proposed by Sundberg and Clynes. Both Clynes and Sundberg have concentrated on prescribing specific techniques for generating sonic variation, whereas my formalisms focus on enriching the structural description with information necessary for performing expressive transformations. However, the relative paucity of structural

information in the Clynes and Sundberg systems suggests that the degree of "consciousness" the systems can model is rather low (see section 2.4). I agree with Honing and Desain's contention that it is the appropriate *representation* of musical structure which holds the key to effective use of the computer as a performance medium. The choice of mapping of that structural information into the sonic realm is then a matter of taste and compositional objective. Any combination of the techniques of sonic variation discussed in section 2.5, for example, would probably suffice to communicate the structural information to the listener; the computer-musician needs merely to choose which techniques to employ and to what degree. There are, of course, perceptual limits to the effectiveness of any variation technique; too small a variation will not be noticed, and too great a variation will destroy the categorical sense of the information presented. I will defer the discovery of these limits to future research, for which systems such as *Rulle* are excellent tools.

3.1. Intensity Curves

The application of deviation values to tone parameters is intended to lend varying degrees of *intensity* to a musical performance. Within these formalisms, musical *intensity* is defined as a percentage of deviation from the median or default values (in the case of subordinate parameters) or notated values (in the case of dominant parameters) of a musical event. We may then define a two-dimensional "intensity space" having time along the horizontal axis, and intensity along the vertical axis, extending above and below zero. By plotting points along time in this space, we can define an *intensity function*, which is a graph of musical intensity as a function of time. We can also use this definition to think in terms of the *amplitude* of an intensity function, which may be either positive or negative.

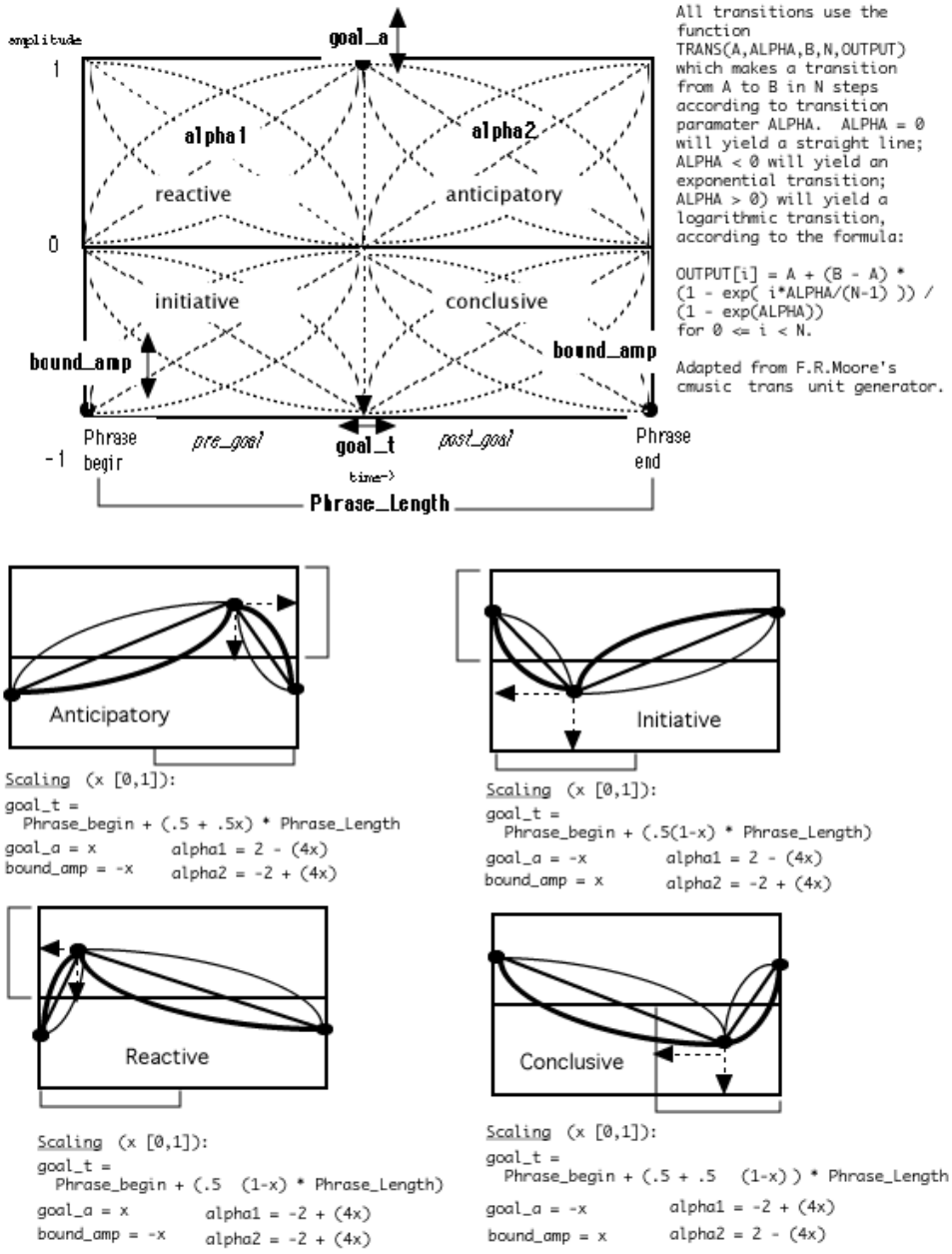
Research in expressive timing encouraged me to pursue the idea of automatically generating intensity curves by utilizing knowledge of phrase boundaries and by applying some kind of general description of the properties of the phrase. The strength of the theoretical foundations of Todd's model of rubato curves (see section 2.4.1) encouraged me to try some experiments using the parabolic equations given in the article to modify expressive timing and amplitude. Applying parabolic rubato curves to some musical phrases (both traditional melodies and randomly generated fragments) indicated that Todd's model was too limited in its restriction to one type of parabolic function in which the direction of the tempo curve was always falling toward a minimum point, then rising exponentially to a point commensurate with the boundary strength associated with the

phrase. Although the parabolic function in general made sense from a computational and musical standpoint, it seemed necessary to add to the model the possibility of exerting finer control over the direction, steepness, offset, and starting point of the parabola so that the tempo curve could be made to match the more variably shaped tempo curves found in actual human performances (see especially Palmer, 1989). It also seemed clear to me as a performer that an intensity curve might take on four general shapes, depending on whether the goal point of the parabola was in the antecedent or consequent half of the phrase, and whether the intensity was generally increasing or decreasing toward the goal. This observation led to the hypothesis that there might be a connection between these four possible parabolic shapes and the four structural functions introduced by Berry (see section 2.5.3). This happy coincidence resulted in the formalization of four *intensity curve* types: *anticipatory, initiative, reactive,* and *conclusive*. The properties of these curves are described below.

The intensity curve of each phrase type has two segments, one on either side of a goal point which is moved toward, reached, and moved away from. Dividing the intensity space of a musical phrase into four equal "quadrants", the position of the goal point of the intensity curve in a given quadrant becomes the determinant of the *functional identity* of the phrase. The goal point position and curvature are determined in this formalism by a single scaling factor *x*, a floating-point value in the range [0,1]. The scaling factor affects the amplitude of the goal point **goal_a** (plotted on the vertical axis) and its position in time relative to the phrase boundaries **goal_t** (plotted on the horizontal axis). The amplitude of the curve at phrase boundaries (**bound_amp**) is the inverse of the goal amplitude. The scaling factor *x* also affects the exponentiality or logarithmicity of the curve transitions.

Figure 5 presents the computational details of curve formation. A few items bear extra explication, however. The goal points of each curve type are scaled closer to the nearest phrase boundary as *x* increases, except for the Conclusive phrase type, in which the goal point is scaled further away from the end of the phrase. Also, in the implementations I have used to test this formalism, the domain of the function is actually the note number, rather than absolute time.

Fig. 5. Intensity Curves



In accordance with the widespread belief that music is hierarchically structured (Lerdahl & Jackendoff, 1983; Cooper & Meyer, 1960), I have included in the formalism the possibility of nesting multiple intensity curves inside one another. This nesting is predicated on the condition that structural descriptions at each level must be complete (without gaps); for example, the following is a legal description:

begin anticipatory .5 (level 2)
begin anticipatory .7 (level 1)

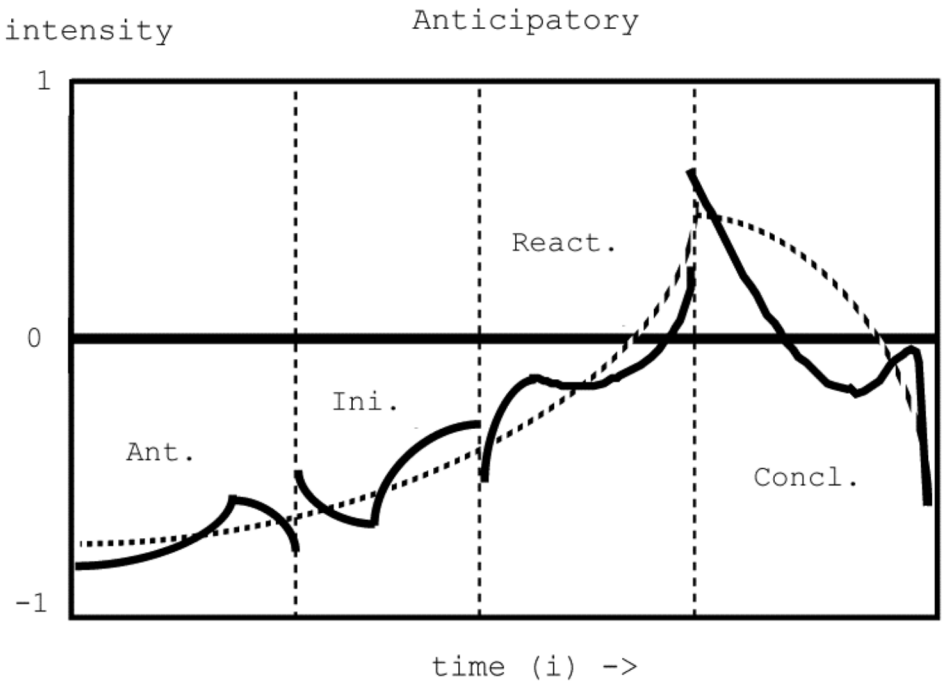
note
note
note
end anticipatory (level 1)
begin conclusive .2 (level 1)
note
note
note
end conclusive (level 1)
end anticipatory (level 2)

...while this is not:

begin ant .5 (level 2)
begin ant .7 (level 1)
note
note
note
end ant (level 1)
note
note
note
end ant (level 2)

The final function is defined by summing the curve values at all levels, as shown in figure 6. As the hierarchical level of the function increases, the ratio of curve amplitude levels is also increased (by a constant factor chosen by the experimenter) so that lower-level curves appear as deviations around the higher-level ones. The entire function is post-normalized to lie within the range [-1, 1].

Fig. 6: Nesting of Intensity Curves



In the computation of the intensity curves, boundary amplitudes are, by definition, always equal to each other (and to the inverse of the goal amplitude). In practice, however, it is often the case that for scaling values of *x* very close to 1.0, the boundary value near the goal point is never actually reached. This is due to the fact that the implementation of the curves as sampled functions forces a truncation or rounding when determining phrase lengths. When the length of a curve segment falls below the length of the first or last note occuring at that phrase boundary, the segment length falls to zero.

3.2. Pitchvoluration

Intensity curves provide a useful formalism for controlling deviations over relatively long stretches of time. For variations at the note level, I have developed the percept of "pitchvolurations" (a term I encountered in J.K. Randall's "Two Lectures to Scientists", 1972) into the formalism described here. The term derives from the fact that I have generally chosen pitch, volume, and duration as the dominant parameters for the analysis procedure; thus the deviation value would represent the weighted sum of the pitch, volume, and duration quantities of each note. The formalism itself does not demand

that any particular set of parameters be used as dominant parameters. Equation 2 shows the generalized mathematical representation of this value:

equation 2: Formula for "PitchVoluration"

$$x(n) = \sum_{i}^{N} \frac{V(i_n) - Vmin(i_s)}{Vmax(i_n) - Vmin(i_s)} W(i)$$

N: number of parameters

$V(i_n)$: value of parameter i at event n

$Vmax(i_S)$: maximum value of parameter i over section s

$Vmin(i_S)$: minimum value of parameter i over section s

W() is a normalization function containing a weighting value for each parameter i such that the sum of all weighting values is 1.

Equation 2 shows that for values $V(i_n)$ within the specified range, the output will be constrained between 0 and 1. This value can then be used to scale subordinate parameters of expressive control such as timing, vibrato rate, vibrato depth, brightness, attack time, etc. The formalism reflects local characteristics of the music by accounting for local parameter maxima and minima. As an example, say we have analyzed a hypothetical input phrase (containing MIDI data) and found it to have the following characteristics:

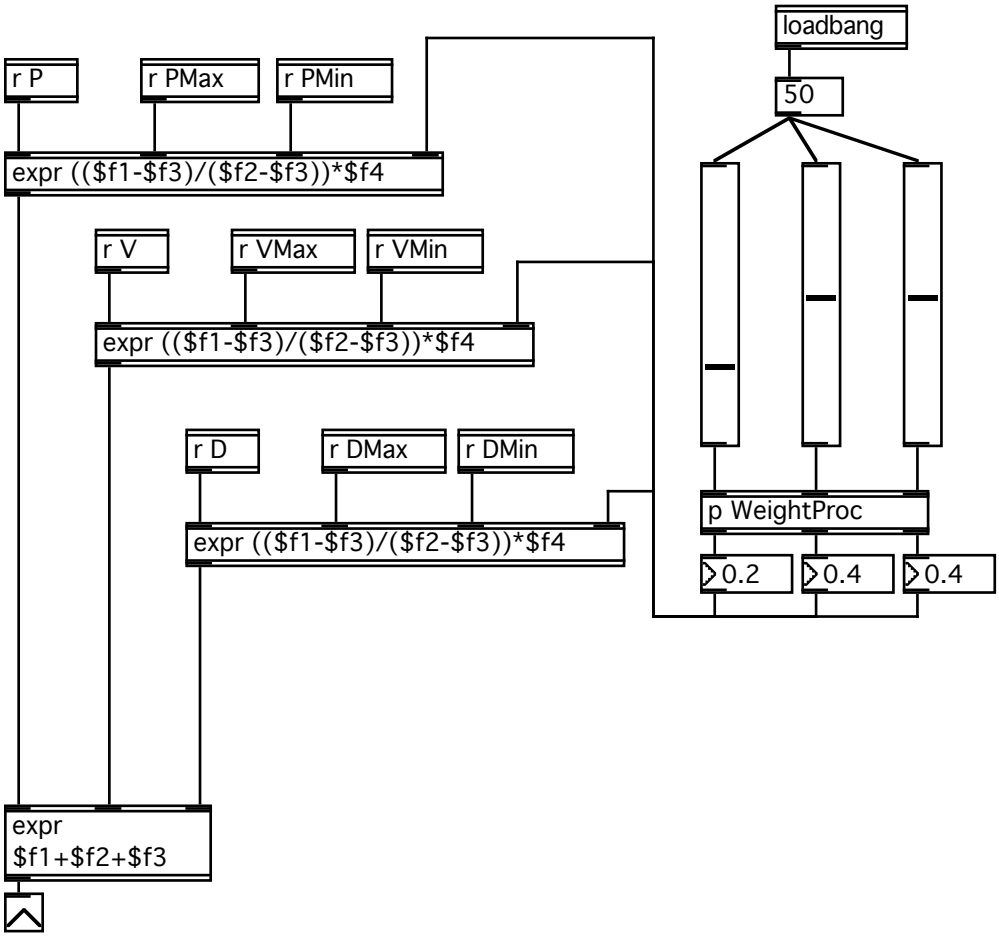
parameter	maximum	minimum
pitch	90	50
velocity	100	50
duration	1500	250

After assigning weights to each parameter to the analysis, we can evaluate each note in the phrase according to equation 2:

pitch (20%)	velocity (40%)	duration (40%)	pitchvoluration
50	100	250	.4
75	50	500	.2
90	88	1500	.9
...etc.			

Figure 7 shows an example of an implementation of the pitchvoluration construct within Opcode's Max™ programming environment (see section 4). In this dynamic "patch", the user can control the weighting of each parameter manually in real-time. Moving the three sliders on the right produces integers between 0 and 99. These numbers are passed through a procedure (WeightProc) which normalizes all weight values accordingly.

Fig. 7: pitchvolurations in Max:



Alternatively, these weights may be determined by further analysis of the score; for example, a measure of the entropy value (according to

$H = -\Sigma p(i) \log p(i)$ or some similar procedure) of a given parameter could be used to determine the appropriate weight assigned to that parameter over that section. This would reflect the notion proposed by musical advocates of communication theory that a parameter that varied more drastically over the course of a section would be conveying more information and should therefore be a stronger determinant of expressive deviation.

3.3. Markov Analysis

Shannon's communication theory may be applied even further to the production of expressive formalisms if we enlarge the window of analysis slightly through the application of "Markov chain" procedures. In such processes, a histogram of all transitions from one state to another is made on a chosen dominant parameter. The number of occurrences of each individual transition is divided by the sum of all transitions in the piece to yield a probability value for every state transition. These values can be said to reflect a measure of the uniqueness of each state transition in a piece of music. According to the laws of communication theory, the "uncertainty" of the transitions (considered now as "joint events") must be less than or equal to the sum of their individual uncertainties (Shannon and Weaver, 1949, p. 51).

The "Markov analysis" formalism simply implements the above procedure in the chosen domain in the manner just described, with an additional step of normalization of all probability values to lie between 0 and 1. The probability measurements are then mapped into the range [-1,1] and appended to a score to be used as deviation values. The implication of this procedure is that more unique transitions should be played with higher intensity. This assumption tested in the section that follows. One inherent problem with implementing this assumption within a formal music system, however, is that one is forced to assign the "uniqueness" value of a note transition to either the first or second of the two events which comprise the pair. In these experiments, I chose to place the value on the second note, based on the assumption that one could only register the change once it had occurred. Unfortunately, the placement of the Markov-analysis-derived values on the goal note prevents the analysis from having a predictive function. The very fact that within the formal constraints of my microworld experiments, I have had to make this choice at all points to an already-incomplete structural description of music in which note transitions should probably be considered somehow as separate entities from the note onset. Determination of the importance of such structural omissions will have to await further experimentation.

4. Microworld Experiments

For reasons stated in section 2.4, I have taken an analysis-by-synthesis approach in order to evaluate the validity and effectiveness of the formalisms just described. Two programming environments were used in the development of the prototypes: *MAX* by Opcode Inc. and Apple Computer Inc.'s *HyperCard*, with MIDI extensions by Ear Level Engineering's Nigel Redmon. MAX began its life as the "Patcher" control language developed by Miller Puckette at IRCAM for the 4X synthesizer (Puckette, 1988). The program was later implemented as an interactive, graphic, object-oriented programming environment for MIDI on the Macintosh. MAX is particularly well-suited to applications requiring real-time calculations based on user interaction. One of its most powerful features is the option of "encapsulating" a patch created by the user into an object. User-created patches behave identically to those provided by Opcode, which are written and compiled in Think C's object-oriented environment. With a significant increase in effort, users can create their own compiled objects in this manner, with a commensurate increase in computation speed.

HyperCard is a highly flexible object-oriented authoring environment for the Macintosh which was found to be better suited to applications which involved extensive manipulation of data arrays outside of real-time (such as calculation of intensity functions and Markov analyses). HyperCard's graphic capabilities enabled me to create visual representations of the expressive score deviations with simple drawing commands. Although both MAX and HyperCard have proved quite useful as prototyping tools, they are probably inappropriate environments for developing full-blown applications (given the current state of the platform technology) due to the high computational overhead of their programming interfaces.

4.1. Expressive Asynchrony

Palmer (1989) provides evidence for the existence of a systematic tendency for keyboard performers to de-synchronize chords played on the downbeat. This "rule of thumb" provided an example of a simple formalism which could lend a degree of expressivity to computer-performance. Listening example 6 contains a sample of output from a MAX patch I created (see figure 8) which generates a 3-voice block chord texture of pitches determined by a random walk process. When toggled on, the `metro` or "metronome" object begins sending out a "bang" (or trigger) message once every second. Each bang triggers the counter to step ahead one notch in a cycle from 0 to 3; adding 1 to the output of the counter, we thus have a beat pattern of 4/4 time at 60 beats per minute.

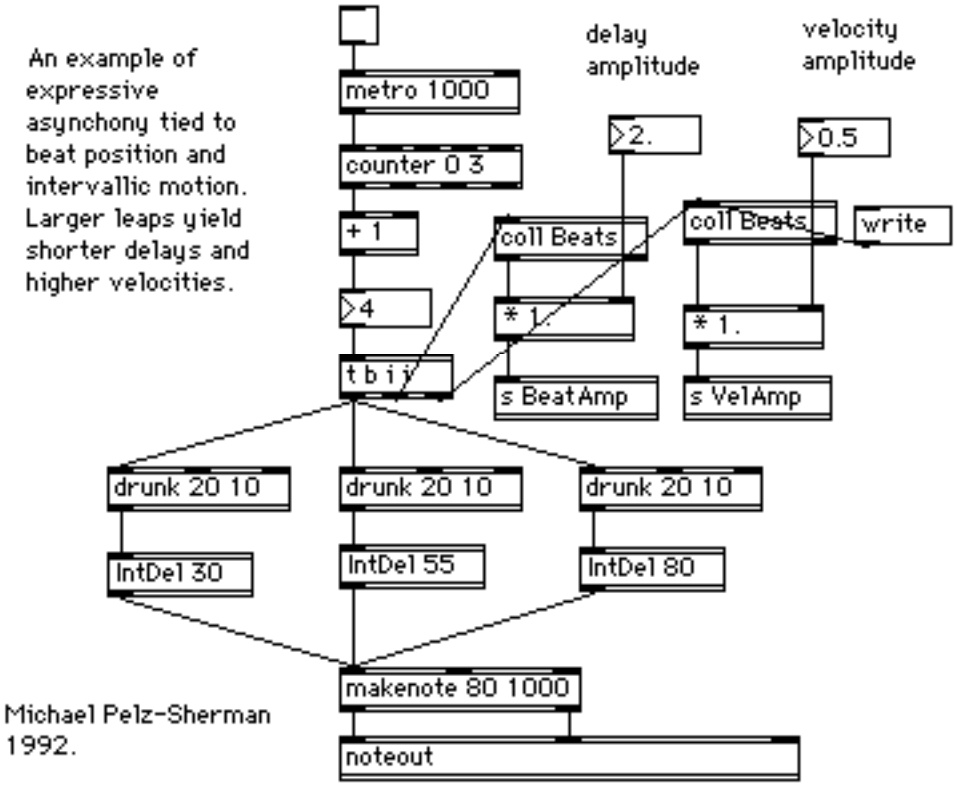
On each beat, a new pitch is calculated for each voice by the `drunk` object, which implements a random walk; the range of the random walk is set to 20 and the maximum step size to 10. The pitches are processed by the `IntDel` object before being sent through the MIDI interface. The `IntDel` object adds a timing delay to the randomly-generated note, the value of which is specified by its argument. The difference between the resulting pitch the previous resulting pitch is given by the `Interval` object. This difference is then used to compute both the amount of delay before playing the pitch and the velocity of the note; thus larger leaps are played both louder and sooner. Downbeat emphasis is acheived by table lookup. The `coll` (short for "collection") objects share a table of values corresponding to the 4 beats of the measure. The contents of the table are:

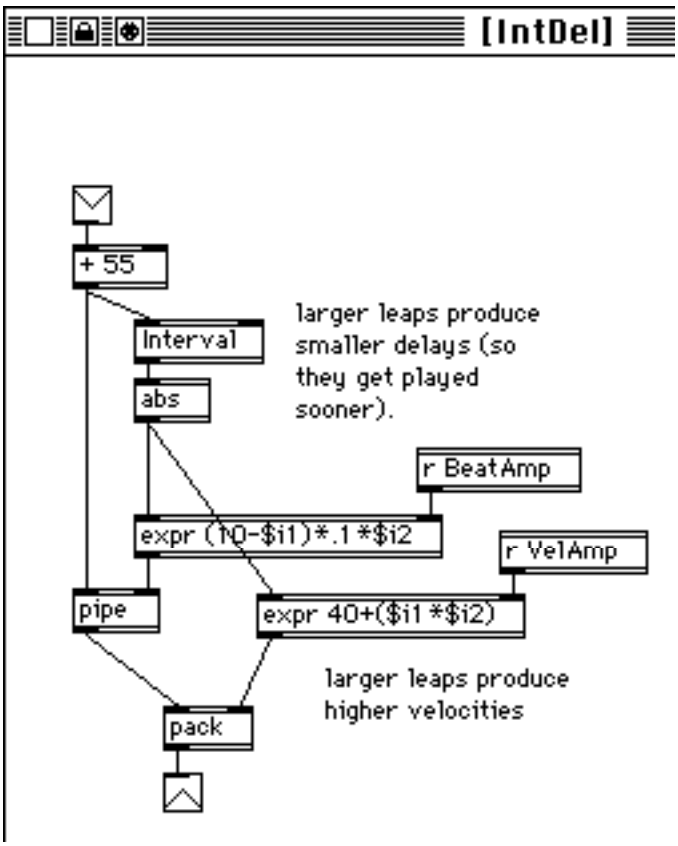
index	value
1	100
2	10
3	20
4	5

Thus, each of the three notes will have a delayed start time of *A**((10-*Interval*)/10)***100** milliseconds for beat 1, *A**((10-*Interval*)/10)***10** milliseconds for beat 2, and so on, *A* being an amplitude constant which is available to the experimenter to be adjusted in real-time. Velocity is scaled using the same constants; the extent of this affect is also controlled by an amplitude adjustment available to the user.

The patch just described is shown below in figure 8. Listening example 6 demonstrates the effect of the delay and velocity amplitudes shown. The most obvious effect of this process is to create a strong sense of 4/4 time; there is also a tendency for the ear to be drawn more strongly to the lines which are changing the most.

Fig. 8: Expressive Asynchrony patch





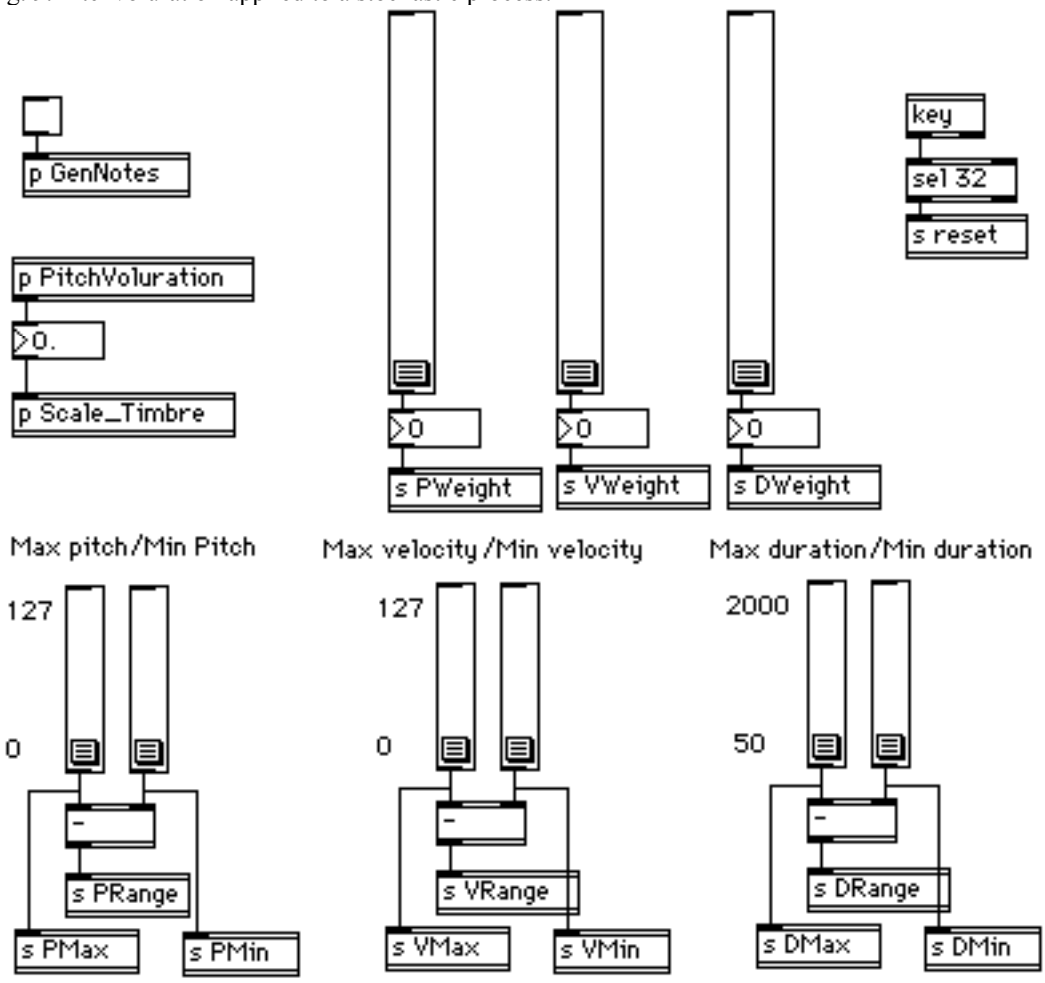
4.2. Pitchvoluration applied to a Stochastic Process

In section 3.2, I proposed the idea of applying weights to the pitchvoluration parameters based on the amount of variation the dominant parameters exhibit as a means of generating meaningful deviation values. Listening examples 7a through 7d contain samples of output from a MAX program I constructed which generates a monophonic string of pitch, velocity, and duration values according to a random walk procedure. For each note generated, a pitchvoluration value is calculated using the patch shown in figure 7. This value is used to scale the harmonicity ratio, the brightness, and the attack or "prefix" envelope characteristics of a synthetic timbre produced by a Yamaha TX7. Vibrato depth and rate are also scaled by the same pitchvoluration value.

Both the weighting values for each pitchvoluration and parameter the step size of the random walk for that parameter may be controlled in real-time by the sliders shown in figure 9. Thus, a correspondence is established between the amount of correlation in the signal of each dominant parameter and its importance in determining the subordinate timbral characteristics, which may be called the *dominance* of the parameter. Listening example 7 demonstrates the effect of focusing the dominance in a single parameter vs.

spreading it evenly through all parameters. The results point toward the idea that no single condition is very satisfying to the ear for very long. The complete MAX patch is shown below in figures 9 and 10.

Fig. 9. Pitchvoluration applied to a stochastic process.



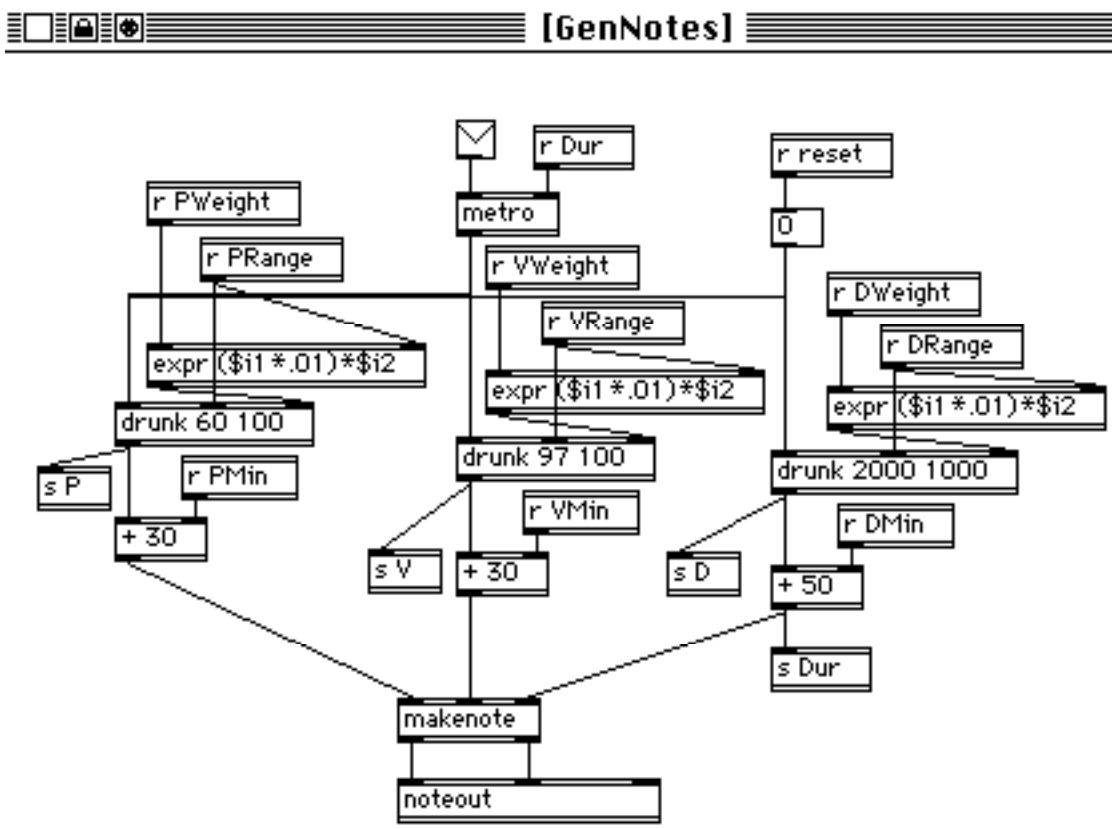
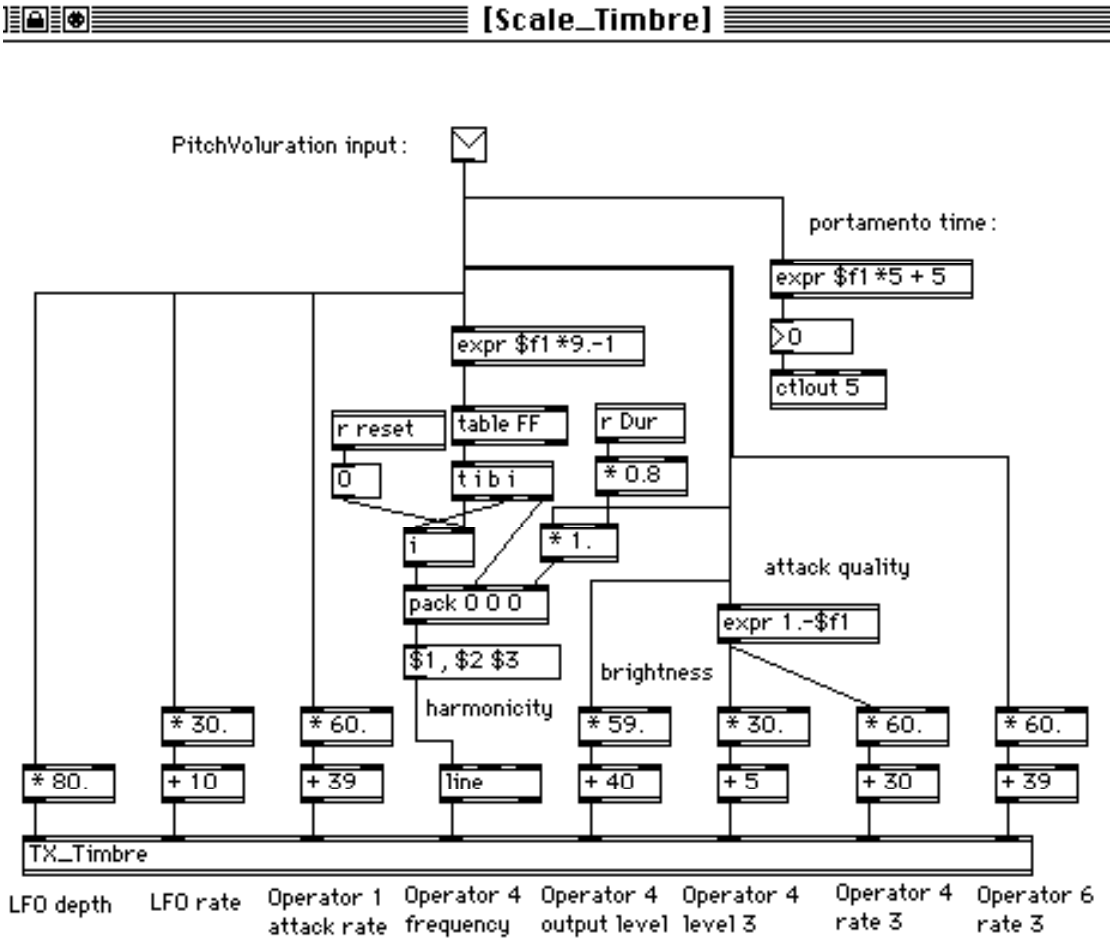

[GenNotes]
r Dur
r reset
r PWeight
metro
0
r PRange
r VWeight
r DWeight
r VRange
r DRange
expr ($i1*.01)*$i2
expr ($i1*.01)*$i2
expr ($i1*.01)*$i2
drunk 60 100
drunk 97 100
drunk 2000 1000
s P
r PMin
r VMin
r DMin
+ 30
s V
+ 30
s D
+ 50
s Dur
makenote
noteout

Fig. 10. Timbre Scaling in MAX



4.3. Musical Excerpts

Three diverse musical excerpts were used as test cases for the Intensity Curve and Markov analysis formalisms. In each case, a flat score was entered into a HyperCard stack (called "ScorePlot") which I devised for this purpose. Intensity curve boundaries, types, and scaling values were inserted into the score with values chosen experimentally to produce an appropriate intensity curve. A Markov analysis was then performed on the score, and deviation values were calculated and appended to the note list. Performances were subsequently synthesized using the deviation values generated from the application of the formalisms. Prior to calculation of the playback note list, the experimenter may enter six values to scale the maximum percentage of deviation from notated Inter-Onset Interval (ΔIOI), Duration (ΔDuration) and Velocity (ΔVelocity) to be produced by deviation values determined by the Intensity Curves (ival's) and probability values

computed from the Markov analysis (pval's). The deviations for a given parameter p were calculated according to

$$\Delta p = (p*ival*S(p_{ival})) + (p*pval*S(p_{pval}))$$

where S represents the scaling value for that parameter. Recall that since ival and pval both fluctuate between -1 and 1, the deviations in each parameter will be both negative and positive. It is also worth pointing out that all *subordinate* parameters in these examples are also *dominant*; therefore, the deviations are being made around the notated values, rather that around a pre-determined median value, as was the case with the timbre experiment in section 4.2.

A value to adjust the global playback tempo can be entered at the time of note list calculation. Note lists can be generated in either HyperMIDI or Csound formats. The procedure for the Csound format recognizes the envelope family types proposed in section 2.4.3. Both score formats recognize *crescendi* in the form **cresc start [dynamic]**...**cresc end [dynamic]**; in the Csound format, all dynamics are handled by a global instrument, so that crescendi can be made continuously over the course of several notes. CPN dynamic markings are converted to MIDI velocity values according to the following table:

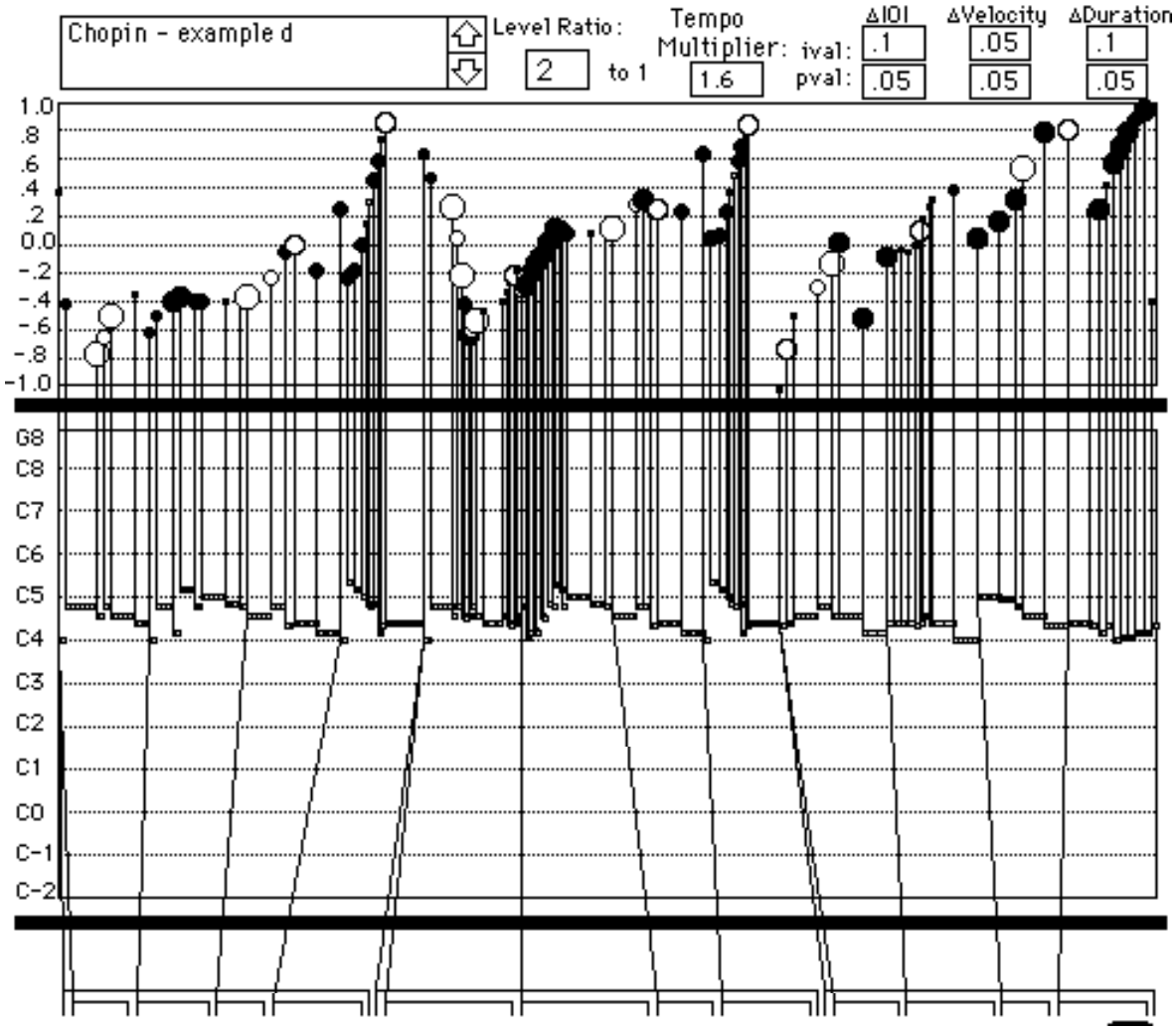
fff: 120
ff: 106
f: 88
mf: 72
mp: 60
p: 48
pp: 32
ppp: 16

In addition to performing the deviation parameter calculations mentioned previously, the ScorePlot draws a graph of the score in piano-roll form, displaying the intensity-curve-generated deviation values in the graph above, drawing the notated structure of the music below, and representing the Markov-analysis values (pval's) in the size and shade of the dot which marks the intensity curve value (positive pval's are black, negative white). Velocity is represented by the darkness of the pattern of the note rectangle. I have found this representation very helpful in developing an overview of these expressive deviations. For each musical excerpt presented on tape, I have presented

the ScorePlot representation below. The scores for each example can be found in Appendix A. Code listings (in HyperCard's HyperTalk scripting language) are found in Appendix B.

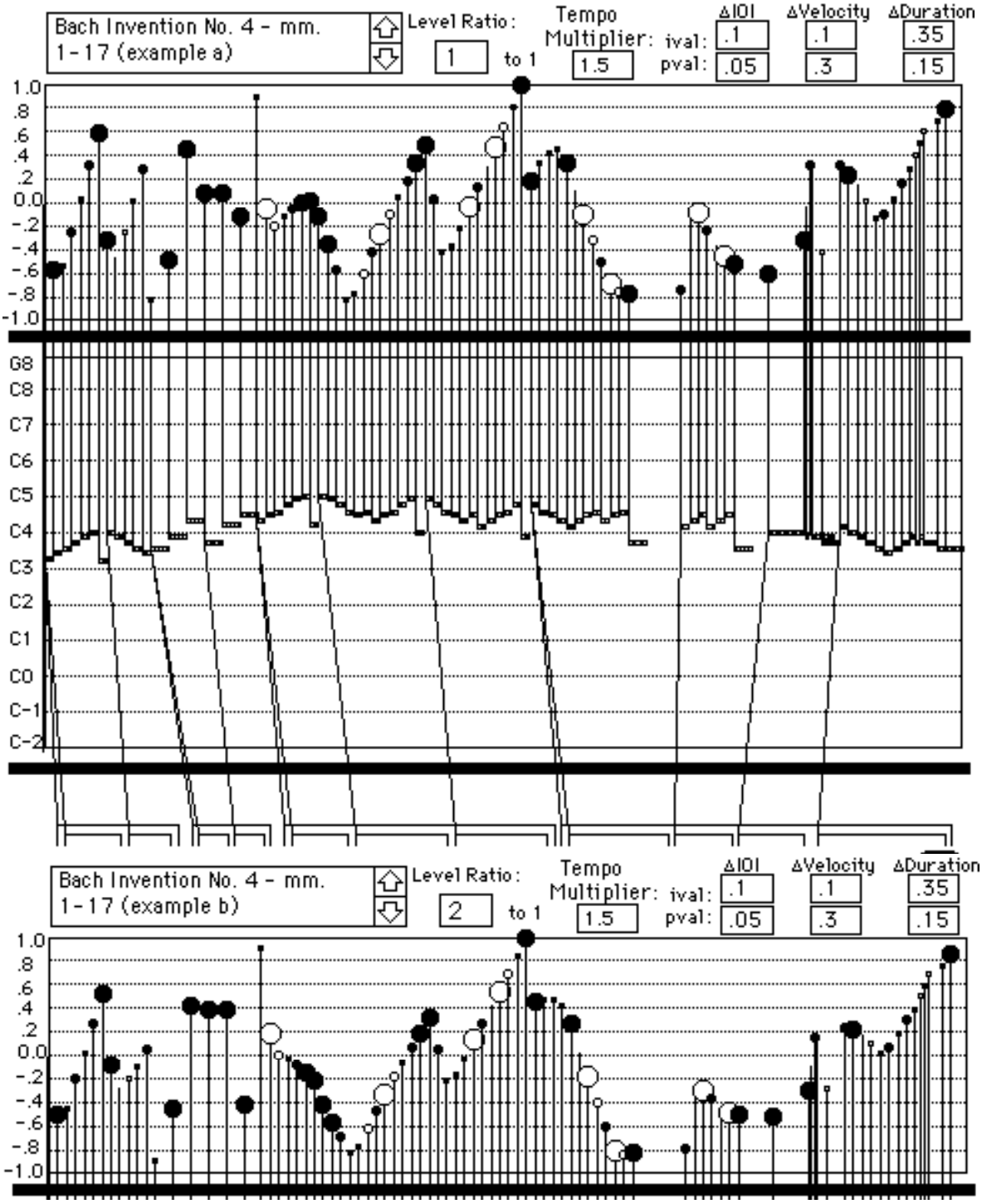
4.3.1. Chopin: Nocturne No. 2 in Eb (Listening examples 8a-8e)

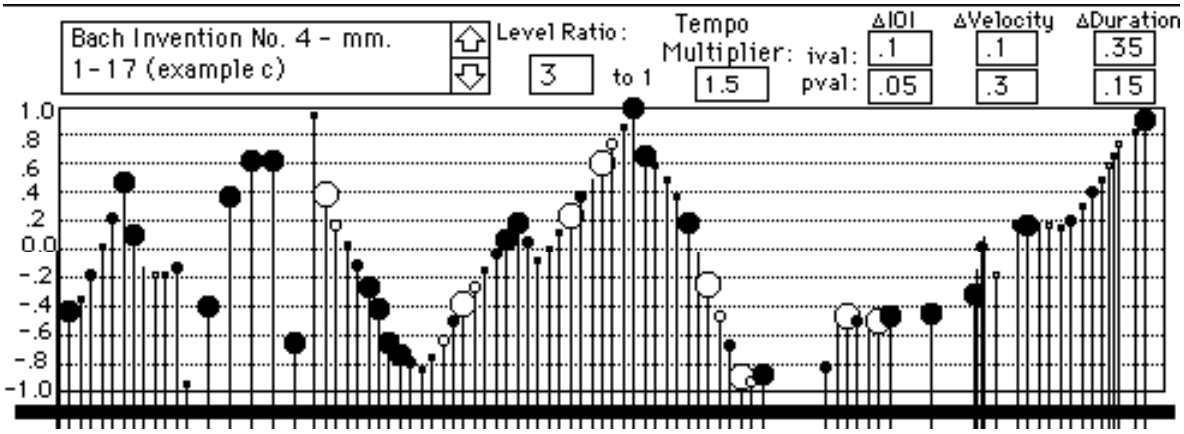
The rubato curves in this example reflect the heavy application of phrase-final lengthening characteristic of Romantic piano music. Five examples were synthesized demonstrating the effects of varying the "playback settings" ΔIOI, ΔVelocity, and ΔDuration. Thus, one should listen for increasing deviation amplitudes in tempo, loudness, and articulation. Grace notes were deliberately omitted from the score in this example because they need to be treated differently from regular notes (see section 2.6.4).



4.3.2. Bach: Invention No. 4 (Listening examples 9a-9c)

These examples demonstrate the subtle effects of changes to the Level Ratio parameter. As the Level Ratio increases, intensity curves on higher levels become more predominant; lower-level curves have less of an effect. The emphasis in playback *duration* as the primary subordinate parameter reflects the emphasis on articulation in Baroque performance practice. The grace notes were left in this example to demonstrate the harmful effect of "blind" tempo transformations on them.


Bach Invention No. 4 - mm. 1-17 (example a)
Level Ratio:
1
to 1
Tempo Multiplier:
1.5
ival:
pval:
ΔIOI
.1
.05
ΔVelocity
.1
.3
ΔDuration
.35
.15
1.0
.8
.6
.4
.2
0.0
-.2
-.4
-.6
-.8
-1.0
G8
C8
C7
C6
C5
C4
C3
C2
C1
C0
C-1
C-2
Bach Invention No. 4 - mm. 1-17 (example b)
Level Ratio:
2
to 1
Tempo Multiplier:
1.5
ival:
pval:
ΔIOI
.1
.05
ΔVelocity
.1
.3
ΔDuration
.35
.15
1.0
.8
.6
.4
.2
0.0
-.2
-.4
-.6
-.8
-1.0



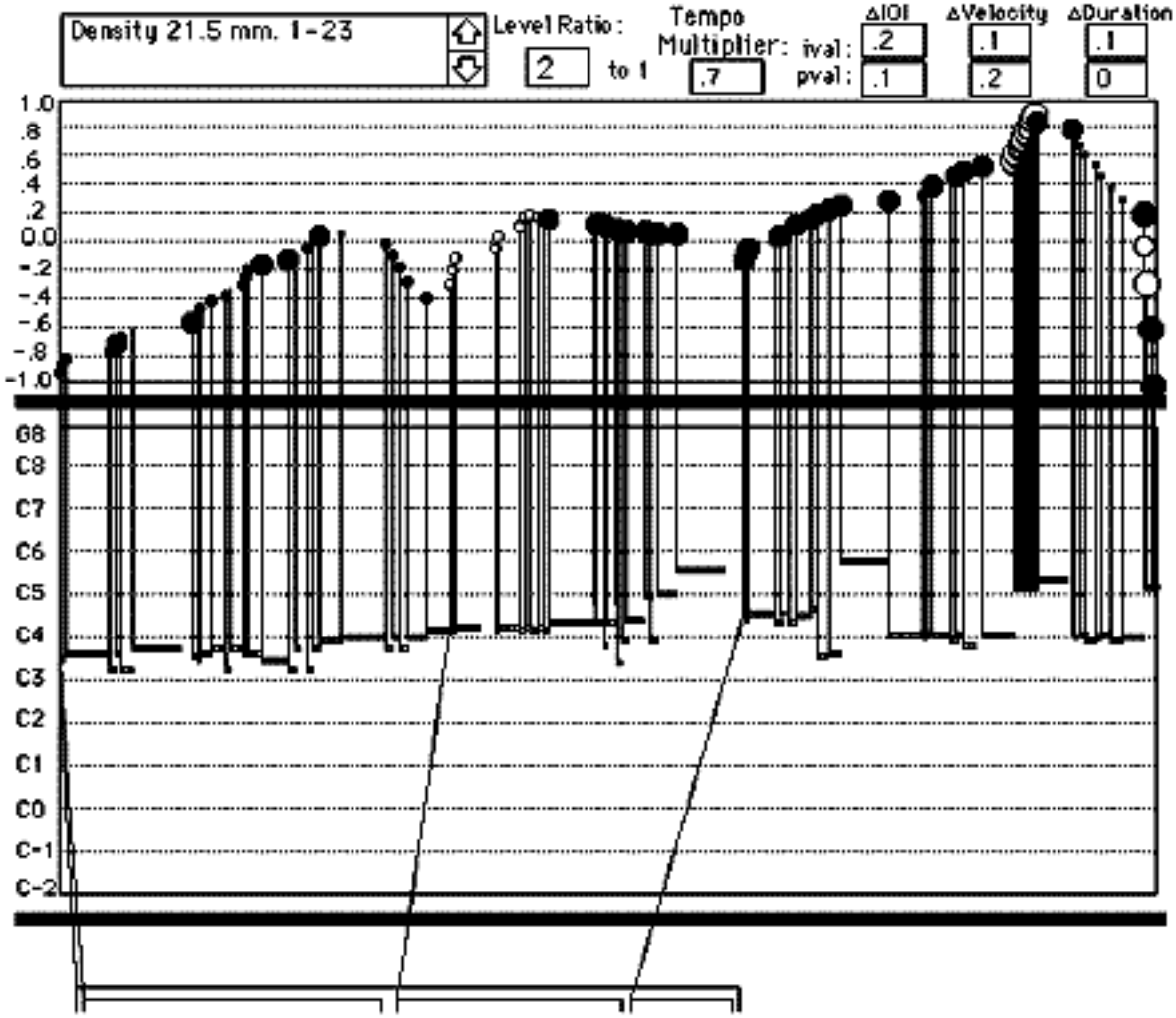
4.3.3. Varése: Density 25.1 (Listening examples 10a and 10b)

This excerpt was chosen primarily to demonstrate the implementation of some of the envelope families using software synthesis (MIT's Csound). It also demonstrates the potential for integration with automated analysis; the phrase boundaries chosen are taken directly from Tenney and Polansky's (1980) computer-derived analysis of this piece.

Listening example 10a contains a MIDI realization which, despite the realism of the flute timbre provided courtesy of the engineers at E-mu Systems Inc., illustrates a shortcoming of MIDI-based instruments in using continuous information to create variable amplitude envelopes under programmable control. Although continuous control message number 7 is universally recognized as a volume controller, only with some instruments is this volume "absolute", meaning the controller value overrides the current amplitude of the synthesizer. If this were the case, a MIDI volume value of 127 would always force maximum output from the synth, and a value of zero would produce no output. Unfortunately, on most MIDI synthesizers, MIDI volume is relative to note-on velocity and to the instrument's local amplitude envelope; notes played with higher velocities will sound louder than notes played with low velocities, even if both notes are placed under continuous control. While this is clearly desirable if one's objective is to use MIDI volume like a "fader" adjustment on a mixing console, it is not universally capable of generating true amplitude envelopes.

The more flexible environment of software synthesis provides the ability to create an instrument whose precise envelope parameters can be included with the parameters of each note. Further volume control can be provided by use of a global instrument which outputs a continuous volume control signal; when this instrument is "played", a conditional operator causes its amplitude value to override the local amplitude of the sounding instrument, allowing for crescendi and diminuendi to occur over the span of several notes, or to control more precisely the envelope of a single note. The great variety

of amplitude contours in the Csound realization of this score more than compensates, in my opinion, for the lack of timbral interest in the sine wave used as a carrier signal. On the other hand, the example suffers somewhat from the Csound implementation of legato, which causes notes to overlap. This trade-off is typical of the compromises one is often forced to make in exchange for the precision of software synthesis.



5. Discussion

5.1. Evaluation technique: perception as measurement

We have already seen how the formalisms and expressive techniques presented here are based on a subset of features of human performance. Now that some of these formalisms have been implemented, a number of questions arise: How important are the features of human performance not represented by the formalisms? Are we choosing appropriate sonic variation techniques to map the deviation values on to? Most importantly, does the output sound like it *could have been* performed by a living, (at least pseudo) intelligent entity?

To definitively answer these questions would require extensive psychological research. Balzano (1987) argues persuasively for the validity of "perception as measurement" in such research, advocating that the priority for defining musical phenomena be removed from physics and acoustics and "placed squarely in the ears and hands of musicians (p. 177)."

Hypothetically, one could subject these formalisms to a more rigorous version of our musical Turing test, in which the computer-generated expressive performances were compared against recorded human performances. A fair test could be devised only if both the human and machine performers used MIDI-triggerable instruments, as the discrepancy between live acoustic recordings and computer-generated sound would be a dead giveaway. However, the goal of this research is not to accurately simulate human performance, but simply to endow the computer with the faculty of making certain expressive decisions based on structural features of the input score. Many aspects of human performance arise from spontaneous decisions made while in the act of performing; still others (as pointed out in section 2) are gradually incorporated into a performer's individual style through the imitation, habit, and creativity. Failing to perfectly mimic human performance is not an indication of failure, just as speech synthesis by computer is not rendered useless by our ability to detect it from the real thing.

The most important criterion of the success of the formalisms is the degree to which they assist composers in generating and controlling the sense of a believable "presence" behind the performance of one's music. At this point I am partially satisfied that the formalisms have the potential to contribute significantly toward creating the

illusion of a performing intelligence. While the synthesized performances are certainly not ideal, they contain expressive elements which seem to be "plausible" outcomes of decisions based on the structure of the music. However, within the intentionally constrained scope of these microworld experiments, it has always been clear to me why the system is doing what it is doing. While this makes it easier for me to debug the programs, my foreknowledge of the processes involved makes it more difficult for me, as the designer and user of the system, to suspend my disbelief. For further verification, it will be necessary to obtain additional feedback from musically sensitive but "untainted" subjects, and with longer and more varied musical excerpts.

Informal presentation of these experiments to my colleagues in the Music Department has confirmed the above assessment. One colleague remarked that the tempo fluctuations in the Bach excerpt were too extensive in the middle of the phrase, but sounded fine at the end of the phrase. This may indicate a shortcoming of the intensity curve formalisms: the curves are constrained such that a large *rallentando* must be preceeded by an equally large *accellerando*. It may be necessary to re-evaluate the curve constraints based on this observation.

An important criterion of success, from a composer's perspective, is the effectiveness of the scaling procedures used; that is, whether the the amount and nature of expressive deviation can be effectively controlled by changing a small number of variables. To be effective, formalisms must exhibit the qualities of both *linearity* and *independence*. Linearity refers to the continuity of changes in output in respect to changes in the input. Small changes should not make the expressive deviations become suddenly unuseable. Independance refers to the ability to affect one paramater without changing the others. The distinction between dominant and subordinate parameters, together with the practice of appending deviation parameters to the score which are subsequently arbitrarily mapped to expressive techniques, makes a high degree of independance possible.

To my view, the scaling of the formalisms presented here meet the effectiveness criterion quite well; in particular, they are more effective than the scaling procedures used in the *Rulle* system (see section 2.6.). The rather narrow range of acceptable constants in *Rulle* is most likely a consequence of the large number of rule interactions in the system. With a small number of simple processes connected to a rich structural description, the expressive deviations are easier to predict and control. Listening examples 8a-e demonstrate the effects of changing the playback parameter settings, while examples 9a-c demonstrate the effect of varying the level ratios of the intensity curves.

The main deficiency in the formalisms rests in the under-representation of many important musical features. Among the most blatant omissions are grace-note data structures. Because grace notes do not behave identically to regular notes under expressive transformation, I omitted them from the Chopin excerpt (example 8). I left the grace notes in the Bach example to demonstrate the damaging effects of performing tempo-curve transformations on grace notes. The addition of metrical and harmonic information would no doubt make further improvements on the saliency of the expressive output.

5.2. Technical Issues

5.2.1. Using MIDI: the "black box" factor

The advent of the Max programming environment provided an excellent prototyping environment for this work. In addition to increasing the speed and efficiency of programming through its graphic, object-oriented interface, MAX enabled me to extend the synthesis control capabilities of MIDI through system-exclusive and continuous-control messages generated in real time. However, this approach is still subject to severe hardware limitations. Performance was noticeably slower on the Macintosh II (68020 CPU) than on the Mac IIci (68030), particularly in the calculation of vibrato waveforms. This made me doubt whether the timing values I had established were being realized accurately by the system. Response to system-exclusive commands was best on the Yamaha TX802; both the earlier TX7 and the more recent TG77 tone modules exhibited unacceptably sluggish response to the sys-ex commands.

The black box factor has its benefits as well. One very significant advantage of using a well-engineered MIDI-triggerable synthesizer is not having to worry about common software-synthesis annoyances such as waveform discontinuities (clicks), exceeding the limits of the binary representation system (clipping), and frequency "foldover" or aliasing. Commercial digital synthesizers are designed to eliminate such problems, although it is seldom made clear in the owner's manual just how this is done. The phenomenon of *legato* articulation in monophonic instruments, for example, is one of the most important and yet difficult-to-synthesize expressive techniques, for the goal in the production of *legato* is to connect the attacks of each new note within a single phrase group to the release of the previous note in such a way that the individual attacks are not heard. In terms of digital sound synthesis, this necessitates a special handling of legato notes for monophonic instruments in order to avoid discontinuities (e.g.: the concatenation of pitches into a single continuous frequency transition function, as well as

the connection of the phase of all digital oscillators involved). Thankfully, most commercial synthesizers (e.g.: the Yamaha ‘X series) have a built-in “mono” mode which causes envelope re-initialization to be skipped when a new note-on message precedes the note-off message of the previous note; pitch changes at sustained amplitudes generally do not create problems with discontinuities.

5.2.2. Software Synthesis: the responsiveness barrier

Software synthesis techniques have proved to be rather unwieldy for the exploratory aspects of this work, due to the unacceptably large amounts of time spent waiting for soundfiles to be calculated. Since the “fine-tuning” of expressive mapping processes must be carried out experimentally (i.e.: “by ear”), fast response from the system is very desireable. Nevertheless, I remain certain that the ultimate realization of the virtual performer model (especially in the areas of timbre and pitch control) will require the power and flexibility of software synthesis. The implementation of envelope families, for example, is rather difficult with MIDI-based systems due to the fact that most MIDI synthesizers used fixed, rather than relative, envelope rates. Timbre changes via system-exclusive and/or continuous controller messages tend to overload the MIDI data stream; synthesizer manufacturers to date have not generally engineered their instruments to respond gracefully to such nuances.

The ScorePlot program used in this research provides an example of a possible solution to this dilemma, as the program accepts standard MIDI files as input and outputs a software synthesis score. The notion of a system in which MIDI can serve as an preliminary representation format to allow ease of data-entry and editing of a score is certainly not entirely unique: the PatchWork system under development at IRCAM is another, far more ambitious example. Although I have not yet personally composed a complete musical work with a MIDI-to-software synthesis approach, initial musical results of work with the ScorePlot program are most promising. On the other hand, the rapidly falling cost and rising speed of computer CPU’s could eliminate the responsiveness barrier within the next few years, at which point a translation between MIDI and software synthesis will become unnecessary; we will simply need more general and powerful editing environments for computer music scores, in which the formalisms presented here may play a useful role.

5.4. Problems of de-bugging and the musical uncertainty principle

This study has employed adductive reasoning, which allows us to either accept or reject a hypothesis based on the success or failure of its implementation. It is often difficult, however, to be absolutely sure that the implementation faithfully represents the hypothesis. Difficulties I have encountered in the debugging phase of programming the prototypes are related to both the "black box" factor of using MAX to trigger commercial synthesizers with MIDI messages, and to the "Medusa data" syndrome. Regarding the former: there is a non-linear inverse relationship between the complexity of a MAX algorithm and the accuracy of scheduler timing. This phenomenon is most noticeable when porting a MAX patch between machines with different CPU speeds. Theoretically, scheduler accuracy should not be affected by CPU clock rates; however, there was a noticeable difference in timing characteristics between the 68020-based Macintosh II platform and the 68030-based Macintosh IIci.

The instrument-specific nature of MIDI parameter changes contribute further to the non-objectivity of the results. This problem was most acute in formally evaluating the timbre mapping procedures in the experiments so that the results could be applied in more general-purpose synthesis systems. Without extensive research or access to engineering specifications, it is impossible to know precisely how, for example, changes in velocity affect oscillator amplitude, how envelope breakpoints are scaled, how many LFO cycles-per-second "21" represents, etc.

Checking to see that correct values are being generated for expressive parameters brings us back to the "Medusa data" problem. Printing out a list of the variables used in generating expressive behavior at each stage of calculation results in an enormous data explosion. Furthermore, such data would be of limited value, because it is impossible to evaluate the perceptual effects of synthesis parameter changes without auditioning the result. Once programs reach sufficient complexity to begin to model human musical behavior, it becomes extremely difficult to trace the cause-and-effect relationships between input and output. While this may be a sign that things are going well, it's also frustrating if one is attempting to evaluate the individual contribution of a given aspect of the system. The *Rulle* system attempts to tackle this problem by effectively putting a "gain" knob on every rule in the system and evaluating the effects of the rules one by one. I have found, however, that it is usually impossible to describe the effects of processes in isolation as "musical"; one can verify their individual perceptibility, but not their effectiveness in generating expression.

This situation is similar to Heisenberg's general uncertainty principle, which states that the more accurately we measure something, the less accurately we can

perceive it, and vice-versa. In studying musical performance using formal, generative procedures (analysis-by-synthesis), it seems to be the case that the more one attempts to isolate expressive parameters, the less musical the synthetic performance becomes. Thus we end up destroying the very phenomenon we are attempting to observe.

The present study has certainly not been immune to this problem. In evaluating the formalisms, it is often difficult to know whether the success or failure of a particular experiment is due to a fault in the analysis procedure or to the mapping of the deviation parameters to expressive output. Since the choice is between one of two possibilities, however, the problem is fairly easy to track down. Synthesizing several versions of the same score with different synthesis parameters mapped to the same deviation values can help isolate the effects of the analysis from any particular manifestation.

As a musician, I am neither surprised nor alarmed by the discovery that expression should be so resistant to Aristotelean analysis-by-dissection. Having had significant performing experience, I can state that my best performances are usually the ones during which I am least *conscious* of my own mental and physical processes. (As Minsky says, "We understand the least what we do the best.") The same axiom applies equally to both improvisation and memorized performance of a score. In this sense, skilled musical performance seems to bear a resemblance to complex dynamical systems such as weather patterns or the human organism. A change in one parameter must invariably affect the others. This is why I believe a more integrated, Platonic approach to expression (e.g. Clynes) seems to be more effective, if still somewhat mysterious.

5.5. Avoiding predictability "overdose"

Although a certain amount of predictability is necessary for the perception of a pseudo-intelligence, my experience with these microworld experiments has indicated that any performance which applies a particular formal set of rules or functions in the same way for too long will suffer from an "overdose" of predictability. For example, the Pitchvoluration experiments show that a concentration of dominance in any one parameter tends to be less interesting over longer time periods than an equal distribution of dominance among all three parameters. Since in this experiment, parametric dominance is determined by the amount of variability in each parameter, this finding would seem to indicate that composed music intended as input to the system should maintain an appropriate balance of variation in each of its dominant parameters, or, alternatively, should shift the focus of variation among its dominant parameters frequently enough to avoid predictability "buildup".

Subordinate parameters appear to be equally susceptible to predictability overdose. One remedy for this problem is to simply cause the program to invert the sign of the deviation values at prudent points, which would retain the sense of deviation, but in the opposite direction, emphasizing notes which would normally be de-emphasized, and vice-versa. Another possible remedy might be to cause the system to focus more on a small subset of subordinate expressive parameters at any given moment, rather than affecting all of them all the time. Thompson et al (1989) refer to the possibility that

> We may find, for example, that a note is emphasized either by lengthening its duration, *or* by increasing its vibrato, *or* by slightly raising its pitch, *or* by inserting a micropause in front of it, *or* by playing it louder. In other words, there may be *synonyms* for various aspects of musical expression, and a rule may simply represent one out of several possibilities. (p. 80)

Knowing when to use which expressive "synonyms" would be an excellent subject for future study.

The Markov analysis procedure itself may be partly to blame for this predictability overdose. Communication theory tells us that as a particular sequence of events recurs over time, its level of predictability increases. However, the effects of temporal sequencing are not represented by the Markov analysis procedure presented here; every ocurrence of a given transition is given the same degree of emphasis. This clearly violates a common musical principle of variation of repeated material, although at the first order of Markov analysis, the "materials" represented are most likely too short to be perceived as repeated units. Thus, among my reccommendations for future work is to experiment ways of representing temporal sequencing, possibly using higher-order Markov chains.

6. Problems for future study: a composer's perspective

Most research in the field of musical expression has been done by psychologists interested in discovering something about how the mind functions in the perception and/or execution of a performance. As a composer who happens to use computers as a medium of expression, my concerns are of a more practical nature, stemming from a desire to communicate within my medium more effectively. It has gradually become clear that formalisms such as those presented here for generating and controlling the parameters of expressive behavior are very helpful in realizing the expressive potential of computer-generated sound. The present study has begun to fill the void; however, it has also raised several issues which could be profitably addressed by future research.

6.1. The synchronization problem

While applying structure-based deviation values to a melody at least partially solves the problem of timing monotony, it creates difficulties when we bring in secondary or accompanimental voices whose timing relationship to the main voice we wish to control. Simply defining a method of voice synchronization over-simplifies the problem; there appear to be differences in the way performers treat timing relationships among multiple voices which are structurally-based as well (Palmer, 1989). Although this topic is a bit beyond the scope of this paper, it is worth some discussion, as it will be an inevitable source of trouble in implementations of the formalisms in more complex musical situations.

My early efforts toward finding effective solutions to this problem have not been successful. One problem involves the inadequacy of non-relational music representation formats (e.g. standard MIDI files). For example, in a MAX prototype I designed, two aspects of musical structure must be manually entered prior to performance: sectional divisions at which a "re-synchronization" of melody and accompaniment should occur, and classification of notes according to whether they occur in the score simultaneously with the melody or not. "Melody-synchronous" notes are treated in the prototype using the IRCAM "explode" score-following object (Puckette, 1990): they are matched with incoming melodic notes and output at a small delay time after each corresponding melody note. This delay time reflects the phenomenon of pushing the melody slightly forward in time, which is noted in several studies of music performance, including Shaffer (1981) and Palmer (1989). The amount of melodic "pre-delay" may be scaled by deviation parameters of the melody. The timing curve of melody-asynchronous notes is allowed to "follow" that of the melody, using a "leaky integrator" technique to cause a lag in the tempo changes of the accompaniment. This lag has the effect of making the timing curve of the accompaniment more stable than that of the melody, a phenomenon commonly reflected in piano pedagogy. Nevertheless, this method still does not guarantee the proper relationships between melody and accompaniment will be maintained, even with frequent re-synchronization. It would appear that the complex timing relationships between melody and accompaniment merit further study so that more appropriate data structures can be developed.

6.2. The Presentation problem

Computer-music systems have been capable of "spontaneous" generation of music in real-time for many years. Early efforts in this direction include the GROOVE

system (Mathews and Moore, 1970), Martin Bartlett's "Black Box", completed in 1972 (Bartlett, 1979) and the work of the League of Automatic Composers, partially documented in Bischoff, Gold, and Horton (1978). However, the field of live performance with computers has not enjoyed much success, even among aficionados of New Music. One possible reason for the marginaliztion of live computer music seems to be the fact that there is no perceptual difference, from the audience's perspective, between a work of "tape music" generated in a recording studio, and one which is generated in by the computer in real-time on the stage. The implication here is that simply bringing the calculations of the computer up to real-time speed is not enough to secure it a place in the performance hall. The results of the calculations must first of all be perceptually meaningful, which has been the aim of this research. Even if we succeed in introducing an element of expressivity into the output of the computer, however, it remains necessary to create the illusion of a "presence" emanating from the hunks of silicon before an audience can be expected to tell the difference between an interactive system and "music minus one" tape music techniques.

Two methods for establishing such a presence come to mind. One involves the development of systems which react to improvised input; the greater and more apparent the degree of freedom granted to the performer, the more obvious it will be to an audience that the machine is actually responding to the performer, rather than strictly vice-versa. Another method for establishing an electronic presence would be to add a visual component to the computer's output which was correlated in some way to the values controlling the musical output. This might involve projecting images on a screen which correspond to various conditions of musical output (facial expressions, for example) or even the use of servo-motors to control props on stage. While the idea of a "Robo-Performer" is probably in poor taste, the addition of visual elements would certainly bear fruit toward creating more engaging performances.

6.3. The Style Problem

Perhaps the most serious question is the importance of one element of expression which was most strongly filtered out of this study: namely, *style*.

Human listeners seem to be incredibly sensitive to traditional stylistic factors in music performance; it makes sense, therefore, that the absence of reference to specific musical traditions should be problematic in these formalisms. Although my attempts at universality seem to have yielded formalisms which are fairly adaptable to various kinds of music, I fear that this adaptability may come at the expense of obtaining a more

personal voice; in other words, the "jack-of-all-trades" approach may leave us with a virtual performer who is a "master of none".

What seems to be required to surmount this problem is the development of an environment for describing "meta-expressive" formalisms; that is, a toolkit for developing one's own rules and processes, which could include specific stylistic references, and which could be invoked when desired. It is to this task I intend to devote significant amounts of energy in the future. From first appearances, it would seem that the POCO system by Honing and Desain would be the ideal environment for such work, as it provides a standard and flexible framework for the development and integration of expressive constructs.

7. Conclusion

This paper has presented three formalisms for generating and managing expressive performance in computer-music systems, as well as providing a review of some of the theoretical and technical background issues involved in the field. The formalisms appear to be effective in injecting the illusion of a fairly convincing, highly malleable, but somewhat generic, musical "presence" into computerized performances of diverse styles of music. However, due to insufficient data, more conclusive statements about the general applicability of these formalisms will have to await future study. Future work will concentrate on further verification of these formalisms and on making further contributions to the design of useful expressive constructs.

An important benefit of using computers to develop formalizations of expression in music is the unique insight it provides into the relationship between structure and performance. The mastery of this relationship is critical to the art of composition, for the best confluence of musical structure and performance occurs when the composition and notation of the music reinforce specifically those aspects of its structure that most strongly affect its expressive realization. This area of research should therefore be of great interest not only to music theorists, but to all practitioners of music in an age where use of the computer as performer is already a fact of life.

Listening Examples: See the text for further explanation.

1) stochastically toggled vibrato (p. 25)

Csound orchestra file:

```
          instr 1
iamp      =         32767

;global amplitude envelope
k300      linseg    0, .1, 1, p3-.2, 1, .1, 0

;vibrato density factor envelope (0% to 100%)
k250      linseg    0, p3*.1, 0, p3*.7, 1, p3*.2, 1
k200      randh     1, 10

;vibrato function
k160      linseg    4.5, p3, 7
k150      oscili    .03, k160, 1 ;sine
k100      =         (abs(k200) > k250 ? 0 : k150)

;fundamental frequency jitter
k20       randi     .005, 1/.05
k30       randi     .004, 1/.1111
k40       randi     .003, 1/1.2186
k55       =         k20+k30+k40+k100
k50       port      k55, .025

;carrier sine wave
a10       oscili    1, 440+(440*k50), 1

          out a10*k300*iamp
          endin
```

Csound score file:

```
;SINE WAVE
f1 0 8192 10 1
i1 0 10
e
```

2) interpolated square/sine vibrato (p. 26)

Csound orchestra file:

```
          instr 1
iamp      =         32767

;global amplitude envelope
k300      linseg    0, .1, 1, p3-.2, 1, .1, 0

;vibrato interpolation envelope (100% sine to 100% square)
```

```
k200    linseg  1, p3, 0

;interpolated vibrato function
k150    oscili  1, 5, 1 ;sine
k140    oscili  1, 5, 2 ;square
k100    =       30*((1-k200)*k140+(k200*k150))

;carrier sine wave
a10     oscili  1, 440+k100, 1

        out a10*k300*iamp
        endin
```

Csound score file:

```
;SINE WAVE
f1 0 8192 10 1
;SQUARE WAVE
f2 0 8192 7 1 4095 1 1 -1 4096 -1
i1 0 10
e
```

3) “asymmetrical” vibrato using a conditional operator (p. 26)

Csound orchestra file:

```
        instr 1
iamp    =       32767

;global amplitude envelope
k300    linseg  0, .1, 1, p3 -.2, 1, .1, 0

;vibrato assymetricality envelope (100% positive to 100% negative)
k200    linseg  1, p3, 0

;vibrato function
k150    oscili  50, 4, 1 ;sine
k100    =       (k150 > 0 ? k150 * k200 : k150 * (1-k200))

;carrier sine wave
a10     oscili  1, 440+k100, 1

        out a10*k300*iamp
        endin
```

Csound score file:

```
;SINE WAVE
f1 0 8192 10 1
i1 0 10
e
```

4) Yo-Yo Ma recording/Phase Vocoder example (p. 17, 27)

5) predictive portamento vs. pitch lag (p. 27)

Csound orchestra file (voice part only)

```
;VOICE SYNTHESIS INSTRUMENT
sr = 20000
kr = 2000
ksmps = 10

          instr 2
;VOWEL TRANSITION FUNCTION
gk100     oscil1i   0,1,p3,10
          endin

          instr 1
iamp      =         32767/5
itex      =         .003
idebat    =         .01
iatten    =         .007
iolaps    =         16
ifref     =         800

;amplitude envelope
k300  linseg        0, p12, 1, p3-(p12+p13), 1, p13, 0

;PITCH TRANSITION FUNCTION
k200      expseg    p5, p3*3/4, p5, p3/4, p6

;tremelo and jitter
k47       expseg    p9, p3, p11   ;p9=starting base speed, p11=ending base speed
k46       randi     p8, 10 ;frequency randomizer-p8=randomness depth
k45       randi     p5*.01, 10 ;amplitude randomizer

;tremelo: p7=# of 1/4 steps above fundamental
k250      oscili    ((p5*p7*.03)+k45)*k300, k47+k46, 2
k2        port      k250, p10 ;p10=lag time->tremelo shape (square vs. sine)

;fundamental frequency jitter
k20       randi     .01, 1/.05
k30       randi     .01, 1/.1111
k40       randi     .01, 1/1.2186

;adjusted fundamental frequency-vibrato
k50       =         k200+(k20+k30+k40+k2)

;formant bandiwdth lookup
k31       table     gk100,31,1
k32       table     gk100,32,1
k33       table     gk100,33,1
k34       table     gk100,34,1
k35       table     gk100,35,1

;formant frequency  lookup
k21       table     gk100,21,1
k22       table     gk100,22,1
```

```
k23     table   gk100,23,1
k24     table   gk100,24,1
k25     table   gk100,25,1

;formant amplitude lookup
k12     table   gk100,12,1
k13     table   gk100,13,1
k14     table   gk100,14,1
k15     table   gk100,15,1

;FORMANT AMPLITUDE SCALER
k5      =       k300 * (0.8 + 1.05*(1000-k50)/1250)

;FOFS:
;xamp,xfund,xforma,xformb,koct,ktex,kband,kdebat,katt,iolaps,ifna,ifnb,idur
a1      fof     1*k5, k50, 0, (k50 <= k21 ? k21:k50), 1, itex, k31, idebat, iatten, iolaps, 1, 1, p3
a2      fof
        k12*k5, k50, 0, (k22/k21 <= 2 ? k22:(k50*2)+30), 1, itex, k32, idebat, iatten, iolaps, 1, 1, p3
a3      fof     k13*k5, k50, 0, k23, 1, itex, k33, idebat, iatten, iolaps, 1, 1, p3
a4      fof     k14*k5, k50, 0, k24, 1, itex, k34, idebat, iatten, iolaps, 1, 1, p3
a5      fof     k15*k5, k50, 0, k25, 1, itex, k35, idebat, iatten, iolaps, 1, 1, p3
a300    = (a1+a2+a3+a4+a5)*k300*iamp
        out     a300
        endin
```

Csound score file:

```
;SINE
f1 0 4096 10 1
;SQUARE
f2 0 4096 -7 0 12 1 2036 1 12 0 2036

;formant amplitude functions
f12 0 4096 -7 .063027 1024 .102661 1024 .084592 1024 .078922 1024 .062752
f13 0 4096 -7 .004121 1024 .051880 1024 .119087 1024 .131995 1024 .255102
f14 0 4096 -7 .003090 1024 .06396  1024 .100016 1024 .108365 1024 .283594
f15 0 4096 -7 .002600 1024 .028840 1024 .089648 1024 .054131 1024 .114815

;formant frequency functions
f21 0 4096 -7 280 1024 400 1024 650 1024 500 1024 330
f22 0 4096 -7 650 1024 840 1024 1100 1024 1750 1024 2000
f23 0 4096 -7 2200 1024 2800 1024 2860 1024 2450 1024 2800
f24 0 4096 -7 3450 1024 3250 1024 3300 1024 3350 1024 3650
f25 0 4096 -7 4500 1024 4500 1024 4500 1024 5000 1024 5000

;formant bandwidth functions
f31 0 4096 -7 70 1024 86 1024 69 1024 75 1024 89
f32 0 4096 -7 132 1024 109 1024 95 1024 104 1024 114
f33 0 4096 -7 156 1024 130 1024 95 1024 123 1024 132
f34 0 4096 -7 200 1024 138 1024 102 1024 140 1024 145
f35 0 4096 -7 220 1024 157 1024 120 1024 165 1024 162

;ins start dur maxamp fundamental endingFreq inititalVowel transitionPoint finalVowel
;tremeloDepth(1/4-tones above fund.) tremRandomness startingTremSpeed
;tremShape(sine/square) nextDuration endingTremSpeed(next note's startSpeed)
```

i1 0.000000 2.612265 0.888780 550.000000 432.235742 0.000000 0.817309 0.078702 1 1.970341 4.614818 0.036709 0.925133 4.641301
i1 2.572265 0.925133 0.520643 351.349976 354.113218 0.092999 0.779405 0.072978 2 2.498735 4.641301 0.079022 2.545818 5.512615
i1 3.457399 2.545818 0.268194 364.914093 342.019618 0.105639 0.714526 0.197309 7 1.257703 5.512615 ...etc.

6) Expressive Asynchrony (p. 63)
-software: MAX patch (see text)
-sound source: Ensoniq Inc. ESQ-1™

7) Stochastic output with PitchVoluration-based expressive deviations (p. 65)
-software: MAX patch (see text)
-sound source: Yamaha Corp. TX802™

a. with Pitch emphasis
b. with Velocity emphasis
c. with Duration emphasis
d. all parameters equal

8) Chopin Nocturne in Eb
-software: ScorePlot HyperCard stack (see text and Appendix B)
-sound source: Ensoniq Inc. ESQ-1™

a. flat (no deviation)
b. ival only (.2 .2 .2)
c. pval only (.2 .2 .2)
d. ival: .1 .05 .1
pval: .05 .05 .05
e. ival: .2 .1 .2
pval: .1 .1 .1

9) Bach Invention #4
-software: ScorePlot HyperCard stack (see text and Appendix B)
-sound source: E-mu Inc.'s Proteus/2™

a. Level Ratio = 1:1
b. Level Ratio = 2:1
c. Level Ratio = 3:1

10) Varése: Density 21.5

a. MIDI realization (E-mu Inc.'s Proteus/2™)
b. Csound realization

Orchestra File:

```
	instr 1 ;cresc/dim. control
;p4 = startAmp
```

```
;p5 = endAmp

gk100   linseg   p4, p3, p5
        endin


        instr 2
;p3 = duration
;p4 = pitch
;p5 = T1
;p6 = T2
;p7 = T3
;p8 = T4
;p9 = L1
;p10 = L2
;p11 = L3

iamp    =        32767
ivibfamp = p4*.004
ivibrate = (2.*(p4/2000.))+4.7

; LOCAL AMPLITUDE ENVELOPE
k150    linseg   0, p5, p9, p6, p10, p7, p11, p8, 0
k100    =        (gk100 == 0 ? k150 : k150*gk100)

;VIBRATO FUNCTION
k60     linseg 0, .35, 1 ;delay .35 seconds
k50     oscili   k100*k60,ivibrate, 1

;CARRIER SINE WAVE
a10     oscili   k100+(k50*.01), p4+(k50*ivibfamp), 1
        out a10*iamp*.4 ;scaling constant to prevent clipping
        endin
```

Score File:

```
;SINE WAVE
f1 0 8192 10 1

i2 0.000 .225 587.329536 .143 .024 .008 .051 .571 .571 .057
i2 .146 .203 554.365 .127 .018 .018 .040 .573 .573 .573
i2 .294 3.026 622.254 .175 1.372 .646 .833 .182 .699 .182
i2 2.487 .392 466.164 .150 .076 .076 .089 .578 .578 .578
i2 2.789 .273 622.254 .150 .030 .030 .063 .602 .602 .602
i2 2.999 .462 466.164 .090 .103 .103 .166 .582 .582 .582
i2 3.617 2.446 659.255 .075 1.695 .596 .080 .121 .674 .121
i2 6.574 .184 587.330 .100 .031 .031 .022 .609 .609 .609
i2 6.736 .182 554.365 .100 .030 .030 .022 .589 .589 .589
i2 6.896 .737 622.254 .100 .272 .272 .093 .590 .590 .590
i2 7.539 .873 659.255 .100 .428 .231 .114 .592 .592 .059
i2 8.298 .253 466.164 .100 .059 .059 .034 .594 .594 .594
i2 8.517 .731 659.255 .100 .264 .264 .102 .553 .553 .553
i2 9.146 .130 622.254 .100 .006 .006 .019 .598 .598 .598
i2 9.257 .132 659.255 .100 .006 .006 .019 .600 .600 .600
i2 9.370 .934 622.254 .100 .347 .347 .141 .602 .602 .602
i1 10.163 2.663 .567 .252 ;diminuendo
```

i2 10.163 1.800 554.365 .100 .711 .711 .277 1.000 1.000 1.000
i2 11.686 .560 466.164 .100 .186 .186 .088 .736 .736 .736
i2 12.158 .263 659.255 .100 .060 .060 .042 .374 .374 .374
...etc.

Appendix A: Note lists used in listening examples 8, 9, and 10:

8. Chopin Nocturne in Eb

```
begin ini 0.8 39 29 1 ok
begin ini 0.6 9 6 0 ok
0 70 60 250 .38 .25
250 79 60 1000 -.41   .34
1250 77 60 250 -.75   -1.00
1500 79 60 250 -.64   -.56
1750 77 60 750 -.49   -1.00
2500 75 60 500 -.33   .12
end ini 0
begin reac 0.2 19 8 0 ok
3000 70 60 250 -.61   .34
3250 79 65 500 -.48   .34
3750 72 70 250 -.39   .78
4000 84 88 500 -.36   .78
4500 79 75 250 -.38   .56
4750 82 70 750 -.39   .56
5500 80 65 500 -.38  .12
6000 79 60 250 -.37  -.12
end reac 0
begin ant 0.3 26 5 0 ok
6250 77 60 750 -.36  -1.00
7000 79 60 500 -.22  -.56
7500 74 60 250 -.04  .56
7750 75 60 750 .01  -.78
8500 72 60 750 -.17  .56
end ant 0
begin conc 0.7 38 10 0 ok
9250 70 60 250 .27  .56
9500 86 64 250 -.22  .56
9750 84 68 250 -.16  .56
10000 82 71 125 .01  .56
10125 80 74 125 .17  .12
10250 79 77 125 .32  -.12
10375 80 80 125 .47  .56
10500 72 83 125 .61  .56
10625 74 86 125 .75  .12
10750 75 88 1250 .88  -.78
end conc 0
end ini 1
begin conc 0.7 95 46 1 ok
begin ini 0.3 57 15 0 ok
12000 70 42 250 .65  .34
12250 79 42 750 .49  .34
13000 77 42 125 .29  -1.00
13125 79 42 125 .07  -.56
13250 77 42 125 -.20  -1.00
13375 76 42 125 -.41  .56
13500 77 42 125 -.62  .78
```

```
13625 79 42 125 -.59  -.56
13750 77 42 250 -.52  -1.00
14000 75 42 625 -.45  .12
14625 77 42 125 -.38  .12
14750 75 42 125 -.32  .12
```

Chopin, cntd.

```
14875 74 42 125 -.26  .12
15000 75 42 125 -.21  -.78
15125 77 42 125 -.16  .12
end ini 0
begin ant 0.2 74 15 0 ok
15250 79 42 125 -.31  -.56
15375 71 45 125 -.27  .78
15500 72 48 125 -.22  .56
15625 73 51 125 -.17  .78
15750 72 54 125 -.12  .78
15875 77 57 125 -.07  .78
16000 76 60 125 -.02  .56
16125 80 63 125 .04  .78
16250 79 66 125 .10  -.12
16375 85 69 125 .13  .78
16500 84 72 125 .11  .78
16625 79 75 125 .10  .56
16750 82 78 750 .10  .56
17500 80 70 500 .10  .12
18000 79 60 250 .11  -.12
end ant 0
begin reac 0.2 82 6 0 ok
18250 77 60 750 .13  -1.00
19000 79 60 250 .31  -.56
19250 79 60 250 .34  .78
19500 74 60 250 .29  .56
19750 75 60 750 .27  -.78
20500 72 60 750 .26  .56
end reac 0
begin conc 0.8 94 10 0 ok
21250 70 60 250 .66  .56
21500 86 66 250 .07  .56
21750 84 70 250 .09  .56
22000 82 75 125 .25  .56
22125 80 80 125 .39  .12
22250 79 88 125 .51  -.12
22375 80 88 125 .61  .56
22500 72 88 125 .70  .56
22625 74 88 125 .78  .12
22750 75 88 1000 .85  -.78
end conc 0
end conc 1
begin ant 0.9 137 32 1 ok
begin ant 0.8 105 7 0 ok
23750 74 88 250 -1.00  .12
24000 75 88 250 -.72  -.78
24250 77 48 750 -.48  .12
25000 79 48 500 -.28  -.56
25500 77 48 250 -.11  -1.00
25750 77 32 750 .03  .78
```

```
26500 72 32 750 -.51  .78
end ant 0
begin ini 0.2 117 10 0 ok
27250 75 32 250 -.06  .78
27500 75 32 250 -.03  .12
```

Chopin, cntd.

```
27750 75 32 250 -.02  .12
28000 75 32 250 -.03  .12
28250 74 32 125 .02  .12
28375 75 32 125 .12  -.78
28500 77 32 187 .20  .12
28687 75 32 63 .28  .12
28750 75 32 750 .34  .12
29500 70 32 750 .40  .34
end ini 0
begin ini 0.8 124 5 0 ok
30250 82 88 750 .06  .78
31000 81 88 500 .18  .78
31500 79 88 250 .34  .78
31750 77 70 750 .55  -1.00
32500 74 64 750 .81  .78
end ini 0
begin conc 0.8 136 10 0 ok
33250 75 60 750 .83  -.78
34000 74 60 250 .25  .12
34250 72 60 250 .27  .78
34500 74 60 250 .44  .12
34750 70 60 250 .58  .78
35000 71 63 250 .70  .78
35250 71 66 250 .80  .78
35500 72 69 250 .89  .56
35750 72 72 250 .95  .78
36000 74 78 250 -.38  .12
end conc 0
end ant 1
```

9. J.S. Bach: Invention #4:

```
stac
begin ant .2 19 12 1 ok
begin ini .4 11 7 0 ok
0 62 88 120 .20
120 64 88 120 -.56 .76
240 65 88 120 -.52 .24
360 67 88 120 -.23 .50
480 69 88 120 .05 .24
600 70 88 120 .33 .50
720 61 88 120 .60 .76
end ini 0
begin ini .5 18 5 0 ok
840 70 88 120 -.30 .76
```

```
960 69 88 120 -.45 -.00
1080 67 88 120 -.23 -.24
1200 65 88 120 .03 .24
1320 64 88 120 .30 .50
end ini 0
end ant 1
leg
begin ant .5 32 6 1 ok
begin ini .3 26 3 0 ok
1440 65 72 240 -.80 .24
```

Bach, cntd.

```
1680 69 72 240 -.47 .76
1920 74 72 240 .47 .76
end ini 0
begin ini .4 31 3 0 ok
2160 67 72 240 .10 .76
2400 73 72 240 .10 .76
2640 76 72 240 -.10 .76
end ini 0
end ant 1
stac
begin ini .5 72 31 1 ok
begin ini .4 43 7 0 ok
2880 74 88 120 .90 .24
3000 76 88 120 -.04 -1.00
3120 77 88 120 -.19 -.50
3240 79 88 120 -.10 .24
3360 81 88 120 -.03 .50
3480 82 88 120 .02 .76
3600 73 88 120 .04 .76
end ini 0
begin ini .4 57 12 0 ok
3720 82 88 120 -.10 .76
3840 81 88 120 -.33 .76
3960 79 88 120 -.55 .50
4080 77 88 120 -.81 .24
4200 76 88 120 -.76 .24
4320 77 80 120 -.59 -.50
4440 74 80 120 -.41 .50
4560 76 80 120 -.25 -1.00
4680 77 80 120 -.08 -.50
4800 79 80 120 .06 .24
4920 81 80 120 .21 .50
5040 70 80 120 .35 .76
end ini 0
begin ini .5 71 12 0 ok
5160 81 80 120 .50 .76
5280 79 80 120 .05 .50
5400 77 80 120 -.41 .24
5520 76 80 120 -.36 .24
5640 74 80 120 -.20 .24
5760 76 80 120 -.02 -1.00
5880 72 80 120 .15 .50
6000 74 80 120 .32 -.00
6120 76 80 120 .48 -1.00
6240 77 80 120 .66 -.50
6360 79 80 120 .83 .24
6480 69 80 120 1.00 .76
end ini 0
end ini 1
begin conc .5 128 39 1 ok
```

```
begin reac .3 88 12 0 ok
6600 79 80 120 .20 .76
6720 77 80 120 .35 .24
6840 76 80 120 .43 .24
6960 74 80 120 .47 .24
```

Bach, cntd.

```
7080 72 80 120 .36 .76
leg
7200 74 48 120 .12 -.00
7320 76 48 120 -.09 -1.00
7440 77 48 120 -.30 -.50
7560 74 48 120 -.49 .50
7680 76 48 120 -.67 -1.00
7800 77 48 120 -.74 -.50
7920 67 48 240 -.76 .76
end reac 0
begin reac .3 97 7 0 ok
8640 72 48 120 -.73 .50
8760 74 48 120 -.09 -.00
8880 76 48 120 -.06 -1.00
9000 72 48 120 -.22 .50
9120 74 48 120 -.35 -.00
9240 76 48 120 -.44 -1.00
9360 65 48 240 -.51 .76
end reac 0
begin ant .4 107 6 0 ok
9840 70 88 480 -.58 .76
stac
10320 70 88 33 -.31 .76
10353 69 88 33 -.01 -.00
10386 70 88 33 .33 .50
leg
10420 69 88 300 .36 -.00
10560 67 88 240 -.40 -.24
end ant 0
stac
begin ini .3 127 14 0 ok
10800 72 88 120 .34 .50
10920 70 88 120 .26 .76
11040 69 88 120 .17 -.00
11160 67 88 120 .04 -.24
11280 65 88 120 -.12 .24
11400 64 88 120 -.09 .50
11520 65 88 120 .05 .24
11640 67 88 120 .19 .50
11760 69 88 60 .30 .24
11820 67 88 60 .42 -.24
11880 69 88 60 .52 .24
leg
11940 67 88 180 .62 -.24
stac
12120 65 88 120 .71 .24
leg
12240 65 88 240 .80 .76
end ini 0
end conc 1
```

10. Varése: Density 21.5

```
leg .7
ten .5
begin ant .7 108 100 1 ok
begin ant .6 31 28 0 ok
0 65 72 240 -.91 .5
leg .5
240 64 72 240 -.85  .50
swell .8
480 66 88 3520 -.80  .50
leg .4
```

Varése, cntd.

```
4000 61 72 480 -.75  .50
4480 66 72 320 -.70  .76
stac .2
4800 61 72 960 -.65  .50
leg .1
swell .9
5760 67 88 3840 -.60  -.00
leg .2
10560 65 72 240 -.55  .76
10800 64 72 240 -.50  .50
11040 66 72 960 -.46  .50
ten .3
12000 67 72 1120 -.41  .50
leg .2
13120 61 72 320 -.37  .50
13440 67 72 960 -.32  -.00
14400 66 72 160 -.28  .50
14560 67 72 160 -.23  .50
14720 66 72 1120 -.19  .50
cresc start mf
15840 64 72 2080 -.15  .76
cresc end pp
17920 61 88 640 -.11  .76
18560 67 48 320 -.07  -.00
cresc start mf
19520 61 72 320 -.03  .50
ten .8
19840 67 77 640 .02  -.00
20480 69 88 1600 .05  .76
cresc end ff
stac .2
sfp .7
22080 70 88 3520 .07  .26
leg .2
25600 67 38 640 -.00  .50
26240 70 48 640 -.08  .50
26880 67 48 480 -.17  .50
27360 70 48 1440 -.27  .50
stac .1
swell .8
28800 72 88 1760 -.38  .50
end ant 0
leg .3
begin reac .4 56 23 0 ok
30720 73 106 240 -.28  -.50
30960 72 106 240 -.19  -.50
stac .1
ramp .7
31200 73 106 1920 -.10  -.50
leg .1
```

```
34320 72 72 240 -.03  -.50
34560 73 62 1600 .05  -.50
36240 72 72 320 .12  -.50
36480 73 72 480 .18  -.50
36960 72 72 840 .20  -.50
```

Varése, cntd.

```
cresc start mf
37760 73 72 320 .19  -.50
38080 72 96 320 .17  -.50
ramp .8
38400 74 120 3520 .16  .76
cresc end fff
stac .1
42000 80 88 240 .15  .50
42240 74 88 640 .14  .76
42880 68 88 240 .13  .76
43200 74 88 480 .12  .76
43680 80 88 160 .11  .50
43840 63 88 320 .10  .76
44160 69 88 480 .09  .76
ramp .8
44640 75 88 1360 .09  .76
leg .6
cresc start ff
46080 81 106 480 .08  .76
46560 69 106 480 .07  .76
47040 82 106 1440 .06  .76
stac .2
48480 88 106 3840 .06  .76
cresc end fff
end reac 0
begin ant .8 107 49 0 ok
53760 76 48 240 -.11  .76
54000 75 48 240 -.07  .76
dive .6
54240 77 48 1920 -.03  .76
dive .9
56160 74 100 480 .01  .50
stac .8
56640 77 48 320 .05  .76
leg .3
57280 74 48 640 .09  .50
57920 76 48 1120 .13  .76
59040 78 48 480 .17  .76
cresc start p
59520 65 48 800 .20  .76
leg .5
sfp .6
60480 66 118 960 .24  .76
stac .05
sfp .4
61440 91 127 3760 .27  .76
cresc end ff
leg .4
65280 71 38 2560 .30  .76
67840 70 48 320 .34  .50
```

```
stac .8
68160 71 48 480 .37  .26
leg .4
dive .5
68640 71 60 1440 .40  .76
```

Varése, cntd.

```
leg .4
cresc start p
70080 69 48 480 .44  .26
70560 71 48 480 .47  .76
71040 68 48 960 .50  .76
cresc end mp
ramp .8
72480 71 88 2400 .53  .76
leg .4
cresc start f
74880 85 88 120 .55  .76
75000 84 88 120 .58  -1.00
75120 85 88 120 .61  -.76
75240 84 88 120 .64  -1.00
75360 85 88 120 .66  -.76
75480 84 88 120 .69  -1.00
75600 85 88 120 .71  -.76
75720 84 88 120 .74  -1.00
75840 85 88 120 .76  -.76
75960 84 88 120 .79  -1.00
76080 85 88 120 .81  -.76
76200 84 88 120 .83  -1.00
76320 85 88 120 .85  -.76
76440 84 88 120 .88  -1.00
76560 85 88 120 .90  -.76
76680 84 88 120 .91  -1.00
cresc end ff
stac .1
sfp .6
76800 86 120 2400 .86  .76
ten .8
79680 71 38 240 .81  .76
leg .2
79920 70 48 240 .75  .50
80160 71 48 480 .69  .26
cresc start p
80640 69 48 960 .62  .26
81600 70 54 320 .55  .26
81920 71 60 640 .47  .26
cresc end mp
cresc start p
82560 69 48 960 .39  .26
dive  .7
83520 70 88 1680 .30  .26
cresc end f
85200 84 48 120 .20  1.00
85320 85 48 120 -.02  -.76
dive .4
85440 84 54 160 -.28  -1.00
stac .6
```

```
85760 84 48 320 -.60  1.00
86080 84 48 320 -1.00  1.00
end ant 0
end ant 1
```

Appendix B: Hypertalk scripts for ScorePlot program.

"Create MIDI Note list" button:

```
on mouseUp

  put empty into field notes
  put 32767 into CStartLine

  put field DeltaIOI_ival into DeltaIOI_ival
  put field DeltaVel_ival into DeltaVel_ival
  put field DeltaDur_ival into DeltaDur_ival

  put field DeltaIOI_pval into DeltaIOI_pval
  put field DeltaVel_pval into DeltaVel_pval
  put field DeltaDur_pval into DeltaDur_pval

  put field GT into GT

  put "Creating note list..."

  --insert a mono pressure value of zero at beginning of score
  put "monoPres" && 0 && 1 && 0 & return¬
  after field notes

  put 0 into altStart
  repeat with x = 1 to number of lines in field score
    set cursor to busy

    if word 1 of line x of field score is "stac" then
      put "staccato" into aType
    else if word 1 of line x of field score is "leg" then
      put "legato" into aType
    end if

    --calculate crescendo (if one begins on line x)
    put 0 into CrescL
    if word 1 of line x of field score is "cresc" ¬
    and word 2 of line x of field score is "start" then
      put dToVel(word 3 of line x of field score) into startVol

      --calculate length of crescendo
      put x into CStartLine
      put x + 1 into lookAhead
      put altStart into CrescStart
      repeat while word 1 of line lookAhead of field score is not "cresc" and¬
        word 1 of line lookAhead of field score is a number
        get line lookAhead of field score
        put word 5 of it into ival
        put word 6 of it into pval
        put word 1 of it into Cstart
        put word 1 of line lookAhead+1 of field score into CnextStart
        if CnextStart is "cresc" then
          --we're at the end of the crescendo, so
          --calculate the modified duration of the last note
          --and add it to CrescL
          put word 4 of line lookAhead of field score into Cdur
```

[**Create MIDI note List, cntd.]**

```
        if aType is "staccato" then
          add round((1-DeltaDur_ival)*Cdur + (Cdur*DeltaDur_ival*ival)) to CrescL
          add round(Cdur*DeltaDur_pval*pval) to CrescL
        else if aType is "legato" then
          add round(Cdur + (Cdur*DeltaDur_pval*ival)) to CrescL
          add round(Cdur*DeltaDur_pval*pval) to CrescL
        end if
      else
        --add the modified IOI to CrescL
        put (CnextStart - Cstart) * GT into CIOI
        add round(CIOI + (CIOI*DeltaIOI_ival*ival)) to CrescL
        add round(CIOI*DeltaIOI_pval*pval) to CrescL
      end if
      add 1 to lookAhead
    end repeat

    if word 1 of line lookAhead of field score is "cresc" ¬
    and word 2 of line lookAhead of field score is "end" then
      put dToVel(word 3 of line lookAhead of field score) into endVol
    else
      answer "Error: cresc start without end at line" && x with OK
      exit to HyperCard
    end if

    --write the crescendo to note list with a time grain of 100 ms (?)
    put round(CrescL*.01) into CrescLOverHund
    put "writing crescendo with" && CrescLOverHund && "points."
    put endVol-startVol into VolRange

    repeat with Ctime = 0 to CrescLOverHund
      put round((Ctime/CrescLOverHund) * VolRange) + startVol into amount
      put "monoPres" && CrescStart + (Ctime*100) && 1 && amount & return¬
      after field notes
    end repeat
    --set pressure back to 0
    put "monoPres" && CrescStart + CrescL+10 && 1 && 0 & return¬
    after field notes
  end if

  if word 1 of line x of field score is a number then
    get line x of field score
    put word 1 of it into start
    put word 2 of it into pitch
    put word 3 of it into vel
    put word 4 of it * GT into dur
    put word 5 of it into ival
    put word 6 of it into pval

    if aType is "staccato" then
      put round((1-DeltaDur_ival)*dur + (dur*DeltaDur_ival*ival)) into altDur
      add round(dur*DeltaDur_pval*pval) to altDur
    else
      put round(dur + (dur*DeltaDur_ival*ival)) into altDur
      add round(dur*DeltaDur_pval*pval) to altDur
    end if
```

[**Create MIDI note List, cntd.]**

```
    --if within crescendo, set velocity to 1 and let pressure
    --determine volume (Proteus responds best this way).

    if x > CStartLine and x < lookAhead then
      put 1 into altVel
    else
      put round(vel + (vel*DeltaVel_ival*ival)) into altVel
      add round(vel*DeltaVel_pval*pval) to altVel
    end if

    --constrain velocity between 1 and 127
    if altVel > 127 then
      put 127 into altVel
    else if altVel < 1 then
      put 1 into altVel
    end if


    put "note" && altStart && altDur && 1 && pitch && altVel && 64 & return¬
    after field notes

    --find score IOI
    put x + 1 into y
    repeat while y <= number of lines in field score
      if word 1 of line y of field score is a number then
        exit repeat
      end if
      add 1 to y
    end repeat

    put word 1 of line y of field score into nextStart
    put (nextStart - start) * GT into IOI

    add round(IOI + (IOI*DeltaIOI_ival*ival)) to altStart
    add round(IOI*DeltaIOI_pval*pval) to altStart

   end if
  end repeat
  beep 2
  put "MIDI note list complete."
end mouseUp
```

“Create Csound Score” button:

```
on mouseUp

  put field DeltaIOI_ival into DeltaIOI_ival
  put field DeltaVel_ival into DeltaVel_ival
  put field DeltaDur_ival into DeltaDur_ival

  put field DeltaIOI_pval into DeltaIOI_pval
  put field DeltaVel_pval into DeltaVel_pval
  put field DeltaDur_pval into DeltaDur_pval

  put field GT into GT
  put empty into field notes
```

[Create Csound Score, cntd.]

```
put "Creating csound note list..."
put 0 into altStart
put 0 into lookAhead
repeat with x = 1 to number of lines in field score
  set cursor to busy
  put "Writing notes."
  --calculate crescendo length (if one begins on line x)
  put 0 into CrescL
  if word 1 of line x of field score is "cresc" ¬
  and word 2 of line x of field score is "start" then
    put "Calculating crescendo length."

    put dToVel(word 3 of line x of field score) into startVol

    --calculate length of crescendo;
    --NB: This must conform to the sum of the altered IOI's of the notes
    --within the crescendo!

    put x + 1 into lookAhead
    put altStart into CrescStart
    put 1 into CIOIRatio
    --altStart = next note's start time

    repeat while word 1 of line lookAhead of field score is not "cresc"
      set cursor to busy
      get line lookAhead of field score
      if word 1 of it is a number then
        put word 5 of it into ival
        put word 1 of it *.001 into Cstart
        put (word 1 of line lookAhead+1 of field score) into CnextStart
        if CnextStart is "cresc" then
          --this must be the last line of the crescendo, so
          --add the duration of this note to the crescendo length
          put (word 4 of line lookAhead of field score) * .001 into Cdur
          put (CIOIRatio*Cdur) into Cdur
          add (Cdur*DeltaDur_ival*ival) + (Cdur*DeltaDur_pval*pval) to Cdur

          if aType is "staccato" then
            add Cdur*(1-envScale) to CrescL
          else if aType is "legato" then
            add (Cdur + (Cdur*envScale)) to CrescL
          end if

        else if CnextStart is a number then
          put (CnextStart*.001 - Cstart) * GT into CIOI
          put CIOI + (CIOI*DeltaIOI_ival*ival) into CaltIOI
          add (CIOI*DeltaIOI_pval*pval) to CaltIOI
          add CaltIOI to CrescL
          put CaltIOI/CIOI into CIOIRatio
        end if
      else if word 1 of it is "staccato" then
        put "staccato" into aType
        put word 2 of it into envScale
      else if word 1 of it is "legato" then
        put "legato" into aType
```

[**Create Csound score , cntd.]**

```
      put word 2 of it into envScale
    end if
    add 1 to lookAhead
  end repeat

  if word 1 of line lookAhead of field score is "cresc" ¬
  and word 2 of line lookAhead of field score is "end" then
    put dToVel(word 3 of line lookAhead of field score) into endVol
  else
    answer "Error: illegal cresc at line" && x with OK
    exit to HyperCard
  end if

  --write the crescendo to note list as a note statement for instrument 1
  set numberFormat to "#.000"
  put "i1" && CrescStart && CrescL && ¬
  startVol/127 && endVol/127 & return¬
  after field notes
 end if


 if word 1 of line x of field score is "stac" then
  put "staccato" into aType
  put "flat" into eType
  put word 2 of line x of field score into aScale
 else if word 1 of line x of field score is "leg" then
  put "legato" into aType
  put "flat" into eType
  put word 2 of line x of field score into aScale
 end if

 if word 1 of line x of field score is in¬
 "ten sfp swell ramp dive delay" then
  put word 1 of line x of field score into eType
  put word 2 of line x of field score into envScale
 end if

 if word 1 of line x of field score is a number then
  get line x of field score
  put word 1 of it * .001 into start
  put word 2 of it into pitch
  put word 3 of it into vel
  put word 4 of it * .001 * GT into dur
  put word 5 of it into ival
  put pitchToFreq(pitch) into freq
  put word 6 of it into pval
  put 127-vel into velRange

  --always scale duration by the same amount as IOI to ensure that
  --duration deviations are not caused by unequal settings for
  --ΔIOI and ΔDur

  --find score IOI
  put x + 1 into y
  repeat while y <= number of lines in field score
```

[**Create Csound score , cntd.]**

```
    if word 1 of line y of field score is a number then
      exit repeat
    end if
    add 1 to y
  end repeat

  put word 1 of line y of field score * .001 into nextStart
  put (nextStart - start) * GT into IOI

  put (IOI + (IOI*DeltaIOI_ival*ival)) into altIOI
  add (IOI*DeltaIOI_pval*pval) to altIOI
  put altIOI/IOI into IOIRatio

  put (IOIRatio*dur) into dur
  put dur + (dur*DeltaDur_ival*ival) + (dur*DeltaDur_pval*pval) ¬
  into altDur

  if aType is "staccato" then
    put altDur*(1-aScale) into altDur
    put .01 + (.1 * (1-aScale)) into T1
    put .01 + (dur - altDur) into T4
    put altDur - (T1 + T4+.01) into diff
    if diff < 0 then
      put altDur/(T1+T4+.01) into shrinkFac
      put T1*shrinkFac into T1
      put T4*shrinkFac into T4
    end if
    put .5 * (altDur - (T1+T4)) into T2
    put T2 into T3
  else if aType is "legato" then
    put altDur + (altDur * aScale) into altDur
    put .05 + (.25 * aScale) into T1
    put (altDur - dur) into T4
    if T4 < 0 then
      put .05 into T4
    end if
    put .5 * (altDur - (T1+T4)) into T2
    put altDur - (T1 + T4+.01) into diff
    if diff < 0 then
      put altDur/(T1+T4+.05) into shrinkFac
      put T1*shrinkFac into T1
      put T4*shrinkFac into T4
    end if
    put .5 * (altDur - (T1+T4)) into T2
    put T2 into T3
  end if

  put (vel + (velRange*DeltaVel_ival*ival)) into altVel
  add (velRange*DeltaVel_pval*pval) to altVel

  if x > lookAhead then
    put altVel/127 into peak
  else
    put 1 into peak
  end if
```

```
if eType is "swell" then
  put peak * (.1+.8*(1-envScale)) into L1
  put peak into L2
  put L1 into L3
  put T2+T3 into Tsum
  put (Tsum * .2) + (Tsum * .6 * envScale) into T2
  put Tsum - T2 into T3
else if eType is "sfp" then
  put peak into L1
  put peak * (1-envScale) into L2
  put peak * (.5+envScale) into L3
  put T2+T3 into Tsum
  put .01 + Tsum * (1-envScale) into T2
  put Tsum - T2 into T3
else if eType is "dive" then
  put peak into L1
  put peak * (.1+.8*(1-envScale)) into L2
  put L2 into L3
  put T2+T3 into Tsum
  put .01 + (T2 * (1-envScale)) into T2
  put Tsum - T2 into T3
else if eType is "ramp" then
  put peak * (.1+.8*(1-envScale)) into L1
  put peak into L2
  put peak into L3
  put T2+T3 into Tsum
  put (Tsum * .5) + (Tsum * .5 * envScale) into T2
  put Tsum - T2 into T3
else if eType is "ten" then
  put peak into L1
  put peak into L2
  put peak * .1 into L3
  put T2 + (T2 * envScale) into T2
  put T3 - (T3 * envScale) into T3
else if eType is "delay" then
  put peak * (.1+.8*(1-envScale)) into L1
  put L1 into L2
  put peak into L3
  put T2+T3 into Tsum
  put .01+(T3*(1-envScale)) into T3
  put Tsum - T3 into T2
else
  put peak into L1
  put peak into L2
  put peak into L3
end if
set numberFormat to "#.000"

if T2 < 0 then
  put 0 into T2
end if
if T3 < 0 then
  put 0 into T3
end if

--check for errors in envelope calculation
```

[**Create Csound score , cntd.]**

```
      put T1+T2+T3+T4 into Tsum
      put Tsum/altDur into envRatio

      --write the note
      put "i2" && altStart && altDur && freq && T1 && ¬
      T2 && T3 && T4 && L1 && L2 && L3 & return¬
      after field notes

      if envRatio > 1.0099999 or envRatio < .99 then
        beep 2
        answer "WARNING: envelope ratio" && envRatio && "at line" && x ¬
        with "continue", "stop"
        if it is "stop" then
          select line x of field score
          exit to HyperCard
        else
          put ";WARNING: envelope ratio" && envRatio && "at line" && x ¬
          & return after field notes
        end if
      end if

      add altIOI to altStart
    end if
  end repeat
  beep 3
  put "Csound note list complete."
end mouseUp
```

Draw Notes:

```
on mouseUp
  global x_axis_ms, startTime

  --clear screen
  choose pencil tool
  click at 200,200
  choose select tool
  drag from 0,0 to 640,480
  doMenu "Cut Picture"

  --find maximum sum of start time + duration; use to scale view
  if x_axis_ms is empty or startTime is empty then
    updateScale
  else
    answer "Update scale first?" with "no", "yes"
    if it is empty then
      exit to HyperCard
    end if
    if it is yes then
      updateScale
    end if
  end if

  hide message box
  doMenu "Draw Filled"
  choose rectangle tool
```

[Draw Notes, cntd.]

```
  repeat with x = 1 to number of lines in field score
    if word 1 of line x of field score is a number then
      drawNote(x)
    end if
  end repeat
  doMenu "Draw Filled"
  choose browse tool
end mouseUp
```

Markov Analysis Button:

```
on mouseUp
  --This handler generates a first order Markov table
  --of the pitch transitions in the score.

  --store histogram of pitches in Ptable

  initField(Ptable)
  put "Generating Pitch Histogram."
  repeat with x = 1 to number of lines in field score
    set cursor to busy
    if word 1 of line x of field score is a number then
      add 1 to word 2 of line ((word 2 of line x of field score)+1) ¬
      of field Ptable
    end if
  end repeat

  --get rid of non-existent notes
  clearNotes(Ptable)

  put "Generating first-order Markov analysis."
  show field ptable
  doMarkov(Ptable)
  put "Pitch Markov Analysis Complete."
  hide field ptable
  answer "Append analysis to score?" with "no", "yes"
  if it is "yes" then
    appendAnal
  end if
end mouseUp
```

Stack Script:

```
on openStack
  hide menubar
  start using stack "hyperMidi installer"
  hmOpenMIDI
end openStack

function linesToitems values
  put empty into array
  repeat with i = 1 to the number of lines in values
    put line i of values & "," after array
  end repeat
  delete last char of array -- strip trailing comma
  return array
end linesToitems
```

[Stack Script, cntd.]

```
on playfield f
  hmWriteMIDI(hmBuilder(linestoitems(field f)))
end playfield

on playlist L
  hmWriteMIDI(hmBuilder(linestoitems(L)))
end playlist

on newCard
  put "ant" & return & "ini" & return & "reac" & return & "conc" into field types
end newCard

function dispH n
  return item 1 of the rect of field n
end dispH

function dispV n
  return item 2 of the rect of field n
end dispV

function dispB n
  return item 4 of the rect of field n
end dispB

function displayWidth n
  return (item 3 of the rect of field n - item 1 of the rect of field n)
end displayWidth

function displayHeight n
  return (item 4 of the rect of field n - item 2 of the rect of field n)
end displayHeight

function startLine
  put trunc (the scroll of field score / the textHeight of field score) + 2 into z
  repeat while word 1 of line z of field score is not a number
    add 1 to z
  end repeat
  return z
end startLine

function endLine
  put startLine() - 2 + trunc((displayHeight(2) / textHeight of field 2)) into z

  repeat while word 1 of line z of field 2 is not a number
    subtract 1 from z
  end repeat
  return z
end endLine

function firstLine
  put 1 into z
  repeat while word 1 of line z of field score is not a number
    add 1 to z
  end repeat
  return z
```

[Stack Script, cntd.]

```
end firstLine

on prettyScore
  put "One moment while I tidy up..."
  repeat with x = 1 to number of lines in field score
    set cursor to busy
    if word 1 of line x of field score is not a number then
      set the textStyle of line x of field score to bold
    else
      set the textStyle of line x of field score to plain
    end if
  end repeat
  put "Thank you!"
end prettyScore

on checkStructure
  put "CheckStructure: under construction"
end checkStructure

on drawNote x
  global x_axis_ms, startTime, UR

  choose rectangle tool
  --x is index to score field
  put word 1 of line x of field score - startTime into s
  put word 2 of line x of field score into p
  put word 3 of line x of field score into v
  put word 4 of line x of field score into d
  put word 5 of line x of field score into ival
  put word 6 of line x of field score into pval

  -- select appropriate pattern for velocity level
  put round(v/16)+3 into ptn
  if ptn = 11 then
    put 12 into ptn
  end if
  set pattern to ptn

  -- draw the note as a horizontal rectangle
  put dispH(3) + round((s/x_axis_ms)*displayWidth(3)) into xStart
  put dispV(3) + round(((127-p)/127)*displayHeight(3)) into yStart
  put xStart + round((d/x_axis_ms)*displayWidth(3)) into xEnd
  drag from xStart,yStart+1 to xEnd,yStart-1

  --draw structure associated with note (if any)
  put x - 1 into lookBack
  repeat while word 1 of line lookBack of field score is "begin" ¬
    and line lookBack of field score is not empty
    put drawStructure(lookBack) into UL
    --draw line to begining of structural bracket
    choose line tool
    drag from xStart, yStart to word 1 of UL, word 2 of UL
    subtract 1 from lookBack
  end repeat
```

[Stack Script, cntd.]

```
  -- draw I-Curve point
  put displayHeight(1) into DH
  put round(dispV(1) + DH*.5) into originY
  put round(ival*DH*-.5) + originY into yPoint
  choose line tool
  drag from xStart,yStart to xStart,yPoint

  --draw pval oval: Black for positive deviations, white for negative
  choose oval tool
  put abs(round((pval)*5)) into pixels
  if pval >= 0 then
    set pattern to 12
  else
    set pattern to 1
  end if
  drag from xStart-pixels,yPoint-pixels to xStart+pixels,yPoint+pixels
end drawNote

function trans a,alpha,b,length,i
  if length = 1 then
    return b
  else if length < 1 then
    answer "Trans: Invalid length" && length with OK
  end if

  if i < 0 or i > length then
    answer "Trans: Invalid index" && i with OK
  else if alpha is not 0.0 then
    return a + (b-a) * (1-exp(i*alpha/(length-1))) / (1-exp(alpha))
  else
    return a + (b-a) * i * (1/(length-1))
  end if
end trans

on GenCurve
  --first, clear all I-curve parameters
  clearParams

  --now, process begin & end statements in score with format
  --[statement] [type] [depth] [matchIndex] [length]
  --making sure statements match
  procCurve

  --generate curve data; store in invisible field "data"
  --after each curve is generated, the word "ok" appears after its begin statement
  --curves are always added to existing data
  genData

  --normalize curve data between 0 and 1
  normCurve

  --append data to score
  appendData

end GenCurve
```

[Stack Script, cntd.]

```
on clearParams
  put "Clearing score parameters..."
  repeat with x = 1 to number of lines in field score
    set cursor to busy
    if word 1 of line x of field score is "begin" and¬
    the number of words in line x of field score > 3 then
      get line x of field score
      put empty into line x of field score
      put word 1 of it && word 2 of it && word 3 of it into line x of field score
      set the textstyle of line x of field score to bold
    else if word 1 of line x of field score is a number and¬
    the number of words in line x of field score > 4 then
      get line x of field score
      put empty into line x of field score
      put word 1 of it && word 2 of it && word 3 of it && word 4 of it into line x of field score
      set the textStyle of line x of field score to plain
    else if word 1 of line x of field score is "end" then
      get line x of field score
      put empty into line x of field score
      put word 1 of it && word 2 of it into line x of field score
      set the textstyle of line x of field score to bold
    end if
  end repeat
end clearParams

on clearCurve
  put "Clearing curve parameters..."
  put 1 into x
  repeat while x < number of lines in field score
    set cursor to busy
    if word 1 of line x of field score is not a number then
      delete line x of field score
    else
      add 1 to x
    end if
  end repeat
end clearCurve

on procCurve
  put "Processing curve data..."
  put empty into field debug
  put 0 into prevMatchedBegin
  put 0 into level
  repeat with x = 1 to number of lines in field score
    set cursor to busy
    if word 1 of line x of field score is "end" then
      --look back to match end to begin
      put 0 into matchFound
      put x into lookBack
      put word 2 of line x of field score into type
      repeat while lookBack > 1
        set cursor to busy
        subtract 1 from lookBack
```

[Stack Script, cntd.]

```
      if word 2 of line lookBack of field score is type ¬
      and word 1 of line lookBack of field score is "begin"¬
      and word 4 of line lookBack of field score is empty then
        put 1 into matchFound
        put lookBack into matchedBegin
        if matchedBegin > prevMatchedBegin then
          put 0 into level
        else if  matchedBegin < prevMatchedBegin then
          add 1 to level
        end if
        put matchedBegin into prevMatchedBegin
        put "end at line" && x && "matches begin at line" && lookBack¬
        &", level is" && level & return after field debug

        put " " & x into word 4 of line lookBack of field score
        put lookBack into y
        put 0 into count
        --count note statements back up to x using y as index
        repeat while y < x
          add 1 to y
          if word 1 of line y of field score is a number then
            add 1 to count
          end if
        end repeat
        put " " & count into word 5 of line lookBack of field score
        put " " & level into word 6 of line lookBack of field score
        put " " & level into word 3 of line x of field score
        exit repeat
      end if
      if lookBack = 1 and matchFound is 0 then
        answer "Unmatched end at line" & x with OK
        exit to Hypercard
      end if
    end repeat
  end if
 end repeat
end procCurve

on normCurve
 put 0 into max
 repeat with x = 1 to number of lines in field data
  set cursor to busy
  put abs(line x of field data) into temp
  if temp > max then
    put temp into max
  end if
 end repeat

 put 1/max into normFac
 put "Normalizing curve data: normFac = " && normFac
 repeat with x = 1 to number of lines in field data
  set cursor to busy
  put line x of field data * normFac into line x of field data
 end repeat
end normCurve
```

[Stack Script, cntd.]

```
on genData
  put "Generating curve data..."
  put empty into field data
  put 0 into x
  put 0 into count

  repeat while x < number of lines in field score
    set cursor to busy
    add 1 to x
    if word 1 of line x of field score is a number then
      add 1 to count
    else
      if word 1 of line x of field score is "begin" and¬
      word 7 of line x of field score is empty then
        put word 5 of line x of field score into length
        put word 2 of line x of field score into type
        put word 3 of line x of field score into depth
        put (word 6 of line x of field score) + 1 into level

        put depth * (level ^ (field levelRatio -1)) into amp
        if type is ant then
          put round(length * (.5 + .5 * depth)) into goalT
          put amp into goalA
          put amp * -1 into boundAmp
          put 2 - (4*depth) into alpha1
          put -2 + (4*depth) into alpha2
        else if type is ini then
          put round((length * .5) * (1-depth)) into goalT
          put amp * -1 into goalA
          put amp into boundAmp
          put 2 - (4*depth) into alpha1
          put -2 + (4*depth) into alpha2
        else if type is reac then
          put round((length * .5) * (1-depth)) into goalT
          put amp into goalA
          put amp * -1 into boundAmp
          put -2 + (4*depth) into alpha1
          put -2 + (4*depth) into alpha2
        else if type is conc then
          put round(.5 + (length * .5) * (1-depth)) into goalT
          put amp * -1 into goalA
          put amp into boundAmp
          put -2 + (4*depth) into alpha1
          put 2 - (4*depth) into alpha2
        end if
        repeat with n = 0 to goalT-1
          set cursor to busy
          add 1 to count
          set the numberFormat to "#.00"
          add trans(boundAmp,alpha1,goalA,goalT,n) to line count of field data
        end repeat
        put length - goalT into postGoal
        repeat with n = 0 to postGoal-1
          set cursor to busy
          add 1 to count
```

[Stack Script, cntd.]

```
        add trans(goalA,alpha2,boundAmp,postGoal,n) to line count of field data
       end repeat
       put " ok" into word 7 of line x of field score
       put 0 into x
       put 0 into count
      end if
    end if
  end repeat
end genData

on appendData
  put "Appending data to score..."

  put 1 into y
  repeat with x = 1 to number of lines in field score
    set cursor to busy
    if word 1 of line x of field score is a number then
      put " " & line y of field data into word 5 of line x of field score
      add 1 to y
    end if
  end repeat
end appendData

function drawStructure x
  --x is index to score
  choose line tool

  put displayWidth(structure)/number of lines in field score into HR
  put word 4 of line x of field score into endd
  put x into begin
  put word 6 of line x of field score into level

  put round(dispB(structure) - (5 + (level*5))) into height
  put round(dispH(structure)+(begin*HR)) into left
  put round(dispH(structure)+(endd*HR)) into right

  --draw bracket
  drag from left,height to right,height
  drag from left,height to left,dispB(structure)
  drag from right,height to right,dispB(structure)

  put left && height into UL
  return UL
end drawStructure

on updateScale
  global x_axis_ms, startTime

  put word 1 of line firstLine() of field score into startTime

  put 0 into maxTime
  repeat with j = 1 to number of lines in field score
    set cursor to busy
    if word 1 of line j of field score is a number then
      put word 1 of line j of field score + word 4 of line j of field score into temp
```

[Stack Script, cntd.]

```
      if temp > maxTime then
        put temp into maxTime
      end if
    end if
  end repeat
  put maxTime - startTime into x_axis_ms
end updateScale

on initField n
  put "0 0" & return &¬
  "1 0" && return &¬
  "2 0" && return &¬
  "3 0" && return &¬
  "4 0" && return &¬
  "5 0" && return &¬
  "6 0" && return &¬
  "7 0" && return &¬
  ...[etc.]
  "121 0" && return &¬
  "122 0" && return &¬
  "123 0" && return &¬
  "124 0" && return &¬
  "125 0" && return &¬
  "126 0" && return &¬
  "127 0 "¬
  into field n
end initField

on clearNotes n
  put "Clearing non-existent notes.."
  put 1 into y
  repeat while line y of field n is not empty
    set cursor to busy
    if word 2 of line y of field n is 0 then
      delete line y of field n
    else
      add 1 to y
    end if
  end repeat
end clearNotes

on doMarkov f
  --assumes field f contains a Histogram
  addZeros(f)
  repeat with z = 1 to (number of lines in field score - 1)
    if word 1 of line z of field score is a number then
      put word 2 of line z of field score into P1
      put z + 1 into q
      repeat while word 1 of line q of field score is not a number
        add 1 to q
      end repeat
      put word 2 of line q of field score into P2
      add 1 to word whichLine(f,P2)+2 of line whichLine(f,P1) of field f
    end if
  end repeat
```

[Stack Script, cntd.]

```
  normProb(f)
end doMarkov


function whichLine f,n
  --returns which line of field f begins with n
  repeat with x = 1 to number of lines in field f --not really
    if word 1 of line x of field f = n then
      return x
    end if
  end repeat
end whichLine

on addZeros f
  repeat for number of lines in field f
    put 0 & " " after zeroString
  end repeat
  repeat with z = 1 to number of lines in field f
    put zeroString after line z of field f
  end repeat
end addZeros

on normProb f
  --calculate the normalized probability of each transition;
  --out of all possible transitions (= # of notes in score - 1),
  --how frequently did this one occur?
  --This is a variation of the usual(?) Markov chain, where the
  --transition probabilites are calculated in reference to each
  --individual pitch.

  put countNotes(score)-1 into transCount
  repeat with x = 1 to number of lines in field f
    repeat with y = 3 to number of words in line x of field f
      set cursor to busy
      if word y of line x of field f is not 0 then
        set the numberFormat to "#.0000"
        put word y of line x of field f/transCount into¬
        word y of line x of field f
      end if
    end repeat
  end repeat

  --normalize probabilities
  --find maximum p-value
  put "Normalizing probabilities."
  put 0 into max
  repeat with x = 1 to number of lines in field f
    repeat with y = 3 to number of words in line x of field f
      set cursor to busy
      if word y of line x of field f > max then
        put word y of line x of field f into max
      end if
    end repeat
  end repeat
```

[Stack Script, cntd.]

```
  put 1/max into normFac
  repeat with x = 1 to number of lines in field f
    repeat with y = 3 to number of words in line x of field f
      set cursor to busy
      set the numberFormat to "#.00"
      if word y of line x of field f is not 0 then
        put (word y of line x of field f) * normFac into ¬
        word y of line x of field f
      end if
    end repeat
  end repeat
end normProb

function countNotes f
  put 0 into noteCount
  repeat with x = 1 to number of lines in field f
    if word 1 of line x of field f is a number then
      add 1 to noteCount
    end if
  end repeat
  return noteCount
end countNotes

function dToVel d
  if d is "fff" then
    return 120
  else if d is "ff" then
    return 106
  else if d is "f" then
    return 88
  else if d is "mf" then
    return 72
  else if d is "mp" then
    return 60
  else if d is "p" then
    return 48
  else if d is "pp" then
    return 32
  else if d is "ppp" then
    return 16
  else
    answer "Error: unknown dynamic" && d with OK
    exit to HyperCard
  end if
end dToVel

on appendAnal
  --assumes field ptable contains a first-order Markov analysis
  --writes the inverse of the probability of each pitch transition
  --to the score according to -(2*p(i)-1)
  --so that the probability values become deviations around 0

  repeat with z = 1 to (number of lines in field score - 1)
    set cursor to busy
    if word 1 of line z of field score is a number then
```

[Stack Script, cntd.]

```
      put word 2 of line z of field score into P1
      put z + 1 into q
      repeat while word 1 of line q of field score is not a number
        add 1 to q
      end repeat
      put word 2 of line q of field score into P2
      set the numberFormat to "#.00"
      put (word whichLine(ptable,P2)+2 of ¬
      line whichLine(ptable,P1) of field ptable) into pval
      put " " & -((2*pval)-1) into word 6 of line q of field score
    end if
  end repeat
end appendAnal

function pitchToFreq pitch
  return 440*(2^((pitch-60)/12))
end pitchToFreq
```

References

Abbott, C. The 4CED program. in *The Music Machine*, MIT Press, 1989, pp. 311-331.

Anderson, D. and Kuivila, R. Continuous Abstractions for Discrete Event Languages. *Computer Music Journal*, **13:3**, 1989, pp. 11-23.

Anderson, D. and Kuivila, R. Formula: A Programming Language for Expressive Computer Music. *Computer,* July 1991, pp. 12-21.

Apel, W. and Daniel, R. *The Harvard Brief Dictionary of Music.* Simon & Schuster, New York, 1960.

Appleton, J. Reevaluating the Principle of Expectation in Electronic Music. *Perspectives of New Music,* Fall-Winter 1969, **8:1**, pp. 106-111.

Babbitt, M. Twelve-Tone Rhythmic Structure and the Electronic Medium. in *Perspectives on Contemporary Music Theory,* Boretz, B. and Cone, T., eds., W.W. Norton, 1972, pp. 148-179.

Balzano, G. The group-theoretic description of 12-fold and microtonal pitch systems. *Computer Music Journal*, **4:4**, 1980, pp. 66-84.

Balzano, G. Measuring Music. *Action and Perception in Rhythm and Music.* Royal Swedish Academy of Music No. 55, 1987, pp. 177-199.

Bartlett, M. Software for a Microcomputer-Controlled Synthesizer for Live Performance. *Computer Music Journal.* **3:1**, pp. 25-29, 1979.

Bauersfeld, P., Bennet, J. and Lynch, G. *Proceedings of the ACM Conference on Human Factors in Computing Systems.* Addison-Wesley, 1992.

Berry, W. *Structural Functions in Music.* Dover Publications, New York, 1987.

Bischoff, J., Gold, R. and Horton, J. Music for an Interactive Network of Microcomputers. *Computer Music Journal.* **2:3**, pp. 24-29, 1978.

Brinkman, A. A Data Structure for Computer Analysis of Musical Scores. *Proceedings of the ICMC*, 1984,pp. 233-241.

Bregman, A. and McAdams, S. Hearing Musical Streams. in *Foundations of Computer Music*, Roads, C. and Strawn, J., eds., MIT Press, Cambridge, Mass. 1985, pp. 658-698.

Burns, E. and Dixon-Ward, W. Intervals, Scales, and Tuning. in *The Psychology of Music*, Deutsch, D., ed. Academic Press, 1982, pp. 241-270.

Carlson, R., Friberg, A., Fryden, L. Granström, B. and Sundberg, J. Speech and Music Performance: Parallels and Contrasts. *Contemporary Music Review*, **4**, 1989, pp. 391-404.

Chadabe, J. and Meyers, R. An Introduction to the PLAY Program, in *Foundations of Computer Music*, Roads, C. and Strawn, J., eds., MIT Press, Cambridge, Mass. 1985, pp. 523-538.

Chafe, C., Mont-Reynaud, B., and Rush, L. Toward an Intelligent Editor of Digital Audio: Recognition of Musical Constructs. *Computer Music Journal,* **6:1**, 1982, pp. 30-41.

Chin, D. Intelligent Interfaces as Agents. in *Intellgent User Interfaces*, Sullivan, J. and Tyler, S., eds., ACM Press, 1991, pp. 177-206.

Clarke, E. Structure and Expression in Rhythmic Performance. in *Musical Structure and Cognition.*, Howell, P., West, R. and Cross, I., eds., Academic Press, 1985, pp. 209-236.

Clarke, E. Generative Principles in Music Performance. in *Generative Processes in Music,* Sloboda, J., ed., Clarendon Press, London, 1988, pp. 1-26.

Clynes, M. Sentics: The Touch of Emotions. Anchor Press/Doubleday, New York, 1977.

Clynes, M. Expressive microstructure in music, linked to living qualities. in *Studies of Music Performance,* J. Sundberg, ed. Royal Swedish Academy of Music, Stockholm, Sweden, 1983., pp. 76-182.

Clynes, M. Secrets of Life in Music: Musicality Relized by Computer. *Proceedings of the ICMC*, 1984, pp. 225-231.

Clynes, M. What can a musician learn about music performance from newly discovered microstructure principles (PM and PAS)? in *Action and Perception in Rhythm and Music.* Royal Swedish Academy of Music No. 55, 1987, pp. 201-234.

Cone, E. *Musical Form and Musical Performance*, W.W. Norton and Co., 1968.

Cooper, G. and Meyer, L. *The Rhythmic Structure of Music.* University of Chicago Press, 1960.

Dannenberg, R., Fraley, C. and Velikonja, P. Fugue: A Functional Language for Sound Synthesis. *Computer,* July 1991, pp. 36-41.

Desain, P. and Honing, H. Time Functions Function Best as Functions of Multiple Times. *Computer Music Journal,* **16:2**, 1992, pp. 17-34.

Deutsch, D. Grouping Mechanisms in Music. in *The Psychology of Music*, Deutsch, D., ed. Academic Press, 1982, pp. 99-134.

Dixon-Ward, W. Physiology of Hearing. in *Structure and Perception of Electroacoustic Sound and Music.* Nielzén, S. and Olsson, O., eds. Elsevier Science, 1989, pp. 101-108.

Dodge, C. and Jerse, T. *Computer Music: Synthesis, Composition, and Performance.* Schirmer Books, 1985.

Erickson, R. *Sound Structure in Music.* University of California Press, 1975.

Fraise, P. Rhythm and Tempo. in *The Psychology of Music*, Deutsch, D., ed. Academic Press, 1982, pp. 149-181.

Friberg, A., Sundberg, J. and Frydén, L. How to Terminate a Phrase: an Analysis-by-Synthesis Experiment on a Perceptual Aspect of Music Performance. *Action and Perception in Rhythm and Music.* Royal Swedish Academy of Music No. 55, 1987, pp. 49-55.

Friberg, A., Frydén, L., Bodin, L., and Sundberg, J. Performance Rules for Computer-Controlled Contemporary Keyboard Music. *Computer Music Journal,* 1991, **15:2**, pp. 49-71

Flanagan, J.L. Voices of Men and Machines. *Journal of the Acoustical Society of America,* 1972, **51:5**, pp. 1375-1387.

Gabrielsson, A. Performance of Rhythm Patterns. *Scandinavian Journal of Psychology,* 1974, **15**, pp. 63-72.

Gabrielsson, A. Interplay Between Analysis and Synthesis in Studies of Music Performance and Music Experience. *Music Perception,* **3:1**, 1985, pp. 59-86.

Gordon, J.W., and Strawn, J. An Introduction to the Phase Vocoder. in *Digital Audio Signal Processing: An Anthology.* William Kaufmann, Inc. Los Altos, California, 1985, pp. 221-270.

Hiller, L. and Bean, C. Information Theory Analyses of Four Sonata Expositions. *Perspectives of New Music,* 1961, pp. 96-137.

Honing, H. POCO: An Environment for Analysing, Modifying, and Generating Expression in Music. *Proceedings of the 1990 ICMC*.

Honing, H. and Desain, P. Towards a Calculus for Expressive Timing in Music. in *Music, Mind, and Machines*, i.d.b., Ltd., New York, 1992.

Honing, H. and Desain, P. A Microworld Approach to the Formalisation of Musical Knowledge. in *Music, Mind, and Machines*, i.d.b., Ltd., New York, 1992.

Jaffe, D. Ensemble Timing in Computer Music. *Computer Music Journal,* **9:4**, 1985, pp. 38-47.

Johnson, M. Toward an Expert System for Expressive Musical Performance. *Computer*, July 1991, pp. 30-34.

Johnson-Laird, P. Jazz Improvisation: A Theory at the Computational Level. in *Representing Musical Structure*, Howell, P., West, R. and Cross, I., eds., Academic Press, 1991, pp. 291-328.

Jones, M.R. Perspectives on Musical Time. in *Action and Perception in Rhythm and Music.* Royal Swedish Academy of Music No. 55, 1987, pp. 153-175.

Katayose, H. and Inokuchi, S. The Kansei Music System. *Computer Music Journal,* **13:4**, 1989, pp. 72-77.

Koffka, K. *Principles of Gestalt Psychology.* London: Routledge & Kegan Oaul Ltd., 1962.

Krefeld, Volker. The Hand in the Web: An Interview with Michel Waisvisz. *Computer Music Journal*, **14:2**, 1990, pp.28-33.

Laske, O. *Music and Mind: An Artificial Intelligence Perspective.* Computer Music Association, San Fransisco, 1981.

Laurel, B., ed. *The Art of Human-Computer Interface Design.* Addison-Wesley, 1990.

Lerdahl, F. Cognitive Constraints on Cognitive Systems. in *Generative Processes in Music,* Sloboda, J., ed., Clarendon Press, London, 1988, pp. 231-259.

Lerdahl, F. and Jackendoff, R. *A generative theory of tonal music.* MIT Press, Cambridge, Mass., 1983.

Longley, D. and Shain, M. *The Van Nostrand and Reinhold Dictionary of Information Technology.* Macmillan Press, Ltd., 1989, p. 28.

Longuet-Higgins, H.C., and Lee, C.S. The Rhythmic Interpretation of Monophonic Music. in *Studies of Music Performance,* J. Sundberg, ed. Royal Swedish Academy of Music, Stockholm, Sweden, 1983., pp. 7-26.

Loy, G. Composing with Computers - a Survey of Some Compositional Formalisms and Music Programming Languages. in *Current Directions in Computer Music*, MIT Press, 1989, pp. 291-396.

Lussy, M. *Musical Expression: accents, nuances, and tempo in vocal and instrumental music.* Novello, Ewer, and Co., London and New York, 1892.

Mathews, M. The Radio Baton and Conductor Program, or: Pitch, the Most Important and Least Expressive Part of Music. *Computer Music Journal*, **15:4**, 1991, pp. 37-46.

Mathews, M. and Moore, F.R. GROOVE - A program to composer, store, and edit functions of time." *Communications of the Association for Computing Machinery.* **13:12**, 1970, pp. 715-721.

Meyer, L. *Emotion and Meaning in Music.* University of Chicago Press, 1956.

Meyer, L. Some Remarks on Value and Greatness in Music. *The Journal of Art Criticism,* **17:4**, 1959.

Minsky, M. *The Society of Mind.* Simon & Schuster, Inc. 1985.

Moore, F. R. *Elements of Computer Music.* Prentice Hall, 1990.

Moylan, D. *An Analytical System for Electronic Music.* PhD Dissertation, Ball State University*,* 1983.

Palmer, C. Mapping musical thought to musical performance. *Journal of Experimental Psychology: Human Perception and Performance.* **15:12**, 1989, pp. 331-346.

Pellman, S. *An Overview of Current Practices Regarding the Performance of Electronic Music*. Master's Thesis, Cornell University, 1978.

Podnos, T. *Intonation for Strings, Woodwinds, and Singers.* Scarecrow Press, 1981.

Pressing, J. Cybernetic Issues in Interactive Performance Systems. *Computer Music Journal,* **14:1**, 1990, pp. 12-25.

Puckette, M. The Patcher. Proceedings, ICMC 1988 (Cologne).

Puckette, M. EXPLODE: a user interface for sequencing and score following. IRCAM, 1990.

Pylyshyn, Z. *Computation and Cognition: toward a foundation for cognitive science,* Cambridge, Mass., MIT Press, 1984.

Rabiner, L. A Model for Synthesizing Speech by Rule. *IEEE Transactions on Audio and Electroacoustics*, **AU-17**, 1969, pp.7-13.

Randall, J.K. Operations on Waveforms. in *Perspectives on Contemporary Music Theory,* Boretz, B. and Cone, T., eds., W.W. Norton, 1972, pp. 208-213.

Randall, J.K. Two Lectures to Scientists. in *Perspectives on Contemporary Music Theory,* Boretz, B. and Cone, T., eds., W.W. Norton, 1972, pp. 116-128.

Repp, B. Expressive microstructure in music: A preliminary perceptual assessment of four composers' 'pulses', *Music Perception*, **6**, 1969a, pp. 243-273.

Risset, J.-C., and Wessel, D. Exploration of Timbre by Analysis and Synthesis. in *The Psychology of Music*, Deutsch, D., ed. Academic Press, 1982, pp. 26-54.

Rothgeb, J. Simulating Musical Skills by Digital Computer. *Computer Music Journal,* **4:2**, 1980.

Rosenboom, D. Cognitive Modelling and Musical Composition in the Twentieth Century: A Prolgomenon. *Perspectives of New Music*, 1986, pp. 439-446.

Rosenboom, D. Method for Producing Sounds or Light Flashes with Alpha Brain Waves for Artistic Purposes. *Leonardo*, **5**, pp. 152-156.

Rosenboom, D. The Performing Brain. *Computer Music Journal,* **14:1**, 1990, pp. 48-66.

Rowe, R. Machine Listening and Composing with Cypher. *Computer Music Journal*, **16:1**, 1992, pp. 43-63.

Ryan, J. Some remarks on musical instrument design at STEIM. *Contemporary Music Review,* **6:1**, 1991, pp. 3-17.

Schottstaedt, B. The Simulation of Natural Instrument Tones Using Frequency Modulation with a Complex Modulating Wave. in *Foundations of Computer Music*, Roads, C. and Strawn, J., eds., MIT Press, Cambridge, Mass. 1985, pp. 54-64.

Seashore, C. *In Search of Beauty in Music: A Scientific Approach to Musical Esthetics.* Ronald Press, New York, 1947.

Shaffer, L.H. Performances of Chopin, Bach and Bartok: Studies in Motor Programming. *Cognitive Psychology*, **13**, 1981, pp. 326-376.

Shannon, C.E. and Weaver, W. *The Mathematical Theory of Communication*. Urbana, Ill. 1949.

Sloboda, J. Music Performance. in *The Psychology of Music*, Deutsch, D., ed. Academic Press, 1982, pp. 479-496.

Strawn, J. Approximation and Syntactic Analysis of Amplitude and Frequency Functions for Digital Sound Synthesis. *Computer Music Journal,* 1980, **4:3**, pp. 3-23.

Sternberg, S., Knoll, R. and Zukofsky, P. Timing by Skilled Musicians. in *The Psychology of Music*, Deutsch, D., ed. Academic Press, 1982, pp. 182-240.

Sundberg, J., Askenfelt, A. and Fryden, L. Musical Performance: A Synthesis-by-Rule Approach. *Computer Music Journal*, 1983, **7:1**, pp. 37-43.

Sundberg, J., Friberg, A., and Fryden, L. Common Secrets of Musicians and Listeners: An Analysis-by-Synthesis Study of Musical Performance. in *Representing Musical Structure*, Howell, P., West, R. and Cross, I., eds., Academic Press, 1991, pp. 161-200.

Sundberg, J. and Lindblom, B. Generative Theories for Describing Musical Structure. in *Representing Musical Structure*, Howell, P., West, R. and Cross, I., eds., Academic Press, 1991, pp. 245-272.

Sundberg, J. Perception of Singing. in *The Psychology of Music*, Deutsch, D., ed. Academic Press, 1982, pp. 59-98.

Tenney, J. META -/- HODOS and META Meta -/- Hodos. Frog Peak Music, 1988.

Tenney, J. and Polansky, L. Temporal Gestalt Perception in Music. *Journal of Music Theory*, 1980, **24:2**, pp. 205-241.

Thompson, W.F, Sundberg, J., Friberg, A., and Fryden, L. The Use of Rules for Expression in the Performance of Melodies. *Psychology of Music*, 1989, **17**, pp. 63-82.

Todd, N. A Model of Expressive Timing in Tonal Music. *Music Perception*, 1985, **3:1**, pp. 33-57.

Todd, N. Towards a cognitive theory of expression: The performance and perception of rubato. *Contemporary Music Review*, 1989, vol. 4, pp. 405-416.

Vercoe, B. *Csound: A manual for the audio processing system and supporting programs.* Media Lab, M.I.T. 1986.

Wessel, D., Bristow, D., and Settel, Z. Control of Phrasing and Articulation in Synthesis. *Proceedings of the 1987 ICMC*, pp. 108-116.

West, R., Howell, P., and Cross, I. Musical Structure and Knowledge Representation. in *Representing Musical Structure*, Howell, P., West, R. and Cross, I., eds., Academic Press, 1991, pp. 1-32.

Wolf, T. A Cognitive Model of Musical Sight-Reading. *Journal of Psycholinguistic Research,* **5:2**, 1976, pp. 143-171.