ENP-Expressions, Score-BPF as a Case Study

Mika Kuuskankare[1] and Mikael Laurson[2]

[1]Department of Doctoral Studies in Musical Performance and Research
[2]Center for Music and Technology
Sibelius Academy, P.O. Box 86, 00251 Helsinki, Finland
email: mkuuskan@siba.fi, laurson@siba.fi

Abstract

ENP2.0 is a music notation program written in Common Lisp, CLOS, and OpenGL. ENP provides a rich set of notational attributes called ENP-expressions. In this paper, we give an overview of the properties of ENP-Expressions. The underlying system used to handle the graphical representation of ENP-expressions is discussed in detail. A special attention is given to an expression called Score-BPF. The specific problems arising from the need to visually synchronize the linearly spaced (Score-BPF) and non-linearly spaced (music notation) objects is also discussed. Finally, some examples are given on how the properties of Score-BPF can be used to implement various types of editors in ENP.

1 Background

There are some commercial programs, mainly midi- and audio-sequencers, that allow to display various types of data in the form of line graphs with a series of editable breakpoints. For example, in Pro Tools, 'volume', 'pan', or another automated parameter, can be displayed against the waveform or midi track. This is, however, quite straightforward since all the aforementioned objects share the same (linear) representation of time. There is also the possibility in Finale to describe both tempo and dynamic changes with the help of special graphical objects. In this case, the problem lies in the poor visual connection between music and the graph and the lack of real precision of data.

In ENP it is also possible to display various types of graphical information as a part of the musical texture. This can be done using a specialized expression called Score-BPF. ENP provides visual synchronization between Score-BPF and music notation. Furthermore, it is possible to edit this information directly in the score without any external editors.

The rest of the paper is structured as follows. Section 2 introduces the basic concepts behind ENP-Expressions. The overall design issues of ENP expressions are discussed along with some musical examples. Section 3 focuses on Score-BPF and gives specific examples of the use of Score-BPF relating to sound synthesis, timing and tempo modification. The paper ends with some concluding remarks.

2 Expressions Overview

Expressions are divided into two main categories: standard expressions (e.g., articulations, dynamics) and non-standard expressions (e.g., groups, Score-BPF's), (see Kuuskankare and Laurson 2002, for further details).

Every expression is attached either to a single notational object (*single expression*) or to a group of notational objects (*group expression*). This provides expressions for information about both timing and positioning of the related notational objects. This is of primary importance when time synchronization is required.

In the following subsections we will briefly examine how ENP handles expressions in general.

2.1 Single Expressions

When an expression is drawn in a score a number of specialized methods are called. The handling and dispatching of any subsequent methods, however, is done by one primary method called *draw-expression.* The *draw-expression* method first checks if the expression is allowed to be drawn according to user definable preferences or certain exceptions in notation (for example, if the note associated to the expression is hidden or if the expression itself is hidden). When applicable, it handles the drawing of the so-called *print symbol* (each expression contains information about a character or a string that is used when drawing it). For an accent expression the print symbol is ">", for an ottava expression it is "8va", and so on.

In the case of group expressions (see subsection 2.2), *draw-expression* method also identifies such things as the beginning of a new line by inserting the print symbol inside parenthesis, e.g., "(8va)".

The primary method also calls two additional methods, which are (1) *draw-auxiliary-symbols* and (2) *draw-auxiliary-notation-objects*. *draw-auxiliary-symbols* is used to draw any supplementary information required besides the print symbol. *draw-auxiliary-notation-objects* method can be used to draw any auxiliary notational objects (i.e. notes or chords) that are related to the expression in question (Figure 1). An example of such expression is a trill. An auxiliary note can be drawn in the score (as customary in modern notation) to indicate the width of the trill. Furthermore, an invisible (or even visible) note can be used in the case of a glissando to exactly specify the end pitch.

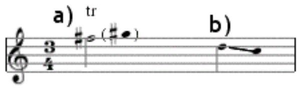

Figure 1. A trill with an auxiliary note (a) and a glissando with a visible end pitch (b).

2.2 Group Expressions

Group expressions inherit from single expressions. Thus, the methods described above are called by the system also in case of group expressions. There are, however, four additional methods that are defined particularly for group expressions: (1) *draw-group-extension*, (2) *draw-group-extension-beginning*, (3) *draw-group-extension-continuation*, and (4) *draw-group-extension-ending*.

The primary method, *draw-group-extension*, is called if there are not any specialized methods defined. Group extension is a visual indication of the extent of a group expression (see Figure 2, point a). *draw-group-extension-beginning* method handles the drawing of the beginning section of the expression. It is also called in the beginning of a new line (Figure 2, point b). *draw-group-extension-continuation* method is primarily used in conjunction with some ENP specific expressions such as groups (as can be seen in Figure 2, point d). It is used to denote that the expression continues to next measure that is either invisible at the time (in another page) or continues in another line. This method is not usually used in standard musical notation. *draw-group-extension-ending* method, in turn, takes care of drawing the appropriate visual clues that indicate the end of the extent of an expression (Figure 2, point c).

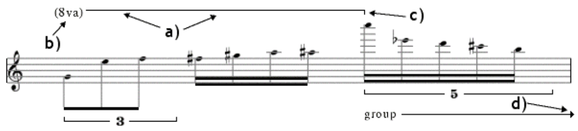

Figure 2. The effects of *draw-group-extension* (a), *draw-group-extension-beginning* (b), *draw-group-extension-ending* (c), and *draw-group-extension-continuation* (d).

2.3 Auto-Cancelled Expressions

Auto-cancelled-expression inherits from group expressions. Its primary function is to provide an automatic mechanism to handle group expressions that initiate a state that needs to be reset or cancelled after the extent of the expression (Figure 3). These kinds of expressions among others are expressions that define playing style (pizz./arco), change in timbre (con sord./senza sord.), and change in register (8va/loco) or even change in tempo (rall./a tempo).

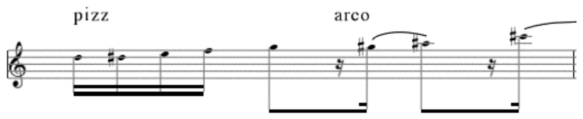

Figure 3. An example of the auto cancellation feature of ENP. The user has inserted a pizzicato expression covering the first five notes (from d^2 through g^2). The word 'arco' is added automatically by ENP to denote the end of the pizzicato style.

2.4 Time-Synchronized Expressions

There are some special expressions in ENP that need to guarantee an exact visual synchronization between linearly spaced objects and the non-linearly spaced music notation. This allows a strong visual feedback for the user since the positioning of all the objects, relational to each other in time, is visually correct.

The x coordinate of any linearly spaced object against music notation can be calculated by comparing the start time of a given object (e.g., point in a break-point function) to the (non-linear) x coordinates of two adjacent notational objects. The equation is as follows,

$$x = x_1 + \frac{t-t_1}{t_2-t_1} \times \left(x_2-x_1\right),$$

where x_1 is the x coordinate of the first notational object, x_2 is the x coordinate of the second notational object, t_1 is the start time of the first notational object, t_2 is the start time of the second notational object, and t is the start time of the linearly spaced object.

In the following section we will examine in detail a time synchronized expression called Score-BPF. A few applications to Score-BPF are presented with the help of some musical examples.

3 Score-BPF

Score-BPF is a specialized form of a group expression. It is a multipurpose graphical object that can represent breakpoint functions as a part of a musical texture (breakpoint functions are piece-wise linear functions). Score-BPF's can be edited directly in the score (see Kuuskankare and Laurson 2002).

There are numerous applications to Score-BPF. It can be used to describe pitch contours and other musical data for constraint-based applications, or to produce control information for model-based instruments (see Laurson et al. 2001). Furthermore, through multiple inheritance the properties and behavior of the Score-BPF can be embedded into other objects (in case of ENP, these kinds of objects are, for example, crescendo and diminuendo and also some special note property editors, see chapter 3.1).

The Figure 4 shows the components of a Score-BPF expression. The expression in question is attached in the first two quarter notes and it is opened (i.e., drawn in an editable state) to reveal the internal breakpoint function with some editable points. The leftmost part of the Score-BPF is reserved for a range display (a). It shows the minimum and maximum y-values, and according to user settings, preferred number of intermediate values. The top-left corner of the Score-BPF (b) shows the number of the current breakpoint function ("1") along with an optional name ("param1"). There is also one selected point in this example (c). Close to it, written inside brackets, are the x-value (time) and the y-value of the point. Furthermore, there is a point with an x-value (time) that exactly matches the time of the note directly below it (d). This synchronization is revealed by a thin vertical line drawn through the point. Finally (e) shows the name and type of the Score-BPF. In this case the user definable name is "parameters" and the type is "time". The characteristics of all the Score-BPF objects of type time is that they are extended to cover the duration of the last notational object (as can be seen in Figure 4). The normal type is extended only to the beginning of the last notational object (see for example Figure 5).

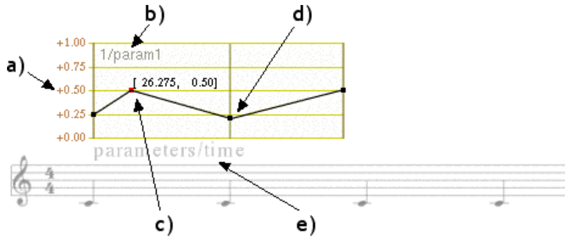

Figure 4. Components of the Score-BPF: range display (a), number and name of the current breakpoint function (b), selected point with numeric display (c), a point synchronizing with the notation (d), and the name and type of the Score-BPF (e).

Next we give a simplified example of the time synchronization feature of Score-BPF. In Figure 5 there are four quarter notes of equal duration, thus the time between each of them is constant. The points (c and d) are placed by the user exactly in between the respective pairs of quarter notes (a and b).

In Figure 6 the last quarter note has been manually moved to the right by the user (this affects only to the spacing of the note, not the time). Now the space between the first two quarter notes (a) is smaller than the space between the last two (b). Nevertheless, the second point (d) is still visually in between the latter two quarter notes (b).

Should there be any further changes in positioning of any of the notes or the overall spacing of the score, the points in the Score-BPF would retain their visual relation to the musical objects unchanged.

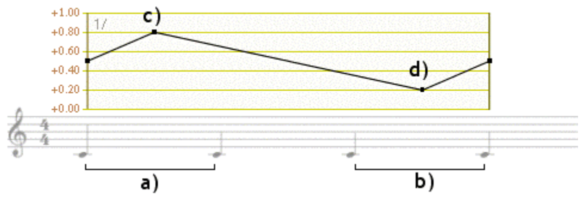

Figure 5. The points (c and d) placed exactly in between the quarter notes (a and b) respectively.

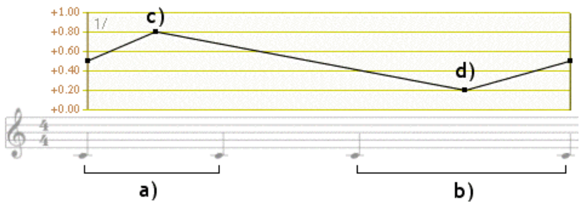

Figure 6. After some editing (manually moving the last quarter note to the right) both points (c and d) are automatically placed in between the respective notes (a and b).

Figure 7 shows some further applications to the Score-BPF. In this example there are two normal Score-BPF expressions (a and b) and an embedded one (c). The first Score-BPF (a) represents an imaginary amplitude envelope for some synthesis engine (for example PWSynth, Laurson and Kuuskankare 2002). The second one (b), in turn, represents a tempo function. The third, the embedded Score-BPF, can be revealed by double clicking the diminuendo expression (c). The result can be seen in Figure 8 (d). Here the user can edit the velocity gesture of the diminuendo.

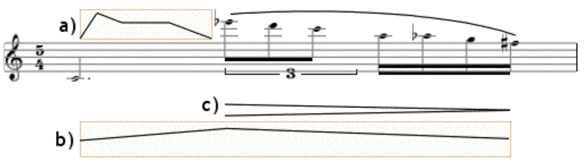

Figure 7. Two different Score-BPF expressions (a and b) and an embedded one (c) (here shown as diminuendo).

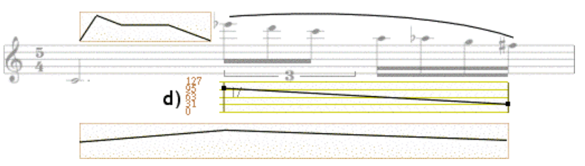

Figure 8. The diminuendo expression (d) is 'opened' to reveal the embedded breakpoint function.

In the following subsections, we present some applications and examples how to use Score-BPF as a general purpose editor inside ENP.

3.1 Note Property Editor

It is sometimes necessary to adjust the data contained by individual notes in the score. In ENP this is accomplished by using a set of specialized *note property editors*. The user can potentially apply various kinds of editors to the score. At the moment there are two editors of this kind in ENP, namely *velocity-editor* and *time-offset-editor*.

The velocity editor can be used to set midi velocity values. This can be done by either drawing an arbitrary breakpoint function describing the continuous variation of dynamics in time (Figure 9, left), or alternatively the velocity values can be set note by note (Figure 9, right).

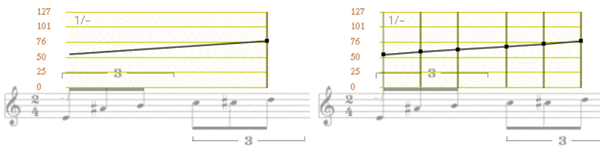

Figure 9. The two edit modes of velocity-editor: the 'breakpoint'-mode (left) and the 'note-by-note'-mode (right).

Time-offset-editor, on the other hand, can be used for adjusting the start times of individual notes inside a chord. This is useful, for example, in keyboard music where it is often unnatural that all the notes of a chord start at exactly the same time. In Figure 10 there are three four-note chords and an open time-offset-editor above the notation. Vertically the points are arranged according to the pitch of the note in question. The horizontal positioning, on the other hand, represents the offset of an individual note from the start time of the chord. The start time of the chord is visually indicated with both a numeric value above the graph and a thin vertical line.

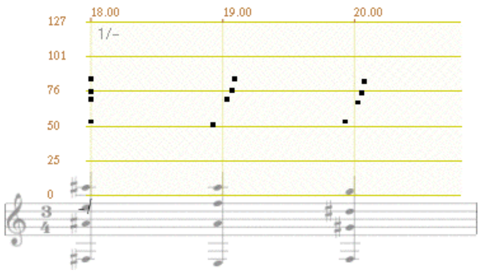

Figure 10. The time-offset-editor. Local adjustments can be made to adjust the start times of individual notes inside a chord.

3.2 Tempo Function Editor

In ENP the relation of performance time to notated time, can be represented as a tempo function, indicating the continuous variation of tempo (Laurson and Kuuskankare 2002). Tempo functions can be drawn in the notation by using a special Score-BPF (Figure 11). The convention selected in our case is to represent 'a tempo' as 100 (e.g., value 200 means twice as fast and value 50 twice as slow).

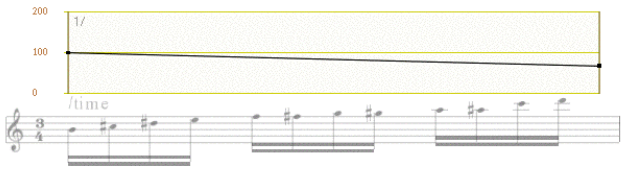

Figure 11. Score-BPF used as a tempo function editor. The example shows a ritardando towards the end of the measure.

4 Discussion and Conclusions

An expression-managing scheme of ENP was presented. Moreover, a special attention was given to Score-BPF and the problems arising from the need to synchronize it with music notation.

There are several advantages offered by the approach presented in this paper. First, it is possible to freely mix linearly and non-linearly spaced objects in music notation and retain a visual time synchronization. Second, the proposed system is uniform and through inheritance can be adapted in various different visualization problems. Third, we do not need any external editors nor do we have to do any kind of adjustments to musical notation (i.e., temporarily to draw it linearly) in order to be able to edit the data.

References

Laurson M., C. Erkut, V. Välimäki, and M. Kuuskankare. 2001. "Methods for Modeling Realistic Playing in Acoustic Guitar Synthesis." *Computer Music Journal* 25(3):38-49.

Kuuskankare M., and M. Laurson. 2002. "ENP2.0, A Music Notation Program Implemented in Common Lisp and OpenGL." *Proceedings of the International Computer Music Conference*. International Computer Music Association, pp. 463-466.

Laurson M., and M. Kuuskankare. 2002. "Instrument Concept in ENP and Sound Synthesis Control." *Journées d'Informatique Musicale*, Marseille.