Expressive Notation Package (ENP), a Tool for Creating Complex Musical Output

Mika Kuuskankare and Mikael Laurson

Sibelius Academy
Centre for Music Technology
P.O. Box 86, 00251, Helsinki, Finland
mkuuskan@siba.fi, laurson@.siba.fi

Abstract

This paper presents a new notation package called Expressive Notation Package or ENP. ENP is a PatchWork (PW, Laurson and Duthen 1989, Laurson 1996) user-library. PW, in turn, is a visual language with an emphasis on producing and analysing musical material. PW provides a rich set of applications in computer assisted composition. In this paper we will discuss the structure and properties of ENP at its present state. Several notation examples are provided. Finally we give a musical example created with ENP.

Keywords: music notation, representation of musical structures, computer assisted composition

1. Introduction

ENP is written in Common Lisp and CLOS. It can be further extended and customised by the user. It has a graphical interface requiring no textual input. Musical objects and their properties are editable with the mouse or with specialised editors. The system provides full access to the musical structures behind the notation. Thus ENP can be controlled algorithmically. Furthermore, ENP provides both standard and user definable expressions. The user can create new expressions using inheritance. Note-heads, stem-lengths and slur-shapes can be set either locally or by using global preferences. Scores can be saved as MIDI-, ENIGMA- or ENP-files or they can be exported and printed as PostScript.

ENP has already been used to solve constraint-based problems and to produce control information for several model-based instruments (Laurson 1999a and Laurson et al 1999b). The main purpose of ENP is to provide in the same package both professional notation capabilities and powerful object structures for compositional use.

2. ENP Overview

One of the most important concepts behind ENP is a PW-beat. A PW-beat can be represented in Lisp as a beat-list. A beat-list, in turn, is constructed out of beat-counts and rtm-lists - for more details see Laurson (1996). A

beat-list can be converted into a hierarchical tree structure consisting of beat-objects. The tree can be arbitrary deep. The leaves of the tree structure can contain either chord-objects or rests. A single chord-object, finally, consists of one or several note-objects. A beat-list example with its notational counterpart is given in Figure 1:

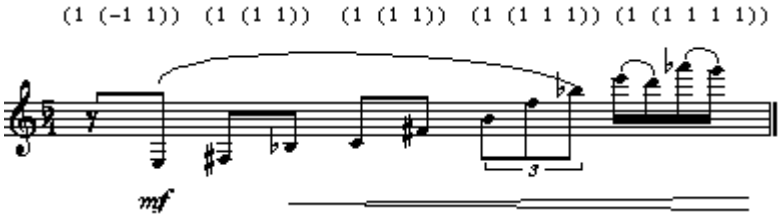

Figure 1. *A beat-list example.*

The beat hierarchy defines besides the rhythmic structure of the musical material also some important ENP user-interface issues. It allows the user to select meaningful musical entities directly by pointing to the respective beat hierarchy level. Thus ENP can distinguish between different musical hierarchies: a note, a chord, a beat (in any depth), a measure, a voice within a part, a part, or even the whole score. For example, in many notation programs the transposition of a note, a chord, or a measure often requires several steps. Furthermore, different modes may have to be utilised. In ENP this operation can be done in one step (Figure 2).

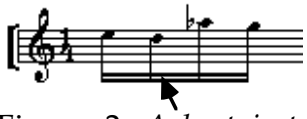

Figure 2. *A beat is transposed by dragging it from the primary beat level.*

Let us consider next an example where the user wants to transpose a complete measure. In many notation programs (such as Finale or Encore) this would mean selecting the measure, opening a dialog, typing some values, maybe selecting a transpose mode and so on. The operation in ENP, by contrast, is done by pointing on the time signature area (the access point to a measure object) and dragging up or down. The effect of the transposition operation can be seen as ENP updates the measure while the user drags the mouse (Figure 3).

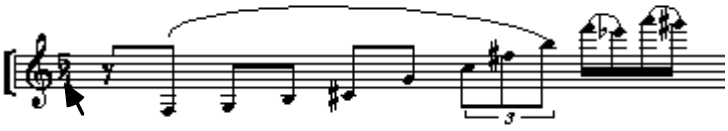

Figure 3. *A measure is transposed by dragging the time signature area.*

ENP contains a set of different edit modes in order to determine the meaning of user actions. In pitch-mode, dragging a note up or down will result in different transpositions. (The previous examples have assumed that the user has operated in the pitch-mode.) In the following we will demonstrate how some other edit modes work. The first mode is the

time-offset-mode (or grace note-mode) and second one is the velocity-mode.

Grace notes are created in PW by editing the time-offset parameter of a note-object. In ENP this is done by dragging the selected object or objects horizontally. Dragging left creates negative offset times and dragging right positive ones. The screen is updated as the user changes the offsets. The spacing between the notes is automatically adjusted. Furthermore, the extra space needed by the grace notes changes the positioning of subsequent notes. (Figure 4).

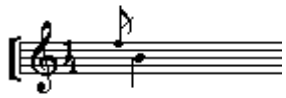

Figure 4. *The user creates grace notes by dragging notes horizontally.*

The power of the hierarchical representation is more obvious in the following example. The user modifies the notes of a chord so that they all become grace notes. Furthermore, in this particular case, the grace notes should form an upward arpeggio. In the time-offset mode this is done by dragging the chord to the right. This creates grace notes with positive offset times and moves them after the first beat. ENP adds also a rest to match the correct number of beats in the measure (Figure 5).

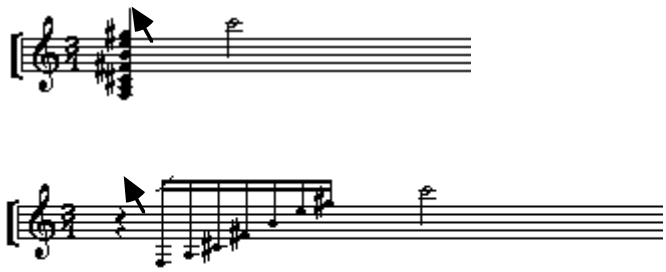

Figure 5. *The user points at the stem of a chord (above). This implies that the subsequent action will affect the whole chord. The time offsets of the notes are changed by dragging the chord horizontally (below) resulting in a group of grace notes.*

The velocity-mode allows to edit MIDI-velocity values of note-objects. This operation can either be absolute or relative. A special velocity editor is used to draw velocity curves. The editor view contains the selected music in the background and a breakpoint function representing the current velocity function in the foreground. The editor can be used to modify the shape of the velocity gesture. A selection of predefined shapes is also at disposal in the form of a breakpoint function library (Figure 6).

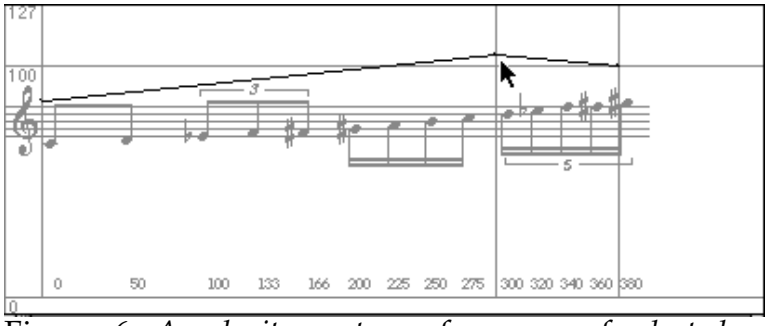

Figure 6. *A velocity gesture of a group of selected notes is edited in the velocity editor. An ascending line is selected from the breakpoint function library. One extra point is added and moved upwards to create the desired curve (crescendo and a slight decrescendo). Values shown on the left are MIDI-velocity values. Values in the bottom represent the start times of the notes.*

3. ENP Expressions

ENP provides also objects representing standard or non-standard musical expressions. Expressions can be applied either to a single note or to a group of notes. ENP allows the user to create new expressions through inheritance. A simple protocol is provided for the user to define new expressions. Properties of the objects, such as typeface and default position, can either be inherited or customised according to personal preferences.

Standard expressions include articulations, tempo indications, lyrics and so on. These have typically a more or less established graphical representation. Standard expressions that are applied to single notes include articulation marks such as accent and tenuto or modes of playing such as tremoli and harmonics (Figure 7).

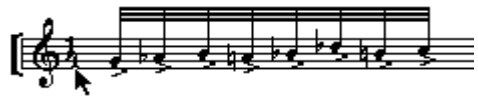

Figure 7. *The user selects a measure by pointing at the time signature and adds an accent - with one operation only - to each note in the measure.*

Standard expressions that are applied to a group of notes include among others slurs, glissandi and ottava signs (Figure 8).

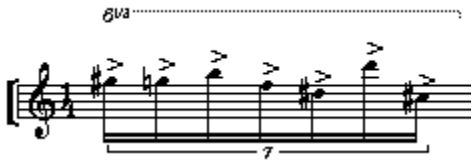

Figure 8. *An ottava-sign is assigned to a whole beat.*

Non-standard expressions include special objects like groups. Groups are entities containing one or more musical objects. They can be used to give a set of objects a common identity. Groups can overlap freely. Thus, groups are useful in representing structures that differ from the ones provided by

the hierarchical beat structure. Groups can be freely positioned in the musical texture (Figure 9).

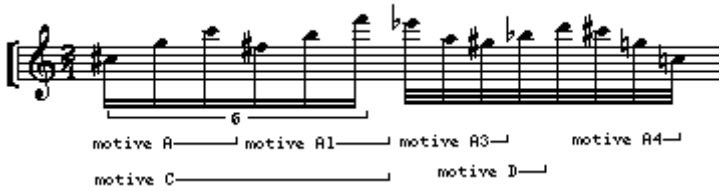

Figure 9. *Overlapping groups showing a result of a motivic analysis.*

A more advanced type of group expression - called group-BPF - can contain breakpoint functions. Group-BPFs have many useful applications. For example, a group-BPF allows to create tempo functions to fine tune the timing of a score. It can also be used to create various envelopes for sound synthesis control (Figure 10).

Figure 10. *An amplitude envelope is defined with the help of a group-BPF. A double-click on the icon above the note opens a specialised breakpoint function editor similar to the one found in Figure 6.*

Besides breakpoint functions the expressions can also include executable code. In the musical example 'Guitarismo!' - the score can be found in the Appendix - the trills, for example, refer to specific compositional algorithms (Laurson 1998 and Laurson et al 1999b). These create auxiliary notes which in turn determine how the score is to be converted into low level control data for synthesis control. This scheme allows precise control of expressive control information. As ENP is method driven the executed code can be instrument specific. Thus a trill for a clarinet part can refer to a different algorithm than, say, a trill for a guitar part (Figure 11).

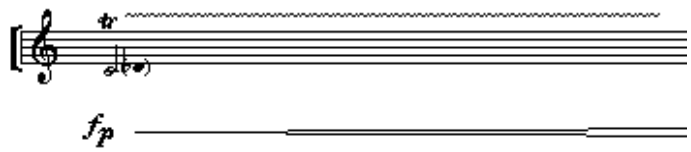

Figure 11. *A trill-expression containing executable code.*

4. Automatic Layout

ENP takes care of the correct placement of the musical symbols. For example, when a note is transposed all the expressions attached to it are automatically moved to their corresponding positions. Multi-measured expressions such as ottava signs are correctly distributed across the systems and pages. They move with the related objects according to a predefined set of rules (Figure 12).

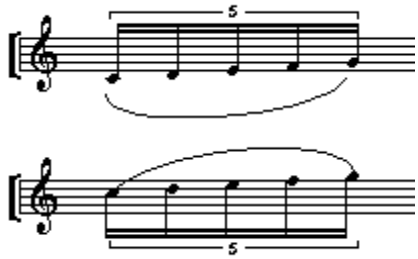

Figure 12. *The slur below the notes (above) is correctly updated when the user transposes the beat by an octave up (below).*

5. ENP Example

To conclude this paper we will present a complete score created with ENP (See Appendix). It is a virtuoso piece for guitar called 'Guitarismo!'. It is not, however, meant to be played by a human performer. Instead, it was composed for demonstrating the timbral and expressional capabilities of a model-based guitar. We will examine some of the points of interest in this particular example.

It should be noted that this example is relatively complex from a notational point of view. The layout was automatically generated with ENP and required almost no 'tidying up' afterwards.

The example utilises some contemporary notational techniques, such as written accelerandi. In many notation programs these are considered to be only of graphical interest. For example, in Finale, making a written accelerando requires that the user edits the graphical representation of the beat, not the actual timing of the notes, as one would expect. In ENP, by contrast, making an accelerando beat requires only to input the proportional durations according to which the beat is divided. ENP takes care of the correct graphical representation.

Some of the seemingly simple notational details can contain powerful expressive qualities. For example, most of the written instructions (such as *molto rubato* or *precipitando*) are in fact group-BPFs. Thus, they contain internally a breakpoint function (the drawing of the excess graphics has been suppressed in the example). These breakpoint functions affect the precise timing of the corresponding gestures.

The playback velocities are calculated by using a simple rule-based system. When the system encounters, for example, an accent it makes the corresponding note or chord somewhat louder according to the surrounding dynamics. This kind of approach makes it easy to produce output with very discrete nuances. It also allows the user to experiment with different approaches in order to find the best possible outcome.

Finally, the last crescendo mark (beginning from the second staff from below and continuing over the line break to the last staff) is an example on how ENP divides automatically multi-measured expressions correctly across parts and pages.

6. Conclusions

Although ENP is still in a development phase it has already proven to be a useful tool for computer assisted composition and analysis. Future plans include among others clarification and optimisation of some of the complex user interface issues discussed above. This phase requires that ENP should be used and tested in the future by a group of professional musicians.

7. Acknowledgement

This work has been supported by the Academy of Finland in project "Sounding Score - Modelling of Musical Instruments, Virtual Musical Instruments and their Control".

References:

G. Read. "Music Notation A Manual of Modern Practice. "
London: Victor Gollanz Ltd, 1982.

M. Laurson, J. Duthen. "PatchWork, a graphical language in PreForm." In *Proc. ICMC'89*, San Francisco, pp. 172-175, 1989.

M. Laurson. "PATCHWORK: A Visual Programming Language and Some Musical Applications." Doctoral dissertation, Sibelius Academy, Helsinki, Finland, 1996.

M. Laurson. "Viuhka". An unpublished user manual, Sibelius Academy, Helsinki, 1998.

M. Laurson. "Recent Developments in PatchWork: PWConstraints - a Rule Based Approach to Complex Musical Problems." To be published in *Symposium on Systems Research in the Arts 1999 - Volume 1: Systems Research in the Arts - Musicology*, IIAS, Baden-Baden, 1999a.

M. Laurson, J. Hiipakka, C. Erkut, M. Karjalainen, V. Välimäki, and M. Kuuskankare. "From Expressive Notation to Model-Based Sound Synthesis: a Case Study of the Acoustic Guitar." In *Proc. ICMC'99*, pp. 1-4, Beijing, China, Oct. 1999b.