CENTER FOR COMPUTER RESEARCH IN MUSIC AND ACOUSTICS
OCTOBER 1993

Department of Music
Report No. STAN-M-82

COMPOSITIONAL TOOLS FOR ADDING EXPRESSION TO MUSIC IN DMIX

Daniel V. Oppenheim

CCRMA
DEPARTMENT OF MUSIC
Stanford University
Stanford, California 94305

Compositional Tools for Adding Expression to Music in DMIX

Daniel V. Oppenheim

Center for Computer Research in
Music and Acoustics (CCRMA)

Stanford University 1993

Compositional Tools for adding Expression to Music in DMIX

Daniel V. Oppenheim

Center for Computer Research in Music and Acoustics (CCRMA)
Department of Music, Stanford University, Stanford, CA 94305
Dan@CCRMA.Stanford.Edu

ABSTRACT

Computer music is often criticized for lack of musical expression. Our goal is to find tools for composers to add expressive details to their music; details analogous to those added by performers of instrumental music. In a theoretical discussion several terms are proposed: Performed Expression, Composed Expression, and Compositional Structure. Through these terms composition, performance, the score, and musical style are examined in their relation to EXPRESSION. Finally, some compositional tools for composing expression within the DMIX environment are demonstrated, and a future project, LeNNY, is outlined.

1. PRELUDE: SCORES, MUSIC, PERFORMANCE, AND EXPRESSION

Wherein music dwells Expression? What is musical expression? Are there different types of musical expression? Is expression **composed**: is it an integral part of a musical idea that cannot be separated from it; is it encoded into the Classical score alongside the composer's musical ideas? Is expression **performed**: does it come to be only when a gifted musician performs the score?

I have no clear cut answers to any of these questions and this paper will not attempt to find them. However, I claim that compositional ideas, as they are notated in Classical scores, are not music; the music, in all of its grandeur and splendor, comes to be only when the score is interpreted by a performer who adds EXPRESSION. Moreover, the traditional score is not a precise representation of music—for the **composer** it is an important visual tool and for the **performer** it is a detailed tabulature. It is true that a competent musician can look at a score and hear the music in his inner ear, but (1) he is probably also adding the expressive nuances that a performer would, and (2) in any case this has no musical meaning to anyone but that individual.

What do traditional scores have to do with computer music? Most systems for Computer Aided Composition (CAC) use the Classical score as their underlying model of composition. In Music-N type languages (Mathews 1969) an *instrument* (synthesis algorithm) is created and then played via a *notelist*. The *notelist* models a Classical score and provides the pitch, amplitude, duration, and other parameters for each *note*. Higher level approaches such as PLA and Common Music allow the user to specify algorithms that in turn produce the *notelists* (*voice* in PLA and *part* in Common Music). However, the composer must still conceptualize his music in terms of *notes* and *notelists* (to avoid confusion I will use 'score' to refer to the Classical score and *notelist* to refer to the alphanumeric score used in composition systems).

Such user interfaces provide the same **conceptual tools** for conceiving and expressing musical ideas that Western composers have been using for centuries. This is a wonderful approach, but alas, several problems directly effect EXPRESSION.

1.1. Scores and Expressive Detail

Traditional music has a solid theoretical foundation that developed over centuries. Music theory provides well defined conceptual elements for thinking about music, analyzing music, conceiving new ideas, and ultimately composing music. On the other hand, performance practice has an equally long tradition but lacks a formal theory (it is passed on from teacher to pupil). There are no well defined concepts that explain expression or expressiveness; event the most detailed treatise CANNOT provide explicit 'formulas' as to what constitutes a 'correct' performance within a given style. Moreover, expressive nuances are NOT absolute. They depend primarily on the performer's personality, but also vary considerably in each performance depending on things such as mood, hall acoustics, the specific instrument, and even humidity.

This did not affect the Classical composer: he was able to notate his musical ideas in a score. However, while doing so he was also taking into account many aspects of performance practice. He could thus be sure (within reason) that the musical EXPRESSION he conceived as an integral part of his musical ideas would indeed come out during a performance. In this sense I regard the score as an accurate and **formal** representation of what I will term the *Compositional Structure*: motivic elements, harmony, counterpoint, orchestration, form, etc. On the other hand, all indications regarding EXPRESSION are minimal and vague (*presto, ritardando, crescendo, sfz, con anima, cedez,* etc.). This works well in Classical music only because both the composer and the performer are familiar with the same performance practice. However, it should be stressed the the *Compositional Structure* is only a PART of the musical idea as it lacks expressive detail.

1.2. *Performed* and *Composed* Expression

In Classical music expression is added during a performance and I propose naming this process *Performed Expression*. In 'traditional' computer music the *notelist* is compiled (played) by a computer with no further human intervention. The composer must therefore add all expressive detail into the *notelist* as the music is being composed. I propose the term *Composed Expression* for describing this process.

One problem of using the Classical score as a metaphor in computer music now becomes clear. Whereas the composer has at his disposal all the traditional **conceptual tools** for working out the *Compositional Structure*, there are no equivalent conceptual or formal tools for accurately specifying or working out expressive detail, i.e. there are no tools for *Composed Expression*. It is obvious that much expressive detail can be lost in such compositions, as all expressive nuances must be explicitly written into the *notelist* and there is no easy way of doing so.

1.3. New Musical Material and Context

I would like to make a further distinction between two general styles of composition: the ***Established*** and the *New*. *Established* music uses familiar musical materials in a style that is generally known and therefore also has an established performance practice. Thus the composer need only specify the *Compositional Structure* and may rely on *performed Expression.*

On the other extreme, *New Music* uses NEW musical materials. The composer must resort to EXPERIMENTAL techniques in his attempt to find the NEW music that naturally derives from such materials (a somewhat idealistic approach). In many cases the language and style of this music are unique so there is no performance practice for it. Here the composer must resort to *Composed Expression*.

In reality, off course, the distinction between *Established* and *New* music is almost never clear cut. It is almost always a question of degree and many works combine both styles. Also, NEW musical materials are typically not so revolutionary as to 'astound' first time listeners; and similarly, the innovations introduced to language and style by such materials are rarely such that experienced performers are no longer able to relate to. Yet, it is a well observed phenomena that it is often hard to predict the perceived result of transforming such new materials (see Schaeffer 1967, Oppenheim 1986). Moreover, music perception is drastically affected by the music context; the very same material might 'behave' completely differently if placed in different parts of a composition. Even the slightest change in such materials might have a dramatic effect on the overall context and this may, in turn, call for a rework of some previously composed sections. Thus, allowing a performer to modify musical attributes might have an unpredicted effect on the *Compositional Structure* and might alter the piece beyond what the composer would accept. I recall a personal experience while composing *Round the Corners of Purgatory* (Oppenheim 1987): I was working with sustained bell-like sounds for several weeks and felt confident that I could work them into a 15 second bridge section. However, as I was working and listening to the result it became obvious that this material 'must' span out and it eventually became the entire second movement over 5 minutes in duration (*Lamentation*). I have heard similar experiences expressed by other composers.

All this is to say that in *New Music* things are more problematic. For extreme cases when the composer cannot anticipate how the *Performed Expression* will effect his ideas, he MUST have TOOLS that enable *Composed Expression*. Note that using such tools does not rule out leaving some aspects open for *Performed Expression*, and that the two forms of expression can be used at any given time.

1.4. Existing Approaches: Separating Composition and Performance

It is surprising to find how **composition** and **performance** are treated in computer music as separate entities. After all, not only must the composer **compose** the work and define the *Compositional Structure* but often he must also **perform** it by adding the *Composed Expression.*

Traditional systems for composition, such as Music-N (Mathews 69), PLA , and Common Music deal strictly with composition and do directly support performance or *Composed Expression*. Rule-based performance systems offer a partial solution to this problem. However, each style (and composer) must have its own set of rules, and in many cases the result must still be refined.

On the opposite spectrum are several new systems that deal specifically with performance. Max Mathew's Radio Drum (Mathews 1991) deals with expression directly; however the system is not designed to take an active role in the process of composition and is geared towards a live performance with realtime synthesis. MAX indeed opens new domains for expression but is geared more towards realtime interaction and less toward composition that is well structured and carefully worked out (Puckette 1991).

Clearly, a synthesis of all these approaches into a single unified system is highly desired. Such a system should also allow the composer to choose what attributes should be *Composed Expressively* and what can be left to a performer. The remainder of this paper discusses my approach to these problems in the design and implementation of DMIX.

2. TOOLS FOR COMPOSING EXPRESSION IN DMIX

DMIX is a large, multi faceted, interactive environment for composing and performing music (Oppenheim 1989, 1992). To my knowledge it is the first environment that directly addresses issues of creativity and expression. DMIX also attempts to enable a composer to interact with his musical ideas while communicating with the full context of his music. It offers a rich palette of tools that include diverse graphical editors, realtime editors, algorithmic composition, functional programing, score tracking, and more. I view its real advantages less in any specific tool and more in the way these tools interact with each other and form a unified music system with a consistent user interface.

The discussion that follows focuses only on aspects that bring together composition and performance. I shall briefly outline some existing tools that enable *Composed Expression.* I will end by outlining LeNNY—a new tool still under development that is designed specifically for adding expression to music.

2.1. Realtime Editing

The Realtime Editor is perhaps the most straightforward tool for adding expression to music. External inputs, such as MIDI controllers, are connected to graphic views where they set or modify any parameter of the music while it is playing. Realtime visual feedback is also provided. Thus, the composer can intuitively set the 'right' parameters in respect to the overall musical context.

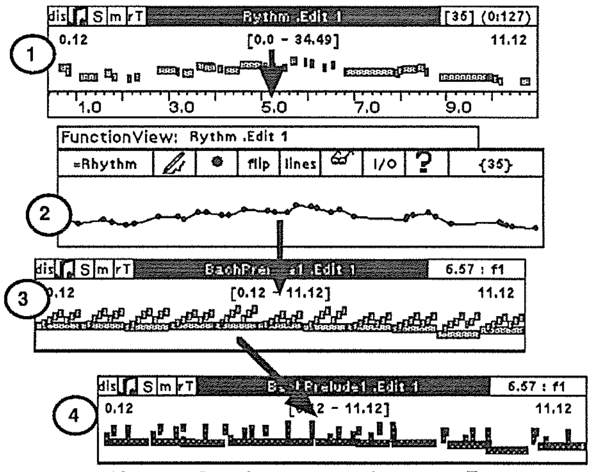
The various modes in which the input effects the music is determined in blocks of compiled Smalltalk code that are stored in a CodeDictionary. The user may edit these modes and add his own, so Realtime Editing is an extendible mechanism that can modify virtually any aspect of the music.\

2.2. Echo: Realtime Interaction with a Performer

Echo represents algorithms and processes that interact with a performer in realtime. Functionally, an Echo resembles a patch in MAX though it is not programmed via graphics (Puckette 91). Materials generated by Echos can be captured in DMIX. Thus, they provide a way for capturing *Performed Expression* and later reusing these gestures as *Composed Expression.*

A different approach in using Echos is within SHADOW—the performance tracking system (Oppenheim 1991). Score tracking is usually a *linear* process: all triggers are predetermined in the notated score. However, SHADOW can also implement what I term *Interactive Tracking* by spawning several Echos that interact with the performer directly (i.e. the performer's gestures will now be captured by the Echos). I view this as a means of expanding the performer's *Expressive Control* over new musical parameters, such as texture, pitch content, rhythms, density, and so on. This is a simple mechanism that enables the composer to blend *Composed* and *Performed Expressions.*

2.3. Modifiers: Capturing and Processing Expressive Gestures



adding ryhtmic expression to Bach

Modifiers include Functions, Filters, and Interpolators and are used extensively within Graphic Views and in the Quill algorithmic music input language (Oppenheim 1990). A unique feature in DMIX is that Music-Events can be transformed into MODIFIERS that can, in turn, be applied to other music objects, and *vice versa.* This offers composers interesting ways to approach *Composed Expression.* In the following illustration the rhythm from a jazz like improvisation [1] was transformed into a Function [2] and then applied to the Bach Prelude in C major [3], resulting in a 'jazzed' up Bach [4]. Similarly, other expressive qualities could be captured, such as dynamics or articulation, and then applied to other sections of music via some intermediating Function.

2.4. Slappability: Distributing Expression between Tools

The most powerful concept in the DMIX user interface is what I term *Slappability:* dragging one view and dropping it on another. For example, an Echo that was just used in a realtime improvisatory like fashion [1] can be *slapped* onto a Quill window where the music it generated will be transformed into an algorithm (in alphanumeric form) [2]. This material, with all of its expressive detail, can now be processed and manipulated out of realtime using high level algorithmic techniques. The reverse will also work: *slapping* a Quill window onto an Echo will cause the music specified in the Quill algorithm to become the input of the Echo as if played by a performer. While this is happening the composer may modify parameters of the Echo and add more expressive detail. The resulting music is, off course, captured and available for further processing.

2.5. LeNNY: Performing Compositional Ideas

LeNNY is a tool still under development that at the time of writing this paper has not yet been implemented. It is intended specifically to enable composers to gradually refine their music by adding *Composed Expression* to the *Compositional Structure*. An underlying idea in LeNNY is to keep the *Compositional Structure* separate from its interpretation. Thus, the composer may first capture his ideas and then gradually add the expressive details that will make them 'work' musically. As structure is kept separate from performance it is always possible to rework sections and give them new interpretations, similar to the way different performers interpret a given work. Alternatively, the same *Structure* could be reused in different parts of the work but modified each time so that it has a different MEANING. LeNNY can be used in any level of the compositional process: from working on a single note to the entire composition. As the composer works he attaches Expressions to the structure. Expressions are *lazy*—they take effect only when the music is played. Expression can be layered ad lib: the composer can modify a certain aspect, listen to the result, and then decide to modify other aspects. There are three main types (Classes) of Expressions: Modifiers, BlockClosures (algorithms), and RealTime.

3. POSTLUDE

Whereas expression cannot be formalized I believe that ways can still be found to specify, edit, and control expression with the aid of computers. My approach is to provide simple tools with a unified user interface that can perform subtle changes to the *Compositional Structure*. I hope that with time composers will be able to conceive the desired musical result in terms of these tools that can produce them. In this way expression itself need not be formalized into a computer program, and the highly complex issues relating to context and meaning can be addressed by the composer directly (in his head).

I believe that we are just beginning to understand the extremely complex issues involved with expression and music composition. It is my sincere hope that as understanding of these issues will increase within the computer music community, new concepts and ways for dealing with expression will be found.

4. ACKNOWLEDGEMENTS

I would like to thank Ruth Oppenheim, Max Mathews, Francisco Kröpfl, and all the wonderful composers at LIPM, Buenos Aires, for their invaluable input, support, and continuous inspiration.

REFERENCES

Mathews, M. V. 1969. "The Technology of Computer Music," MIT Press, Cambridge, Massachusetts.
Mathews, M. V. 1991. "The Radio Baton and Conductor Program, or: Pitch, the Most Important and Leaset Expressive Part of Music," Computer Music Journal 15(4):37-46, Cambridge, Massachusetts.
Oppenheim, D. 1986 "The Need for Essential Improvements in the Machine-Composer interface used for the Composition of Electroacoustic Computer Music" Proceedings of the ICMC, the Hague.
Oppenheim, D. 1987. *Round the Corners of Purgatory*. Electroacoustic composition. Compact Disk. In *Computer Music Currents* vol. 1. WERGO 2021-50.
Oppenheim, D. 1989. "Dmix: An Environment for Composition," Proceedings of the ICMC, Columbus, Illinois.
Oppenheim, D. 1990. "Quill: An Interpreter for Creating Music-Objects Within the Dmix Environment," Proceedings of the ICMC, Glasgow, Scotland.
Oppenheim, D. 1991. "SHADOW: An Object Oriented Performance System for the DMIX Environment," Proceedings of the ICMC, Montreal, Canada.
Oppenheim, D. 1992. "DMIX—A Multi Faceted Environment for Composing and Performing Computer Music: its Design, Philosophy, and Implementation," Proceedings of the *Science and philosophy in Tomorrows Music* conference, Delphi, Greece.
Puckette, M. 1991. "Combining Event and Signal Processing in the MAX Graphical Programming Environment," Computer Music Journal 15(3), MIT Press, Cambridge Massachusetts.
Schaeffer P. and Reibel G. (1967) "Solfége de l'objet sonore" Paris: Edition du Seuil.