Abstract

Commercially available editors for musical scores (all) spot in providing a conceptual level of musical concepts familiar to a musician or composer. This paper presents a solution to the problem of representing deep musical knowledge for a knowledge-based editor of musical scores. The editor assumes 2 augmented and embedded in an object-oriented programming environment. We utilize three conceptually different types of explicitly represented knowledge: (1) display a particular musical symbol, e.g., the peripheral representation of a note, (2) the knowledge that determines the behavior at this symbol during an editing task, e.g., constraints resulting possible locations of the symbol within the score, and (3) to control some aspects of a musical object that are independent of any visual representation of the object, e.g., a note fulfilling a certain role within a chord.

1 Introduction

There are several levels of structure within musical information (cf. [3,14]). At the lowest level there is the bio- or technical representation of a sound on a piece of music in a sample. At the next level of structural information there are attacks of notes, rests, accents, dynamics etc. An automatic transcription system will have to reach this level of abstraction. Even higher there is the level of more complex concepts like a suspended dominant seventh chord which always resides on an accented beat etc. On the level it is not enough to recognize the adequate pitches within the current scale (i.e., the diatonic or melodic spelling) and to qualify this chord as a tonic, disregarding its interrelationships with surrounding concepts as there are accent, basic tone, preceding and succeeding harmonic function etc.

Musicians and composers mostly deal with concepts of the latter kind when performing or creating their music. They do not think of their music in terms of put this quarternote at that position, then put this eightnote at that position and put a single flat left of it, etc. A musician would like to express him- or herself as: I think of a meaning that is accompanied by a framework of the harmonic function up to D T of a major scale (cf. [12]).

None of the existing (commercial) music editors for conventional musical notation (CMN) supports the manipulation of any of these higher level concepts (cf. [12]). What is really manipulated in CMN editors is just the graphical representation of the music that is composed. Using an electronic music editor is still hardly any different from scratching the surfaces of copper plates.
The prototypical editor AMUSE described in this paper is part of the Composers Assistant project. This project attempts to build an experimental environment within which high level communication between musicians and computer's support is possible. We have built the editor as a first application to test our ideas about representational issues of musical concepts.

2 The role of musical concepts

2.1 Is moving bitmaps all that there is to composing?

We do not think that a graphic editor for any musical notation (e.g. CMN, piano roll notation, tabulates) should reflect only the different (graphical) views on the musical object the musician has in mind. A note is just a view of a musical object called tone, which may have interrelationships with other musical objects, e.g. an interval relationship with other tones. A musician knows about these relations when she is working on a piece of music. An editor, therefore, has to reflect this knowledge in one way or another. Take for example:

As a composer, it may come to your mind to transpose a chord to yield another mixture of tones. To do this you wouldn’t want to be forced to change the key or to give the transposition factor in terms of half steps; you just would like to say “transpose the chord a major third upwards”—and have all sharps and flats adjusted automatically.

It makes a great difference to work that way rather than to move the chord first and delete and add the accidentals separately afterwards. Therefore, we need a further visualization independent representation of musical objects.

Everybody who seriously deals with music is aware of the difference between musical phenomena and written music. The notation of music is just a means of archiving the most obvious constituents of music to enable later playing etc. To cast it in computer science terms: “Playing” music is equivalent to “interpreting” musical notations. Musical notions are tightly related to the evolution of musical concepts. They are used both by composers to compose music and by musicologists to describe and analyze musical concepts. The taxonomy of notions establishes a standard which enables us to speak about music [2]. Everybody knows the meaning of a fundamental six-four dominant chord or a major seventh chord. However, not all musical concepts are made explicit by musical notations; the graphic of a CMN of a piece of music is just a visualization of a subset of the musical concepts that constitute the piece. The semantics of the entire score compose the whole of the underlying musical concepts which can be brought to life by playing the music.

2.2 Musical notation and musical concepts

Every musical notation, e.g. the CMN, is just a means for thinking about and communicating musical concepts. A musical notation, especially CMN, constitutes a standard, a language. In everyday musical work it is a means for handling musical concepts. Notations

The work described here was partly made possible by a grant of the Deutsche Forschungsgemeinschaft (DFG) to the first author.
as well as concepts are the product of historical evolution, sometimes accidental, occasionally idiosyncratic, and as such do not serve all purposes equally well. However, to replace a widespread notation like CMN by a stricter and more consistent notation like Equitone is usually rejected because of the lack of acceptance by the broad musical community.

To add more power to a music editor, we link these parts of the graphic representation with their underlying musical concepts that could be represented by musical notations, e.g. a note within a given context (i.e. key signature etc.) represents a tone with a specified pitch, duration etc. The user then may either manipulate graphic symbols directly or manipulate the musical objects they stand for. A musical object takes care of its graphic display; thus sending a tone the message transpose major third upwards first changes the pitch of the tone and then adjusts the graphic display. This enables us to have different visualizations of a single musical object, e.g. a melody represented on different staves for a soprano, alto, tenor and bass voice. (If you then change the melody for example, all four graphic representations will be changed.) We believe that this will facilitate a musician's work in handling musical concepts within a music editor.

2.3 The representation of musical concepts

Musical concepts may be naturally represented as objects in an object-oriented programming language and may easily be organized into taxonomies (cf. [6,7,8,9]). Our current implementation is restricted to a subset of musical concepts, most of them relate to the pitch domain (e.g. tone, intervals, chords). The restriction is justifiable on the relative closeness of this subdomain. Figure 1 displays a selected portion of the hierarchy of concepts.

A considerable amount of musical knowledge is represented within the objects representing the concepts. The detailed structure of an instance of the class Tone is shown in Figure 2. A tone consists of a pitch, a duration etc. which are objects too. It responds to messages like: transpose yourself, who is your pitch? what is your duration? what is your...
pitches pitch-class? what is the size of an interval related to a second tone? etc.

3 AMUSED - A music editor for conventional musical notation

AMUSED follows a hybrid paradigm; it is a cross between an editor based on the WYSIWYG principle and a knowledge representation language. It supports editing operations on the notational level (e.g., adding and deleting of notes), as well as on the conceptual level (e.g., asking a group of notes for its properties or them to change their musical qualities).

Figures 3 - 7 show AMUSED in operation. Selected staff objects are displayed in reverse-

Figure 3: A note is asked for its pitch.
3.1 How much does a music editor need to know?

Dennis Byrd has pointed out [2] the difference between text- or graphic editors and music editors with respect to the semantics of the editor operations. A music editor needs much more semantic knowledge than a traditional text- or graphic editor. There is a delicate tradeoff though: too few domain-specific semantics of a pure graphic editor force the user to create an empty staff by drawing each of its 5 lines separately; too much semantic knowledge reduces the range of tasks the editor may be successfully used for: an editor with a fixed number of sheets only, e.g., piano-only-sheets and a lead-sheet cannot be used for work on a piece of music for chamber orchestra. The same holds for defining key signatures: when editing traditional, tonal music it may be reasonable not to place each accidental separately but to globally determine the key signature as c-minor and have the three accidentals set appropriately by the editor. However, the user does not want to be taken the freedom to place more than one accidental separately if the automatic placement routine based on key signatures for traditional scales does not put them at the positions s/he wants, e.g., key signature in B. Bartok's microcosmos.

Semantics added to editor operations aid in focusing on the hot spot of current work, provided there is a match between what the user thinks is the hot spot and what the system...
Figure 5: A chord is asked for its chord type.

Figure 6: A chord is asked for its interval structure.

Figure 7: A chord is asked for its chord symbol.
thinks it is. Unfortunately, the appropriate amount of semantic knowledge to employ is task- and user dependent. A music typist will probably prefer to have the exact horizontal position of each note determined by the editor for adequate alignment of a multi-voiced piece of music, whereas an amateur pianist may need the individual, horizontal labeling of musical events to denote entry delays.

To answer the question posed: a music editor should know as much as possible in order to be able to support the user in editing tasks. However, it must be very careful about when to bring the knowledge into play in order not to unnecessarily restrict the actions of the user.

3.2 Where to put the semantic knowledge of editing operations?

3.2.1 The display list

The knowledge required for the (re-)display of a musical score is usually held on what is called a display list. For efficiency reasons only, not each occurrence of a music symbol on the screen has a private object counterpart. Moreover, the instance crochets of a class note-class for example supervises the editor related aspects of all crochets in the score and is referred to by each crochet on the screen via the object-reference. Because the position and other context-dependent information of each occurrence cannot be kept in a single object it will be left in the display list. The central display method of the editor traverses the display list and delegates the display message to the reference objects supplying additional information e.g. the position of the occurrence. The knowledge of how to display a clef or a note is thus relegated to the object-reference.

3.2.2 Reference objects modify semantics

It is not only for display that the editor delegates messages to the reference objects. Consider for example creating a clef on a yet empty staff. First of all the editor receives the create message invoked by the user; it then asks the reference object of the current selected occurrence (which is the staff to hold the clef) for objects which legally can be placed on a staff, eventually the user gets prompted to select one of these. The create message will be delegated to the reference object for the selected staff object type; i.e. the instance c-clef of class clef-class. c-clef takes care that the occurrence will be placed accurately, e.g. that the hot spot of a c-clef always resides on a staff line and not in between. This is one of the constraints controlled by the instance c-clef which, upon creation of a c-clef, affects the semantics of the create command.

3.2.3 Manipulation and maintenance of semantic behavior

Semantic behavior upon receipt of a message can be altered by simply changing slot values or by enhancing or replacing existent methods. E.g., it may not be useful to allow notes with an unlimited number of ledger lines because of readability. Therefore we might want to restrict each staff object to at most four ledger lines either above or below the staff. This can easily be done by changing the values of two slots namely range-top and range-bottom. Introducing a new staff object type often entails the creation of new classes. Initially there is only one class for all staff object types called staff-object-class which for example takes

ICMC Proceedings 1986 12
care that no staff object occurrence has more than 4 ledger lines on either side of the staff and implements the response to basic messages like create, delete, and select. The staff object type note, however, requires more semantic actions to be taken than are defined in staff-object-class, e.g., dynamical, real time creation of ledger lines when a note is edited. A new staff-note-class is created when a note is moved across the top or bottom line of a staff. This leads to, a new staff-class with staff-object-class as its superclass and an enhanced create method. By adding and exchanging classes and instances the editor is highly redefinable.

Figure 8 displays part of the inheritance lattice of reference object classes and some of their instances.

4 Conclusions

Our experience with AWEB has proven the feasibility of the object-oriented programming paradigm in representing deep musical knowledge and utilizing it for a knowledge-based musical editor that is able to overcome the deficiencies of pure graphical music editors. Future work will show how the approach can be extended to include higher level musical concepts like harmonic relationships.

References


173

ICMC proceedings 1988


