ELECTRONIC PUBLISHING, VOL. 6(4), 507–518 (DECEMBER 1993)

Representation and manipulation of music documents in ScEX

MIGUEL FILGUEIRAS AND JOSÉ PAULO LEAL

LIACC, Universidade do Porto
R. do Campo Alegre 823
4100 Porto, Portugal

email: mig@ncc.up.pt, zp@ncc.up.pt

SUMMARY

We present the ideas underlying ScEX, a music typesetting system that we are developing at the University of Porto. The focus is on the languages used for representing music documents and on the graphic editor that provides a means for their preparation.

KEY WORDS Music typesetting Music symbolic notation Music graphic editors

1 INTRODUCTION

The problem of typesetting music documents using computers has been mainly addressed in two distinct ways (exactly as for normal texts): systems based on graphic editors commonly known as WYSIWYG ('what you see is what you get') and systems that use commands interspersed in the text. We will not enter the controversy about these two approaches, although we note that, for normal texts, TEX[1] and LATEX[2], which are based on the second approach, are widely used with very good results.

For music texts, and apart from the MusiCopy project [3], we know of two attempts to have TEX as a starting point: Andrea Steinbach and Angelika Schofer's MuTEX[4] and Daniel Taupin's MusicTEX[5] partly using work on MuTEX.

MusicTEX overcomes most of MuTEX limitations and extends it in several directions. Both systems force the user to deal with almost all the details of typesetting. For instance, the spacing between notes must be explicitly chosen. In terms of normal text this would mean that the user would be asked to define the spacing between words or letters. Of course, we are not saying that it is not useful to have commands for fine tuning this kind of details — and having such commands is an advantage over WYSIWYG systems. But it is obvious that there should be default or standard ways of planning the typeset image without user intervention. This is what TEX and LATEX provide us with, and what Daniel Taupin [5] explicitly leaves for 'some pre-compiler' that is to be interfaced to MusicTEX.

The starting point for ScEX (paraphrasing Donald Knuth, insiders pronounce it *skech* for *score (T)eX*) is, then, to have MusicTEX as a target language into which high-level descriptions of scores will be compiled. Such descriptions will be symbolic notations of music texts. Several notations have been proposed so far — for instance, the language of the

CCC 0894–3982/93/040507–12 *Received 15 August 1993*
Revised 1 December 1993

MusiCopy Project [3], SCOT [6], Music [7], and CC [8]. The languages we have defined [9] have some points in common with the first three but possess a much broader coverage. In fact, most of the more complex examples given below cannot be rendered in these languages, as far as we can see from the (maybe partial) descriptions we have of them. Because of the restricted length of the paper it is impossible to present fair comparisons with these languages, as there is no space for describing them.

In this paper, the main focus is on the symbolic notations of music texts we propose, their expressive power, and on the overall architecture of the system. Its implementation being as yet unfinished there is no point in making comparisons with other systems in what concerns resulting scores.

2 ScEX ARCHITECTURE

There are a few important assumptions on the design of ScEX that should be clarified right from the start:

- Our main concern is with music scores and not with interpretations of music pieces. This means, for instance, that we may envisage to have SCOT or MIDI to/from ScEX translators as utilities, but neither SCOT nor MIDI should be compared with ScEX as they are designed for a different purpose, namely, that of specifying the input for sound generators.
- The languages for describing scores should be easy to read so that the end-user may produce scores descriptions by employing a normal text editor. This clearly does not preclude the existence of sophisticated graphic editors and other graphic tools.
- ScEX is intended to make the preparation of a score as light as possible. Therefore the user should not be asked for obvious or repetitive information, and standard ways of typesetting must be available to be applied when concrete information from the user is missing.
- Although we have adopted MusicTEX as the target language, we have not limited the design of ScEX to what is presently offered by MusicTEX. After having a working version of ScEX we will try to overcome the limitations caused by this.

Another important point is that we take an extended notion of ‘voice’ to be central in writing music, and we see a score as a mixture of voices. This leads to the definition of two different representation languages: one for separate voices, ScEX Voice Language (SVL), and another one for complete scores, ScEX Score Language (SSL). Texts in these languages can be created and modified by the user’s preferred text editor, and two programs are required: the *mixer* that creates an SSL text from SVL texts (containing one or more voices) and the *extractor* that extracts a set of voices in SVL from a SSL text.

A typesetting program will translate SSL texts into MusicTEX, which consists of a set of TEX macros using special fonts. The final score is then produced by running MusicTEX and submitting its output to a previewer or a printer. It is to be expected that the complexity of the tasks to be performed by the typesetting program is not compatible with the use of such a specific programming language as the one TEX provides — TEX was not implemented using TEX macros. Therefore appropriate programming languages and sophisticated software tools are needed to implement it.

At the periphery of the ScEX kernel we just described, other desirable utilities are graphic editors for both languages, and translators from/to other languages or representations (such as SCOT and MIDI). Our current concern with graphic software for ScEX is addressed in the last section below.

3 ScEX DESCRIPTION LANGUAGES

In designing ScEX we assumed that the music document is initially prepared in SVL, the user describing the different voices separately. The term ‘voice’ is used here in a not very strict sense as SVL allows for chords to appear instead of single notes whenever needed. This is in line with examples from contrapuntal masterpieces in which at certain points chords with more notes than the number of voices do appear. On the other hand, this is very useful for describing melodic or accompaniment lines having chords, with no need to artificially ascribe each single note to a different voice. We are aware that describing a score on the basis of a division in voices may well lead to difficulties in some situations. We discuss these issues in the next section.

We now briefly present some of the features of SVL and SSL (see [9] for a complete description). We take as an example the first two bars of Mozart’s Sonata in C Major, KV 545, *für Anfänger* (‘for beginners’; [10]) shown[1] in Figure 1. They are also the basis of the ‘simple example’ given in sub-section 1.2 of [5].

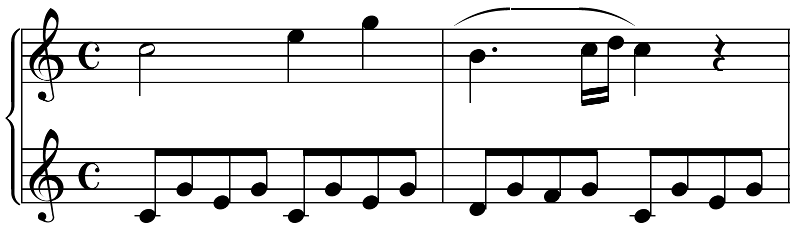

Figure 1. Beginning of Mozart’s Sonata in C Major, KV 545

In this fragment we can easily identify two voices, one in each staff. The notes in the upper voice are denoted in SVL as

```
c5/2 e/4 g | b-/4. c+/16\lnk{2} d c/4 r
```

where `c5/2` means a half-note *c* on the 5th octave (`c4` is the piano’s middle *c*), `e/4` stands for a quarter-note *e* (on the same octave), `g` for a (quarter-note) *g* (same octave), `b-/4.` for a dotted quarter-note *b* on the octave below (4th octave), while `c+/16\lnk{2}` means a 16th-note *c* on the next upper octave (5th octave) and its tail linked to the next note, and `r` denotes a (quarter-note) rest. Alternatively, users of Romance languages may prefer to write[2]

```
do5/2 mi/4 sol | si-/4. do+/16\lnk{2} re do/4 p
```

[1] We thank Daniel Taupin for his kind help with the Figures in MusicTEX.
[2] French speaking people can also use `ut` for *c*.

One of the features of SVL is that some attributes of notes/rests (octave and duration) are inherited from the left (in the same voice) making music descriptions more compact.

The `\lnk` command is an example of a note *embellishment*. In general, embellishments relate to single notes and convey information on signs (like lyrics, ornaments, ties, accents, fingering), as well as indications on how notes and rests should be typeset.

The most important (and mandatory) information within a bar (measure) consists of the denotations of notes and rests. But at the bar level, and for a particular voice, there may be other signs (like expression marks, slurs, brackets, endings), as well as typeset options (e.g., position of signs) for which ScEX provides commands that will appear interspersed with notes and rests. In the present example, the voice in the upper staff can be written in SVL as

```
\begvoice{upper}{piano}
         \staff{1} \clef{g}{2} \time{c}
    c5/2 e/4 g |
    \begslur b-/4. c+/16\lnk{2} d \endslur c/4 r |
\endvoice
```

where `\begvoice` introduces the voice identifier and the identifier of the instrument it belongs to, the pair `\begslur`-`\endslur` is an instance of a denotation for a sign at the bar level, while `\staff{1}` directs ScEX to use the first staff (from above), and `\clef` and `\time` produce the clef and the time-signature.

Another instance of a command at the bar level is `\on` whose arguments are embellishments that will be applied to notes/rests, if appropriate, until a `\off` is found. It is very useful in avoiding repetitions, as will be apparent from the description of the lower voice in our example:

```
\begvoice{lower}{piano}
         \staff{2} \clef{g}{2} \time{c}
    \on{\lnk{4}} c4/8 g e g  c g e g |
    d g f g  c g e g |
\endvoice
```

Bars may start with denotations for selection of staves, clefs and key- and time-signatures, left-repeat marks, and the bar numbers. All these items are optional, although they have to be given in this order (the usual order in a score). Bars are numbered from 1 by default, and an incomplete bar at the beginning should explicitly be given the number 0. By using appropriate bar numbers, bars can be skipped. For finishing a bar, one of the denotations for a single vertical bar (already used above), left-repeat, right-repeat, both left- and right-repeat, double bar, or final bar must appear.

Each SVL document defines one or more voices for a piece of music. Before the description of the voices there is a preamble in which information related with the whole piece can be stated. The only mandatory information in the preamble is given by `\instr` commands that specify the identifier and the number of staves to be used for each instrument. All instruments for which there are voices described in the document are declared at this point. For our example, we would have[3]

[3] The commands in the preamble are commented as this information does not appear in Figure 1.

```
\begsvldoc
% \name{Sonate, KV 545}
% \author{W. A. Mozart}
% \date{Wien, 26 June 1788}
\instr{piano}{2}
\begvoice{upper}{piano}
        \staff{1} \clef{g}{2} \time{c}
   c5/2 e/4 g |
   \begslur b-/4. c+/16\lnk{2} d \endslur c/4 r |
\endvoice
\begvoice{lower}{piano}
        \staff{2} \clef{g}{2} \time{c}
   \on{\lnk{4}} c4/8 g e g  c g e g |
   d g f g  c g e g |
\endvoice
\endsvldoc
```

The translation of the SVL text into an SSL text is done by the mixer. While in SVL each voice description may be seen as a sequence of denotations that are totally ordered with respect to the horizontal dimension of the score (i.e., time), in SSL there is a mixture of denotations from the different voices that are now only partially ordered with respect to the same dimension. Therefore it is to be expected that SVL texts are easier to read and prepare than SSL ones.

In SSL documents there is also a preamble in which, apart from optional information similar to the one in SVL preambles, all the instruments and their voices are declared. The rest of the document is a sequence of bar denotations whose boundaries are the same as in SVL, but whose contents consist of a partially ordered[4] sequence of (global) signs and so-called *voice contributions.* Voice contributions start by an at-sign (‘@’) followed by a voice name between braces. Each contribution ends either when another contribution starts, or when a single at-sign is found. The contents of a contribution are a sequence of bar elements as defined in SVL (with some restrictions).

The first few lines of the SSL translation of the SVL text above are

```
\begssldoc
\instr{piano}{2}
   \voice{upper}
   \voice{lower}
\staff{1} \clef{g}{2} \time{c}
\staff{2} \clef{g}{2} \time{c}
\anystaff
@{upper} \staff{1} c5/2
@{lower} \staff{2} \on{\lnk{4}} c4/8 g e g
@{upper} e5/4
@{lower} c4/8 g
%...
```

[4] In the sense that a denotation of a sign that should be put to the left of another sign, ought to appear before the other’s denotation.

4 SCORES AND VOICES

We discuss now some problems of converting scores into a mixture of voices (see [9] for a more detailed presentation). Such problems obviously will not appear in music written in more or less strict counterpoint or with a single voice as is usual for instruments like the flute or the violin. Even if some of these instruments can produce more than one note at a time (and this can be dealt with in SVL by using chords instead of a note in a single voice), their single-staff scores will never be as complex as scores for the piano or the organ might be.

One of the difficulties is deciding how many voices should be considered. Three situations are common: doubling of notes (with the same pitch and duration), holding of notes belonging to a voice or part while it goes on with other notes, and the use of counterpoint in localized sections (semi-contrapuntal treatment of parts).

These situations are sometimes hardly distinguishable from one another. For instance, in the accompaniment line of Chopin's Scherzo op. 31 [11] starting in bar 65, shown in Figure 2, we have the first *d* doubled.

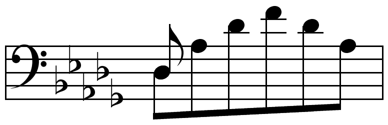

Figure 2. Lower staff of bar 65 of Chopin's Scherzo in B minor, op. 31

This can be rendered in SVL by resorting to chords[5]:

```
(d3/8\stup d\stdn\lnk{6}) \on{\stdn} a d+ f d a-
```

However, from a closer inspection of the context one might say that the doubled note belongs to a different voice, and therefore this is a case of semi-contrapuntal treatment of parts. The fact that bars in SVL may be omitted altogether (by using suitable numbers) makes it easy to have 'intermittent' voices for dealing with semi-contrapuntal treatments and also for contrapuntal music having sections with different numbers of voices (see an example of this below).

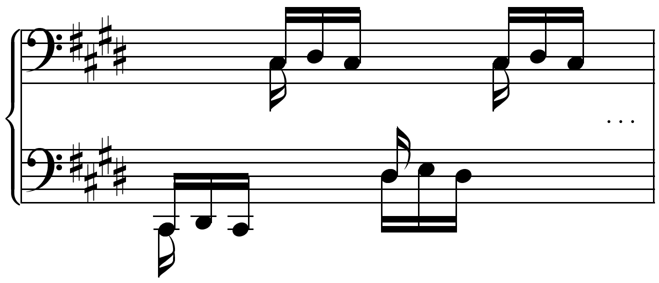

Figure 3. Beginning of bar 30 of Fauré's Prélude op. 103, 3

[5] Note that a chord is denoted by a sequence of notes/rests between parentheses. The \stup and \stdn commands control the orientation of note stems (either up or down).

Although in the Scherzo we prefer having separate voices, there are cases where it seems that the purpose of doubling notes is merely that of reinforcing them, and chords seem to be the right choice. For instance, the first notes in the right-hand part of bar 30 of Fauré's Prélude op. 103, 2 [12] in Figure 3 can be written as[6]

```
\hide{r/8t} (c3/16t\stdn c\stup\lnk{3}) d\stup c\stup
```

Chords (with notes of different durations) seem also to be appropriate to deal with holding of notes when the idea appears to be that of creating special sonority effects, as in the first notes of bar 1 of Brahms's Intermezzo op. 117, 2, [13] in Figure 4.

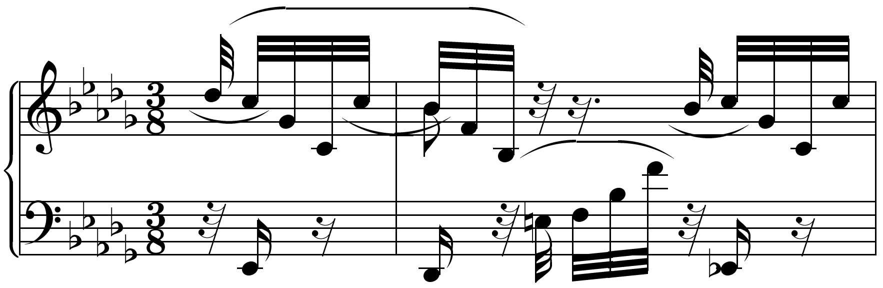

Figure 4. Beginning of Brahms's Intermezzo op. 117, 2

The first few notes in the upper staff of bar 1 may be written in SVL as

```
(b4/8\stdn b/32\stup\lnk{3}) f b-
```

A different kind of difficulty results from the need to assign symbols to each voice. Our first, maybe surprising, example is from *Die Kunst der Fuge* [14]. Figure 5 shows the lower staff of bar 89 of its *Contrapunctus V*.

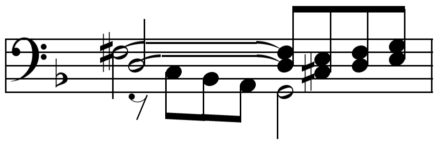

Figure 5. Lower staff of bar 89 of Contrapunctus V of the Art of the Fugue

The last 5 bars (86–90) of this piece do have (at least) 5 voices, the previous 85 bars having only 4 voices. An analysis with 5 voices in these bars implies that there will be chords (some of which with notes of different durations). In favour of this analysis there are the facts that parallel thirds are used and written with the notes tails linked, and that if chords were not allowed we would end up with 7 voices. Assuming that parallel thirds with tails

[6] \hide produces a space whose length is that of its argument; `8t` means a third eighth-note.

linked belong to the same voice, we have a note in the bass that is tied to a note in the tenor in bar 89; in SVL[7] we have for the bass

```
#f3/2\tieto{tenor} |
```

and for the tenor

```
d3/2\tie (d/8\lnk{4} f\tiefrom{bass}) (#c e) (d f) (e g) |
```

Probably the more problematic cases appear in music for keyboard and result from the separation of notes between the two hands not being coincident with a separation between parts or voices. This is obvious in Figure 6 that shows bar 38 of Brahms's Intermezzo op. 117, 1 [13].

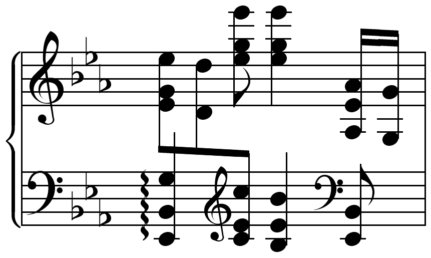

Figure 6. Bar 38 of Brahms's Intermezzo op. 117, 1

The reinstatement of the first theme begins in this bar. In the beginning of the piece we can identify three parts: the main theme, an accompaniment line in octaves on the right hand, and another accompaniment line on the left hand. This leads to the use of three SVL voices. But in the reinstatement the main theme alternates between the two hands and the accompaniment lines become fragmentary. The following is a quite satisfactory rendering of bar 38 in SVL.

```
\begvoice{main}{piano}
        \staff{1} \clef{g}{2} \key{-3} \time{6/8}
[38] \begslur \on{\stdn} (e4/8\lnk{3} g e5) (d4 d5)
   \staff{2} \clef{g}{2} (c4 e c5) (b3/4 e4 b)
\endslur
   \staff{1} \begslur \on{\stup}
   (a3/16\lnk{2} e4 a) (g3 g4) |
\endvoice
\begvoice{upper-acc}{piano}
        \staff{1}
[38] \hide{r/4} \on{\stdn} (e5/8 g e6) (e5/4 g e6) |
\endvoice
\begvoice{lower-acc}{piano}
        \staff{2} \clef{f}{3} \key{-3} \time{6/8}
[38] \on{\stup} \arp (e2/4 b g3) \hide{r/4.}
   \clef{f}{3} (e2/8 b) | \endvoice
```

[7] See [9] for a complete analysis of the 6 last bars.

Another example we selected for discussion is taken from the beginning of Brahms's Intermezzo op. 118, 1 [13]; see Figure 7.

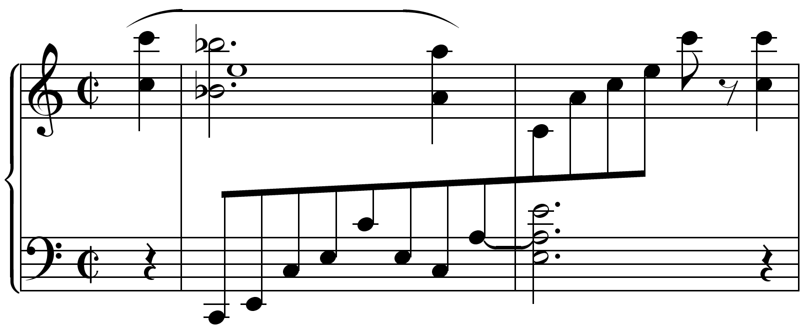

Figure 7. Beginning of Brahms's Intermezzo op. 118, 1

The question here is not only of how many voices one should use, but also how to deal with holding notes that seem to belong to different voices. In the SVL transcription in [9] we have three voices (main theme, intermediate accompaniment line, which alternates between the two hands, and a left hand accompaniment) and resort to the SVL commands for tying notes belonging to different voices. Another possible solution would have only two voices (one for each hand) in which case the SVL commands for continuing slurs and tail links from one voice to another ought to be used.

An analysis of many other keyboard (mostly piano) scores covering composers from Sweelinck to Shostakovitch did not reveal more complex problems than those just addressed. Even knowing that some ingredients are still missing in SVL, we remain confident about its potential as a good means for coding scores.

5 IMPLEMENTATION

In this section we describe the ongoing work on the implementation of ScEX. So far, we have implemented a graphic editor for SVL and the translator from SVL to SSL. We hope to have in the near future working versions of a translator from SSL to SVL, and then to start the work on the program to produce MusicTEX from SSL.

5.1 The Graphic Editor

The ScEX Graphic Editor (SGE) is intended only as a friendly interface with which the user may enter and modify ScEX documents faster than with a text editor. However, we are not aiming at a WYSIWYG system, and for the time being we will only address editing SVL documents.

The major design goals of the ScEX graphic editor were: simplifying the creation and editing of SVL documents, and avoiding syntactic errors by allowing only correct input. To achieve them we adhered to the following guidelines:

1. providing some formatting (at the bar level) so that the user may form an idea of the final output (but What You See Is NOT EXACTLY What You Get),

2. using interaction mechanisms as similar as possible to those of current word processors, with which we assume users are already familiar,
3. editing of symbols in the score by selecting them with a pointing device and using icons presented on the screen,
4. allowing input of symbols directly from the terminal keyboard, as a 'short-cut' to the use of pointing devices.

The harder problems in editing a score are related to changing the position of the insertion cursor, i.e., the point where the next bar element will be inserted. When formatting a score described by an SSL text, the insertion position depends just on the position of the previous notes and current settings. With a graphic editor the user may not only insert but also delete bar elements and use the mouse to move the insertion cursor to a place having different settings (like duration, or grouping).

Some of the implementation problems and solutions we have adopted are:

1. **The representation of each voice** is a sequential data structure for the bar elements in the screen and other information like the user-defined settings. This makes it possible to recover all the settings of an editing position when the cursor is moved around.
2. **The horizontal positioning of the insertion cursor** depends on the position and duration of the previous note and the position of notes that are in time with it in other voices. We use pre-defined minimum distances according to the duration and which are adjusted in view of notes in other voices. This may require changes in notes of voices other than the one currently selected.
3. **Visual vs. text editing:** as ScEX is based on tools that process text files, documents produced with SGE are saved as SVL texts. The information related to settings is annotated using SVL comments with a special syntax. The text files can be further edited using a text editor provided that the SVL and the annotations syntax is respected. In any event, the SGE parser will detect and report possible errors, allowing the user to correct them.

SGE's interface has 4 rectangular areas stacked vertically (Figure 8):

1. *command menus* that group the global functionalities of the editor, like file operations, general editing commands, and global settings,
2. *modifier tools* that are associated to pressable icons and are used to insert modifiers of a note or bar,
3. the *insertion area*, a scrollable rectangular area where the bars are displayed,
4. the *keyboard tool* representing four of the eight octaves of a keyboard; the middle *c* of the keyboard can be set using a separate tool and pressable icons are used to fix duration and create links and chords.

The present implementation was done in `Tcl` using the `Tk` graphical toolkit for X-Windows. We selected them for their simplicity of use, the possibility of rapid prototyping they allow, and their availability on different platforms.

5.2 The Mixer

We now turn to the ScEX Mixer (SM), the translator from SVL into SSL. Although our aim is to make an implementation in the C programming language (or some similar language),

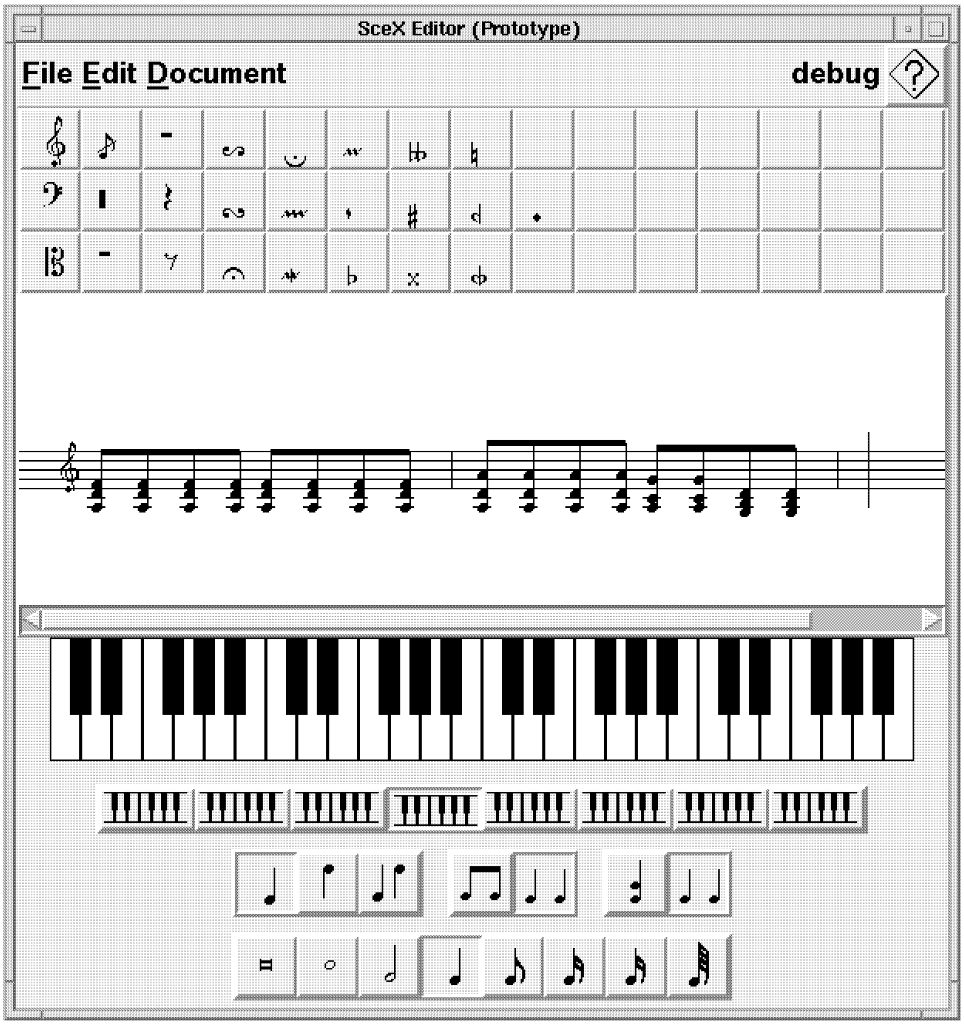

Figure 8. The ScEX Editor screen

we started by using Prolog which has clear advantages in what concerns rapid-prototyping and testing of ideas and alternatives.

We built an SVL parser, using a Definite-Clause Grammar (DCG) written from the SVL grammar in [9], which produces a representation of the SVL document as a Prolog term in a 'Prolog Internal Representation' language (PIR). The kernel of SM translates this language into a 'SSL Internal Representation' language (SIR), by picking elements from the voice descriptions in SVL and forming SSL voice contributions, which must be (partially) ordered with respect to time. The ordering is done by using a 'next-event scheduling' simulation technique. Another DCG generates SSL from SIR. We plan to use the same DCGs in the translation of SSL into SVL, taking profit from the fact that a Prolog implementation of a DCG may be (in certain cases) used both as a parser or a generator for a language.

REFERENCES

1. D.E. Knuth, *The TEX Book*, Addison-Wesley, Reading, MA, 1984.
2. L. Lamport, LATEX*: A Document Preparation System*, Addison-Wesley, Reading, MA, 1986.
3. J.S. Gourlay, ‘A language for music printing’, *Communications of the ACM*, **29**(5), 388–401, (1986).
4. F. Jalbert, ‘MuTeX User’s Guide (Version 1.1)’, Technical report, Computer Science Department, University of British Columbia, (1989).
5. D. Taupin, ‘MusicTeX: Using TEX to Write Polyphonic or Instrumental Music (Version 4.98)’, Technical report, Laboratoire de Physique des Solides, Centre Universitaire, Orsay, (1993).
6. B. Vercoe, ‘Csound — A Manual for the Audio Processing System and Supporting Programs with Tutorials’, Technical report, Media Lab, M.I.T., (1986, rev. 1992).
7. E. Foxley, ‘Music — A language for typesetting music scores’, *Software — Practice and Experience*, **17**(8), 485–502, (1987).
8. P.S. Langston, ‘Unix music tools at Bellcore’, *Software — Practice and Experience*, **20**(S1), S1/47–S1/61, (1990).
9. M. Filgueiras and J.P. Leal, ‘A Preliminary Formulation of SceX: A Music Typesetting System’, Technical report, LIACC, Universidade do Porto, (1993).
10. Wolfgang Amadeus Mozart, *Klaviersonaten, Band II*, volume 2, G. Henle-Verlag, München, 1977. ‘Urtext’, ed. E. Herttrich.
11. Fréderic Chopin, *Scherzi*, volume 279, G. Henle-Verlag, München, 1973. ‘Urtext’, ed. E. Zimmermann.
12. G. Fauré, *Complete Préludes, Impromptus and Valses-Caprices*, Dover Publications, New York, 1988. Reprint from first publications.
13. Johannes Brahms, *Klavierstücke*, volume 36, G. Henle-Verlag, München, 1976. ‘Urtext’, ed. M. Steegmann.
14. Johann Sebastian Bach, *Die Kunst der Fuge*, volume 423, G. Henle-Verlag, München, 1989 (1750). ‘Urtext’, ed. D. Moroney.