

The Writing of Style Libraries for Music Notation and Performance

James Ingram

http://home.t-online.de/home/j.ingram
j.ingram@t-online.de

Freelance music copyist (no academic affiliation). 1969-72 Studied composition under Harrison Birtwistle at the Royal Academy of Music in London. 1971-74 music copyist for Universal Edition (printed scores by Morton Feldman, Earle Brown, Birtwistle etc.). 1974-2000 Karlheinz Stockhausen's principal copyist. Currently working on a large score for Peters Edition in Frankfurt, — and looking for an appropriate framework within which to develop *this* project.

Abstract

This paper proposes a GUI for authoring music with nested music symbols.

The spatial aspects of the symbols (their *names* and the behaviour of those names in space) are strictly separated from their temporal meanings. The spatial behaviours and temporal meanings are stored in separate *functions* which means that they can become increasingly subtle, and that different styles of performance can be associated with identical notation conventions.

A distinction is also made between the spatial and temporal symbol *definitions* (which are stored in style libraries), and their local *instantiations* in a particular score. Users would have access to, and be able to edit, both the definitions (which provide default values) and particular instantiations of the symbols.

More "intelligent" libraries can be trained by directly demonstrating variations of local meaning. The sharing and independent development of these libraries by different users implies that *written* traditions of performance practice are possible.

The background problem

The collapse of the 19th century paradigm for music notation. We are not simply clocks with added "expressivity". It is therefore difficult to program the relation between standard music notation and real events or performance practice.

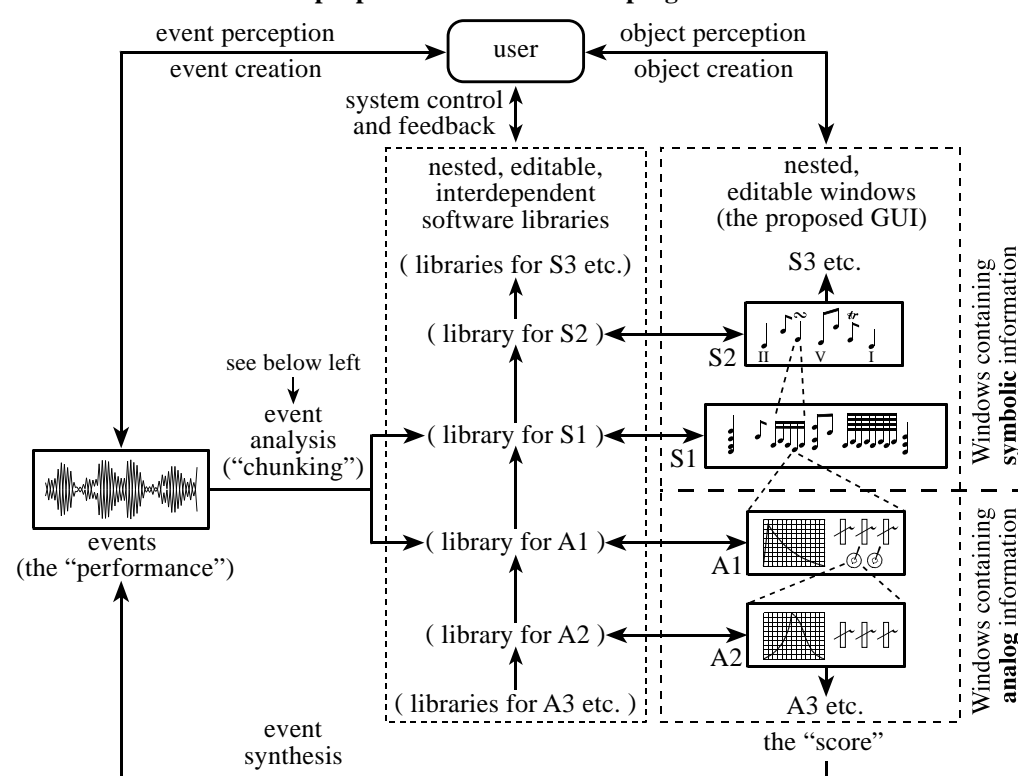
Conclusions and relation to AI

The event editor proposed here is a general framework within which many AI projects might fit - for example in the development of expressive speech for automata.

AI research into performance practice and expressivity may well suggest algorithms which could be used to program the temporal meanings of symbols. This in turn would create *written* traditions of performance practice (software is a form of writing) which would especially benefit lesser known performance traditions and New Music.

Currently, temporal styles have to be learned in real time with two or more people present. The use of written temporal style libraries would enable individuals to learn a new style in their own time, making expensive group rehearsals more productive.

A proposed editor for developing music



The libraries define the spatial and temporal behaviours of each symbol-class in each window of the GUI (e.g. the quaver symbol-class in S1).

The spatial and temporal definitions are strictly independent of each other.

All symbols are defined in terms of the symbols in lower level windows.

The libraries define *control sets* and functions which supply the *default values* of those controls.

Local values (both spatial and temporal) can be edited in the GUI.

The GUI contains a unique, *local value* for each symbol.

The temporal information in the GUI can be used to synthesize events. In effect, the GUI contains a unique, hierarchically organised, editable recording of the events symbolised in the "score".

Music notation as a programming language

Adding a staccato dot to one of the symbolic levels of the GUI would change the envelope stored in the related notehead. This is an example of a general rule: When a user changes or edits an object in one of the symbolic level windows in the GUI, the software has to take the local context of that change into account in order to generate default temporal values for the known parameters of all the related symbols in the vicinity.

The precise value of each individual parameter in each individual symbol is unique, and can relate both actively and passively to the symbol's local context in ways defined in the library. Intuitively understanding the "correctness" of the default value which the software provides, is to recognise the style defined in the library.

Notice that because the default values of particular instantiations of symbols are discovered by performing calls to functions, there are no limits to the complexity of the spatial or temporal style.

It is even possible to think of music notation as an event-oriented programming language having noteheads for function names and other objects in the local context (e.g. staccato dots) as arguments to those functions.

Importantly for getting this project off the ground, libraries may initially use very simple procedures, but become more complex later. There is no reason why libraries with a recognisable style, but whose inner workings can only be understood by a few experts, should not develop. Such libraries could be said to be "more intelligent".

The development of temporal styles

There are two levels at which the temporal styles in the libraries can be defined:

1. Assigning a set of controls to each symbol. Non-expert users could skip this level of the software, but nevertheless use method 2 below.

IRCAM's OpenMusic uses a visual programming environment to define control structures for "Maquettes". A similar solution could be used here, with users constructing custom control sets by dragging and connecting predefined, abstract control objects and the symbols defined at lower levels.

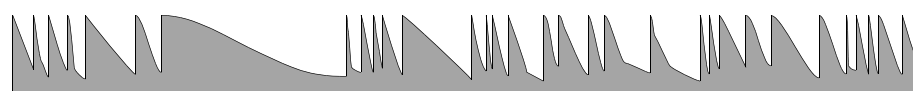
2. Setting the preferences for the default values of a given set of controls. This could for example be done in real time - a method which would be especially important if the library is "more intelligent", and cannot be easily understood or directly edited by the user.

Default values for each symbol's known control set could be related to a statistical analysis of many performances of a given score. (The default value could, for example, easily be related to a Gaussian distribution.) A multi-level score of a piece of classical music could be fed with unlimited numbers of recordings in order to seed its libraries with a classical performance style.

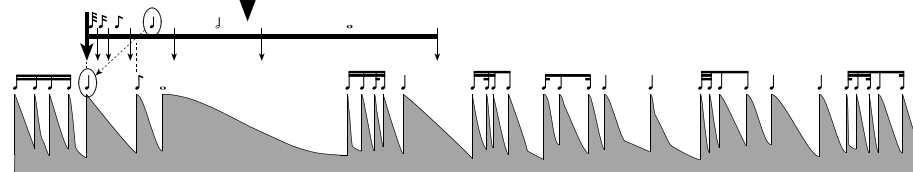
The original, printed score of the piece need not appear verbatim in one of the GUI's symbol-level windows. *All* symbols - even precise metronome marks - have to be considered to be the names of functions.

Some libraries might learn a user's preferences by observing how that user sets or changes particular values in the score, or allow users to edit Gaussian distributions directly...

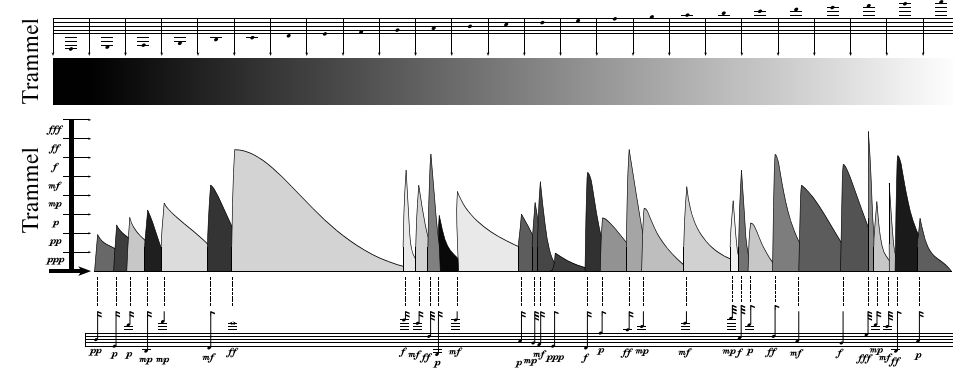
The transcription of durations



Trammel construction (user control via histogram)



Trammels for other symbols (e.g. "pitches" and "dynamics")



Symbol overloading



Event analysis ("Chunking")

This is the process whereby each event in a series of events (represented in a space-time diagram) is given a separate symbol (or name), and the event's name is connected to the space-time information at a lower level.

The top line of the first diagram in the series on the left is a typical **space-time representation** of a series of events. As far as a machine is concerned, this is a single, undifferentiated curve. People however instinctively break such curves into manageable chunks. Such chunks can be labeled just by putting a dot on each peak (the dot can be oval, like a notehead).

The transcription of durations: The lengths of the "events" can be classified, irrespective of the existence of a tempo, using a logarithmically constructed trammel. Using the classic duration symbols means that legibility can be improved later (horizontal spatial compression, use of beams).

It would be useful for the standard notation of tempoed music to be a special case here: A histogram can always be constructed from the lengths of the events, so if the diagram represented a piece of classical music (without triplets) with durations having proportions ca. 2:1, then it would be very easy to construct a trammel to produce the original notation. If there are no such proportions in the original diagram, the user might relate the trammel to the shortest length, or try to ensure maximum differentiation in the resulting transcription.

Trammels for other symbols: The use of trammels is generalisable for other parameters. All symbols have freely definable meanings in the proposed libraries. Conventional meanings can be stored in standard libraries, but this does not preclude unconventional uses and/or **symbol overloading**.

Example: *Sonal Atoms* by Curtis Roads

Transcription by James Ingram (2002).

A transcription like this would be displayed in one of the symbolic level windows in the proposed GUI (S2 or higher, depending on how the libraries are defined).

A prime concern here was to ensure that the connection between the levels of representation should, in principle, be programmable.

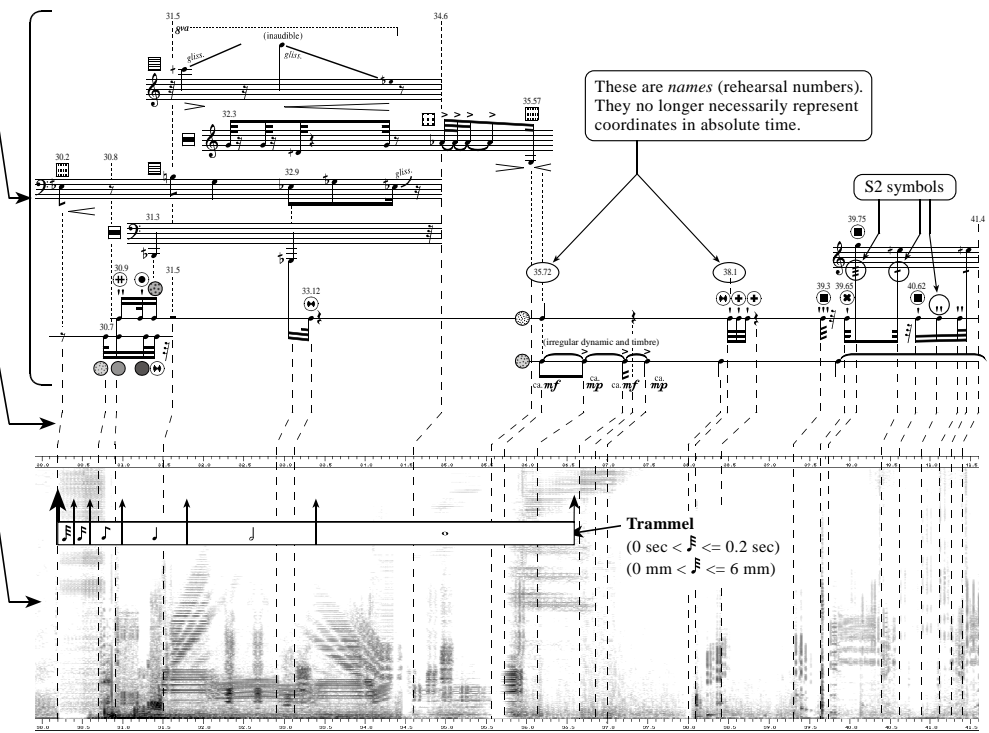
This spatial flexibility is necessary for the efficient use of space and the development of *legibility* in symbolic transcriptions. The spatial and temporal aspects of the symbols must be kept strictly separate.

AudioSculpt* representation (30mm = 1 sec). Space-time diagrams like this might appear in one of the analog level windows.

It should be possible to enhance the utility of many programs which currently use space-time notations by providing them with an interface to libraries of music symbols and their meanings. The existing controls in event-oriented programs (such the knobs and sliders in most synthesizer software) could be fully integrated into such a symbol hierarchy.

Timbres: Curtis Roads asked me to use icons rather than words to designate timbres: "Some electronic music composers have tried to simulate acoustic instruments. Not me." Nevertheless, I used the words in these tables as a general orientation while classifying the timbres by (ear and eye). They may also be helpful for readers of this poster. Note that each icon stands for a *range* of timbres, just as the duration symbols stand for *ranges* of durations. Timbres should also be editable...

* **AudioSculpt** is an IRCAM program.



Symbols (icons) for timbre classes: These timbre classes were chosen because *Sonal Atoms* is one of a collection of electronic music pieces called *Point, Line, Cloud*. Points have a maximum length of 0.1 sec = 3 mm.

