ABSTRACT

This chapter deals with pen interaction and its use for musical notation composition and editing. The authors first present existing pen-based musical notation editors and argue that they are dedicated to classical musical notations and are often constraining for the user. They then focus on their generic method that is in particular based on a formalism that models how to interpret the strokes drawn in online structured documents. To analyze an element, it models a coupling of a global vision (its position relatively to the other elements of the document) with a local vision (its shape and that of its components). The authors also present the hand-drawn stroke analyzer associated to this formalism. Finally, they demonstrate how the presented method can be used to design pen-based systems not only for classical musical score notations, but also for plainchant scores, drum tablatures and stringed-instrument tablatures.

INTRODUCTION

This chapter deals with a new computer system approach, based on pen interaction, for music composition and editing: thanks to a pen, the user creates and modifies digital documents by drawing on a touch screen. The use of a pen is very intuitive, because it reproduces the “paper-pen” metaphor, media everybody knows: it makes it possible to use computer systems the same way as a sheet of paper. Figure 1 presents a tablet PC, which is a computer system with such an interaction: the user writes musical scores in a traditional way by drawing the symbols on the screen. This system is an example of system developed thanks to the methodology presented in this chapter: the user draws musical
Pen-Based Interaction for Intuitive Music Composition and Editing

Figure 1. A tablet PC with a pen-based interface

symbols the same way as on paper. The drawings are recognized and retranscribed neatly directly as the user composes the document. The user can then check the recognition process and interact with the system to easily modify the document, for instance move some of its elements, erase them, and so forth.

Pen-based interaction offers various advantages for the composition and the editing of digital documents. Indeed, the user can benefit from the possibilities of both paper and computer systems. On the one hand, the user can write symbols as he usually does on paper. As stated by many authors, like for instance Anstic, Bell, Cockburn, and Setchell (1996), Macé and Anquetil (2006a) or George (2003), it is more user-friendly and faster than classical mouse-based composition and editing systems which consist in clicking on a menu to select a symbol and dragging it to the appropriate place. Moreover, as presented in Figure 1, a tablet PC is almost as mobile as a sheet of paper, which simplifies its use in various situations. On the other hand, computer systems offer a direct retranscription which is easier to read than hand-drawn symbols. The user can also easily modify the documents, erase or move some of their components, copy, cut and paste some of them, and so forth, and then avoid the loss of quality and neatness of the paper document.

Finally, the user benefits from the traditional computer functionalities; for example, a digital musical score can be played, perfectly formatted, archived, distributed, and so forth.

Actually, there are very few pen-based interfaces for digital document composition and editing, and in particular for musical notations. This is due to the fact that developing such systems is complex. Indeed, the recognition of symbols drawn on the touch screen is a difficult problem of pattern recognition due to the diversity of hand-writing styles. This recognition process is even more complex in the context of highly structured document analysis, since these documents are constituted of many elements of various natures. Moreover, a same drawing can have different meanings according to the context in which it has been realized, which must therefore be taken into account.

This chapter deals with the exploitation of pen interaction to compose and edit structured documents, and, more specifically, documents with musical notations. Before going further, let us introduce the vocabulary used in this chapter. We use the word composition for the action of writing music symbols, and the word editing for the action of modifying one or more of these symbols or more generally the document. We designate as structured documents, documents which present a predefined structural arrangement, and for which it is possible to express composition conventions, that is the way they are classically drawn. Concerned documents come from various domains, such as diagrams, plans, electronic figures, and so forth. Traditional musical scores, plainchant scores, drum tablatures or stringed-instrument tablatures are examples of structured documents with musical notations that we are going to focus on in this chapter. Online interpretation deals with the interpretation of the hand-drawn user strokes, which are the sequences of points captured by the touch screen between a pen-down and a pen-up. Online interpretation can be either lazy (i.e., occurring only when explicitly
Pen-Based Interaction for Intuitive Music Composition and Editing

requested by the user) or eager (on the fly, that is occurring directly while the user is drawing). As presented on Figure 2, lazy interpretation offers the advantage of not intruding into the user’s creative phase during the composition because he can write everything he has in mind before asking the system to interpret the document (Nakagawa, Machii, Kato, & Souya, 1993). Nevertheless, once the recognition process applied at once on all of his strokes, he has to examine the entire document to look for possible incorrect interpretations. Besides, it turns out that lazy systems are so far not robust enough and make too many mistakes, which reduces their usability. We believe that lazy recognition is a promising approach to offer unconstrained understanding of ink, but the difficulties to design automatic parsing coupled with a robust hand-drawn shape recognition system show that it remains an open problem.

Eager interpretation is then another way to consider online structured document analysis. As presented on Figure 3, every time the user draws a stroke, the system interprets it immediately (Blostein, Lank, Rose, & Zanibbi, 2002; Macé, Anquetil, & Coüasnon, 2005a); it then has to deal with documents directly during their composition and to make a decision as quickly as possible so that the user does not have to wait to continue his drawing. Eager interpretation allows exploiting the interaction with the user, who is aware of the system answers, and can then validate or refute them progressively. We believe that eager interpretation is the most pertinent compromise for online structured document interpretation because systems based on this kind of interaction are often more robust and more efficient than lazy ones, which makes them more usable. The work we present aims at exploiting a generic approach in order to eagerly interpret various forms of hand-drawn musical notations and various composition conventions.

Whereas the existing approaches only enable users to draw classical musical score notations, our approach aims at making it possible to easily adapt the system to different ones. In this chapter, we focus on the composition and the editing of various musical notations, such as classical musical scores, plainchant scores, drum tablatures and stringed-instrument tablatures, but the method is generic and has already been exploited for two others structured document domains, which are diagrams and Unified Modelling Language (UML) class diagrams.

In the following section, we present a state of the art in pen-based interface development in the domain of musical notation composition and editing. We first state that, so far, systems are dedicated to classical musical scores. We emphasize, on the one hand, their differences with traditional composition conventions, and, on the other hand, their limitations in term of musical notations. In section “A Generic Approach for Hand-Drawn Structured Document Composition and Editing”, we introduce our generic approach and present each of its components. In section “Pen-Based Systems for Music Composition and Editing with the Presented Approach”, we present how this generic approach can be used to compose and edit four different natures of musical structured documents: classical musical scores, plainchant scores, drum tablatures and stringed-instrument tablatures. Finally, we conclude this chapter by highlighting future trends in the exploitation of pen-based interaction for musical notation composition and editing and by presenting our future works.
STATE OF THE ART IN PEN-BASED MUSICAL SCORE COMPOSITION AND EDITING

Before going further, it is interesting to notice that, so far, there are little pen-based software for musical notation composition and editing. In particular, as far as we know, only systems for classical musical score notations exist.

In this section, we first present the main difficulties that make pen-based musical score editors complex software to develop. We would like to note that these problems also exist in other pen-based structured document composition and editing software. Then, we present the existing approaches and highlight the way they deal with each of these problems.

Difficulties in Developing Pen-Based Musical Score Editors

The two main difficulties in the development of pen-based musical score editors are the interpretation of the user strokes in the context of structured documents and the management of symbols drawn with more than one stroke.

Interpreting the Strokes of Structured Documents

Musical scores contain a lot of symbols of various natures, such as clefs (G-clefs, C-clefs, F-clefs, etc.), notes (whole-notes, half-notes, quarter-notes, etc.), accidentals (flat, sharp, natural, etc.), figures (time signature, etc.), characters (dynamics, lyrics, etc.). Powerful hand-drawn shape recognizers can be developed to interpret these symbols (Plamondon & Srihari, 2000), but we have to take into account the fact that the more symbols a recognizer has to discriminate, the less robust and efficient it is. Thus, it is not possible to use a unique recognizer for all symbols a musical score can contain. It is then necessary to exploit dedicated recognizers, that is recognizers able to interpret subsets of these symbols, for instance one recognizer to discriminate between clefs, one to discriminate between accidentals, and so forth. As a consequence, the first difficulty consists in interpreting the user hand-drawn strokes, which is furthermore complicated by the fact that every scripter has its own way to realize a musical score and draw the musical symbols. Finally, the fact that some of these symbols have the same shape (for instance a whole note, a half note, the figure “0”, the character “o”) complicates even more the problem. It is then essential to take into account the context in which a stroke has been drawn in order to interpret it: depending on the context, which can be for instance structural and/or temporal, the same drawing is interpreted differently (for instance, a circle on a staff is more likely a head note, whereas a circle below a staff is more likely part of lyrics, i.e., the character “o”).

Dealing with Multi-Stroke Symbols

Some of the classical musical symbols cannot be, or are not traditionally, drawn with only one stroke: they are called multistroke symbols (in opposition to unistroke symbols). For instance, the sharp symbol is constituted of two horizontal segments and two vertical segments; as a consequence, such a symbol is classically drawn with four strokes. This presents a second difficulty for the eager interpretation process. Indeed, the system is faced with the dynamic segmentation problem: it has to eagerly decide if a stroke is sufficient to form a symbol, and interpret it as such, or if it should wait for the following strokes. Once again, the solution can be based on the exploitation of structural and/or temporal contexts.

Some authors avoid part of or all these problems, for instance by assuming that each musical symbol has already been isolated before trying to interpret it. Thus, George (2003) proposed to exploit artificial neural networks in order to recognize already segmented symbols. But the use of
this method in an online musical notation editor is not straightforward because the segmentation of the symbols is a complex problem, even more with an eager interpretation process.

**Existing Pen-Based Musical Score Editors**

In order to deal with both of the problems presented in previous section, the main solution consists in changing the *gestures*, that is the way musical symbols are drawn, and to constrain the user to a new way of writing musical scores. Therefore, the authors often define new alphabets, which are designed to reduce the similarity between the different symbols in order to make their discrimination easier. These new alphabets are generally unistrokes, in order to facilitate the interpretation process: as a symbol is always drawn with only one stroke, the hand-drawn shape recognizers can be called each time the user draws a stroke. If an interpretation for this stroke is found, then it is replaced by the corresponding musical symbol. If no possible interpretation is found, then the system has been unable to recognize the stroke: it is *rejected*. It disappears from the screen, and the user has to draw it again.

Naturally, the definition of new alphabets for musical symbol composition tends to find the best compromise between ease of recognition for systems and ease of learning for users. Nevertheless, this solution has a major drawback: the user has to get used to this new alphabet, and does not write musical notations the same way as on paper. We believe that this creates a gap between the musician and the computer which often implies, for instance, a rupture in the creation process in comparison with the use of paper.

One of the first pen-based systems was proposed in 1996 by Anstice et al. and then enriched in 1998 by Ng, Bell, and Cockburn; it was based on this idea. The authors exploited the three primary criteria proposed by Goldberg and Richardson (1993) for their pen-based text input system, which are: *ease of learning*, *high distinction between the input symbols* and *fast writing speed*. Actually, one of the goals of the proposed system, called “Presto”, was to make it possible to write music in a faster way than on paper thanks to the use of pen interaction and the definition of new faster gestures to replace the classical way to draw musical symbols. As a consequence, the gestures are also thought to be as small as possible. Of course, in order to facilitate their learning, the new gestures are as intuitive for musicians as possible, but are still different from the classical ones. For instance, the gesture to draw a filled-note head is a dot. In order to add an accidental to a note head, the user has to draw a small vertical segment starting on this head, uprising for a sharp, descending for a flat. In the same manner, a durational dot is added to a note head by drawing a small horizontal segment starting on this head. The second column of Table 1 presents some of the gestures of this system.

Forsberg, Dieterich, and Zeleznik presented in 1998 a pen-based musical score editor, called the “Music Notepad”; a more recent version of this composition system is available on the internet (Forsberg, Holden, Miller, & Zeleznik, 2005); the gestures we explicit in this paragraph and in the third column of Table 1 corresponds to this last version. Once again, the authors exploit a new alphabet to write classical musical notations, which is in fact quite close to the one proposed by Anstice et al. (1996) and Ng et al. (1998). For instance, quarter-notes can be drawn almost the same way as on paper, but accidentals can not: a gesture starting on a note head and uprising on the right is interpreted as a sharp, descending on the right as a flat. A durational dot is added by tapping on the note head.
We proposed a pen-based system dedicated to the editing of musical scores (Macé, Anquetil, Garrivier & Bossis, 2005). In order to facilitate the recognition process, we used a unistroke alphabet, but we just changed the symbols that cannot be draw with only one gesture. The goal was to keep gestures as close as possible to the classical ones in order to make a more user-friendly and more usable system and to limit as much as possible the rupture in the creation process. For instance, a durational dot is drawn as a small dot on the right of a note head; an accidental is drawn close to a note head on its left: a flat is drawn with one stroke the same way as on paper, but the gestures for the sharp and the natural are changed, for a horizontal segment and a vertical segment respectively. We also proposed different gestures for a same symbol when it seemed appropriate. Thus, a filled-note head can be done in several ways, for instance by drawing a dot, a slash, a scribble, and so forth. The originality of our approach was to exploit the structural context of a stroke to interpret it. For example, a stroke drawn close to a note head on its left has a high probability of being an accidental; then, instead of trying to recognize any possible symbol, we try to recognize an accidental. Actually, we kept some of the concepts of this system to design the generic approach we present in the continuation of this chapter. The fourth column of Table 1 presents some of the gestures of this system.

In 2004, Miyao and Maruyama, and then Mitobe, Miyao, and Maruyama, presented an online musical score recognition system, which is able to deal with multistroke symbols. As far as we know, with the exception of the work we present in this chapter, there is no other system with such a capability. The gestures they propose are almost the same as the classical one, with one exception (a filled-note head is drawn with a circle with a slash in order to reduce the input time). The analysis process consists then in two steps. In the first one, each user stroke is interpreted as a primitive form, that is as one of the unitary symbols that can exist in musical scores: they are all the unistroke symbols (whole-notes, half-notes, clefs, flat, etc.) and all the components of the multistroke ones (horizontal and vertical segments, which can be part of a sharp, etc.). In the second step, the system tries to combine the primitive forms to produce multistroke elements. Although the system they present does not propose enough available symbols to be usable, we believe that their approach is interesting. Actually, some of the concepts we present in the following section are close to some of these authors. The fifth column of Table 1 presents some of the gestures of this system.

In the next section, we introduce the generic approach we designed to develop pen-based structured document composition and editing software. We have in particular exploited this.
approach to develop a system for classical musical score notations, which aims at letting the user compose musical symbols the same way as on a sheet of paper. The gestures are then as classical as possible, as presented in the sixth column of Table 1. Figure 4 presents two screenshots of the editor: on the left, the user draws a beam between two quarter-notes, and on the right, this drawing is interpreted and retranscribed neatly.

Screenshots of the pen-based musical score editor developed with the presented methodology: on the left, the user draws a beam between two quarter-notes, and on the right it is replaced by its neatly retranscribed symbol thanks to an eager interpretation process.

Table 1. Alphabet of existing pen-based musical score editors; when the direction of a gesture is constrained, a pen indicates its last point

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Apparently not available</td>
<td>Apparently not available</td>
<td>Not available</td>
<td>Not available</td>
<td></td>
</tr>
</tbody>
</table>
A GENERIC APPROACH FOR HAND-DRAWN STRUCTURED DOCUMENT COMPOSITION AND EDITING

In this section, we present a new generic approach that aims at interpreting hand-drawn structured documents from various domains and with different constraints, such as musical scores, but also diagrams, plans, electronic figures, and so forth. The architecture of the system is illustrated in Figure 5. It is based on a framework constituted of three main components:

- **A formalism for eager interpretation of hand-drawn structured documents** (1), which models how to eagerly interpret the elements of hand-drawn structured documents from a given domain. It permits to write interpretation rules that represent composition conventions (for instance chronological information: it models which element can or must be drawn before another one) and physical information, such as the spatial structure of the document. Finally, they model the driving of the use of hand-drawn shape recognizers depending on the document structural context of a stroke, that is depending on its location in relation to the other elements of the document; this is its main originality. This formalism can be applied to different notation domains and composition conventions, in order to model how to eagerly interpret a very large panel of hand-drawn structured documents. The interpretation rules written for one domain according to this formalism are externalized from the system: they are easily modifiable. They just need to be transformed by the generic compiler of the system. This formalism is presented more in detail in the first subsection.

- **A hand-drawn stroke analyzer** (2), which exploits the knowledge modeled by the formalism to eagerly interpret hand-drawn structured documents. It is able to call the pertinent hand-drawn shape recognizers (for example clef recognizer, accidental recognizer, etc.) depending on the structural context of the stroke, and then update the document contexts that will help recognizing the following strokes. This analyzer is generic: it does not need to be adapted to each specific domain. This analyzer is presented more in detail in the second subsection.

- **A set of graphical functions and pen-based editing functions** (3), which can be exploited by any pen-based system; such functions deal with the user interface and the display of the document. They exploit graphical information, such as images of the neatly retranscribed symbols, which are externalized because they are dependent on the domain. These functions also deal with pen-based interaction and editing: the user can for instance select graphical elements, move them to another part of the document, erase them, and so forth. It is also possible to undo the last element, zoom in or out, save or load a document. All these actions can be done just using the pen. These functions are presented more in detail in the last subsection.

The graphical and pen-based editing functions and the hand-drawn stroke analyzer are *domain-independent components*: they do not need to be rewritten for each pen-based system. The development of such systems with our approach only requires realizing the *domain-specific components*: writing the interpretation rules (embedded under a compiled form), designing the necessary hand-drawn shape recognizers (which can be reused from one system to another) and specifying the graphical information. The components of our system are well separated and can be modified and adapted independently of the others.
A Formalism for Eager Interpretation of Structured Documents

The main component of our approach is a formalism which aims at defining how to eagerly interpret the elements of structured documents from various domains. The goal is to propose a formalism which is as generic as possible in order to be able to deal with a large panel of documents. We previously (Macé et al., 2005a) defined the four basic concepts that are, in our opinion, associated to the modelling of eager interpretation of online structured documents:

- **The modeling of the composition conventions**: As the interpretation process is eager, the system has to deal with incomplete documents; a way to do so is to model how a document is typically drawn, for instance, which element after which other.
- **The representation of the document spatial structure**: It is necessary to exploit the relative positioning of the document elements in order to model, on the one hand, in which document structural contexts an element can be identified, and, on the other hand, which of these contexts are generated due to the creation of an element.
- **The driving of the recognition process by the document context analysis**: We have stated that it is not possible to have a unique recognizer for all the symbols that a document can contain. By exploiting the document context of an element, the system must reduce the likely symbols and choose which dedicated recognizers to use.
- **The pen-based human-computer interaction**: As the result of the interpretation process is directly displayed on the screen, the user is aware of it. Consequently we can take into account the human-computer interaction and exploit it on an as user-friendly way as possible.

The formalism we propose takes all these concepts into account. We present its syntax and explain it on two classical musical score notation examples, which are the filled-note head and the sharp.

The description of the interpretation of a structured document with our formalism is composed of interpretation rules which define the generating of the element which name they bear. Several

We present these components more in detail in the next subsections.
rules can have the same name, which makes it possible to model different ways to compose the same element. A rule takes a set of elements as parameters, enounced in its heading, and returns a new one that can replace them; the parameters are the components, or sub-elements, of the new element. A parameter can be either a stroke or an already interpreted element, which makes possible a hierarchical shape description. For instance, as presented in Figure 6, a filled-note head is typically drawn with one stroke, whereas a sharp is typically drawn with two horizontal segments and two vertical segments. The corresponding rule headings can then be:

**FilledNoteHead (Stroke s)** ...

**Sharp (HorizontalSeg hs1, HorizontalSeg hs2, VerticalSeg vs1, VerticalSeg vs2)** ...

The structure of a rule is composed of four blocks: a document context verification (DCV) block, a shape context verification (SCV) block, a shape recognition (SR) block and a document context creation (DCC) block, in the following order:

**SymbolName (Parameter 1, ..., Parameter n)**

**Document Context Verification block (DCV).**

**Shape Context Verification block (SCV).**

**Shape Recognition block (SR).**

**Document Context Creation block (DCC).**

DCV and DCC blocks enable a global vision of the document in order to define in which document structural contexts an element must be located. A DCV block specifies the document structural contexts in which the symbol created by the rule has to be identified, whereas a DCC block indicates the contexts that are generated due to the creation of this element. The SCV and SR blocks enable, given a document context, a local vision of the element to recognize; it distributes the recognition process among local constraints, formalized in the SCV block, and recognizers, formalized in the SR block; it drives the interpretation process by calling dedicated recognizers depending on the context of elements.

The proposed formalism is based on the definition of structural contexts which model, on the one hand, specific locations in the document, and, on the other hand, which elements can or must exist at these positioning. The syntax of a structural context is as follows:

\[ R[position, part]A. \]

This means that the involved structural context is located at the relative positioning \( position \) (e.g., in, on the left, above, etc.) of a reference \( R \). In order to satisfy this context, an element \( A \) must have its part \( part \) (e.g., one point, all the points, the first point, the highest point, etc.) in this positioning.

**Document Context Verification block (DCV)**

The document context verification block specifies a list of document structural contexts in which the symbol has to be identified; they thus have to be verified by the parameters of the rule. Its syntax is as follows:

\[ DCV: R[position, part]A, ..., R[position, part]A_n. \]
As introduced previously, this means that at the relative position \( \text{position} \) of a reference \( R_i \), the part \( \text{part} \) of an analyzed element \( A_i \) exists. In this block, \( R_i \) and \( A_i \) do not have to be part of the parameters.

For example, a sharp can be drawn on the left of any note head that does not already have an accidental; this means that the four segments forming this symbol must be drawn on the left of a note head. A DCV block for such an interpretation rule can be written as (as seen previously, the parameters of the sharp interpretation rule are two horizontal segments \( hs1 \) and \( hs2 \) and two vertical segments \( vs1 \) and \( vs2 \)):

\[
\text{DCV:} \quad \\
\begin{array}{c}
\text{hs1} \quad \text{vs1} \\
\text{hs2} \quad \text{vs2}
\end{array}
\]

An alias is defined to specify that a \textit{Head} can be a \textit{WholeNote}, a \textit{HalfNote} or a \textit{QuarterNote}. We would like to note that in this block, we use a saving operator \( \rightarrow \) to save an element and reuse it afterwards. Here, it aims at modelling that the four segments forming the sharp symbol must be on the left of the same head, denoted by \textit{head}.

\textbf{Shape Context Verification Block (SCV)}

Once we know that the symbol that the rule generates can exist in the document structural context in which the element is located, we locally focus on its shape in order to check if it corresponds to this symbol. The shape context verification block is a first step: it models local structural constraints about the parameters of a rule in order to interpret an element constituted of several sub-elements: it makes it possible to identify which part of the new element corresponds to each parameter. The syntax is the same as in the DCV block, but this time \( R_i \) and \( A_i \) must be parameters. By default, there is no constraint on the drawing order of the elements; if necessary, a chronological operator can be used.

For example, a sharp is constituted of four segments which can be drawn in any order. A SCV block (in which the alias \text{[intersects]} is used for the structural context \text{[in,one]}) can be written as:

\[
\text{SCV:} \quad \\
\begin{array}{c}
\text{hs1} \quad \text{vs1} \\
\text{hs2} \quad \text{vs2}
\end{array}
\]

These local structural constraints are not necessarily enough to ensure that the four segments constitute a sharp. So, we may want to exploit a classical hand-drawn shape recognizer; this can be done thanks to the SR block.

\textbf{Shape Recognition Block (SR)}

The shape recognition block is the second step of the local interpretation of the analysed element shape: it corresponds to the call to a hand-drawn shape recognizer. It makes it possible to invoke only the relevant recognizers, depending on the context of an element. It is essential to increase the interpretation process robustness, since the less symbols a recognizer must interpret, the more efficient and the more reliable it is. The SR block syntax is:

\[
\text{SR:} \quad \text{SymbolFamily}, \{\text{AcceptedAnswers}\}.
\]

This expression means that the recognizer of \textit{SymbolFamily} is called, with the rule parameters as input. If its answer is included in \{\text{AcceptedAnswers}\} the recognition process is a success. If no \{\text{AcceptedAnswers}\} is specified, then any answer of the recognizer is acceptable. The order of the elements presented to the recognizer is the order of the parameters in the declaration of the rule; so it is always the same. As a result, its work is relieved, because it has to interpret the elements always in the same order.

For example, to interpret a sharp, we exploit a recognizer able to recognize accidentals, and
we want its answer to be a sharp. This way, the coupling of the global and the local visions model that it is more pertinent to call an accidental recognizer than a stem recognizer or a dot recognizer on elements located on the left of a note head. A SR block modelling this can be written as:

\[
\text{DCC: } R\{\text{position}_i, \text{part}_i\}A^{m_i}_i, \ldots, R\{\text{position}_n, \text{part}_n\}A^{m_n}_n
\]

This means that at the relative position \text{position} of an element \(R_i\), the part \text{part} of an element \(A_i\) can exist. The current element, in process of creation, is referenced as “this”. The number \(m_i\) indicates how many \(A_i\) can exist in this context and is \(^*\) if there is no limit.

For example, the creation of a sharp on the left of a note head does not imply the possibility to draw a specific symbol. On the contrary, the creation of a note head allows drawing one accidental on its left, one durational dot on its right and one stem, either uprising or descending; then, document structural contexts modelling where these symbols can now exist are created and will help in interpreting them. A DCC block for such a rule can be written as:

**Table 2. Interpretation rules for quarter-notes and sharps**

<table>
<thead>
<tr>
<th>Interpretation rules</th>
<th>Structural contexts before recognition</th>
<th>Structural contexts after recognition</th>
<th>Re-transcribed symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>FilledNoteHead (Stroke s)</td>
<td>DCV: Staff [on,all] s. SCV:</td>
<td>SR:</td>
<td>HeadNotes {FilledNoteHead 1}. DCC: this[all]Accidental 1. DCC: this[right,all]Down 5. this[left,all]Updown 5. this[right,all]DownThis 5.</td>
</tr>
<tr>
<td>Sharp (HorizontalSeg hs1, HorizontalSeg hs2, VerticalSeg vs1, VerticalSeg vs2)</td>
<td>DCV:</td>
<td>SCV:</td>
<td>SR: Accidentals {Sharp}. DCC:</td>
</tr>
</tbody>
</table>

We would like to emphasize that the accidental document structural context created in this DCC block corresponds to the one exploited by the DCV block of the sharp interpretation rule defined previously and to note that disjunction between contexts is possible. Finally, an alias is defined to specify that an Accidental can be a Flat, a Sharp or a Natural.

The complete QuarterNote and Sharp interpretation rules are given in Table 2.

As presented in previous section, our goal is to design a pen-based musical score composition software as close as possible to classical notations, in order to be as user-friendly and as usable as possible so as to limit the rupture in the creative process. Thus, in the presented rules and figures, the gestures are the same as the classical ones. We would like to highlight the possibility, with this system, to model different ways to draw the same symbol. This implies that it is possible, if necessary, to use new alphabets such as the ones presented in previous section and in Table 1.

**A Stroke Analyzer**

The second component of the system is a hand-drawn stroke analyzer that eagerly interprets the
Pen-Based Interaction for Intuitive Music Composition and Editing

drawings of the user thanks to the knowledge modelled by the formalism. In this section, we first explain how we deal with multistroke symbols. Then, we present the analysis process more in detail, and finally highlight the exploitation of the human-computer interaction.

An Incremental Interpretation Process to Deal with Multi-Stroke Symbols

The system we propose interprets the user strokes dynamically and incrementally. This means that the analysis is applied every time the user draws a stroke, like in unistroke symbol systems. If no interpretation is found for a hand-drawn stroke, it is rejected. On the contrary, if it can be interpreted, that is if such a stroke can exist in such a context, it is replaced by its neatly retranscribed corresponding symbol and the system tries to associate it with other elements of the document in order to constitute multistroke symbols. Thus, the dynamic segmentation process is deduced from the formalism. For instance, the Sharp interpretation rule presented in Table 2 models that where a sharp can exist, two horizontal and two vertical segments can also exist; but once they do, they can be replaced by a sharp if their shape is coherent. This analysis process is close to the one proposed by Miyao et al. (2004) and Mitobe et al. (2004).

Presentation of the Analysis Process

The formal analysis process algorithm is given Figure 7, and it is illustrated by Figure 8. When a new stroke, or more generally a new element, is analyzed, the goal is to find the sequence of interpretation rules to apply. As introduced in previous section, an existing document is constituted, on the one hand, of the already interpreted elements it contains and, on the other hand, of its structural contexts. This knowledge defines a global vision of the document which must be exploited to interpret new elements. The analyzer identifies the document structural contexts in which the analyzed element is located and activates the corresponding associated rules.

Each activated rule is tested. The analyzer searches for possible parameters for the current rule that is, the elements of the document that can be associated to the analyzed element to constitute a more complex one. For each applicable rule, the analyzer verifies the coherence of its parameters with the structural context in which the new element should be located; this is modelled by the DCV block. Only the rules that satisfy both of these criteria remain.

The verification of the coherence of the new element with the document structural context thanks to a global vision means that the corresponding element can exist at this positioning; the next step consists in trying to identify the shape of the parameters of the rule. For that purpose, a local vision of these parameters is exploited to analyze their structural arrangement; this is modeled by the SCV block. A shape recognition system can be used on the parameters of the rule in order to identify their shape thanks to the SR block. The recognition systems that are exploited by our system have a particular characteristic: they have reject options, which means they do not give an answer unless their confidence in it is high enough (Mouchère & Anquetil, 2006). The advantage is to filter possible interpretations, and to prevent from displaying an answer which has a high probability of being wrong. The rules that satisfy the SCV and SR blocks remain: they are applicable.

As more than one rule can be applicable, a Rule Selection component is exploited to make a decision. This component evaluates the degree to which the parameters of each rule belong to the document structural contexts and selects the rule with the higher degree. If the difference between the degrees of the best solutions is negligible, the analyzer decides that it is unable to make a decision and rejects the element. We do not present this rule selection component more in detail,
because we believe that it is outside the scope of this chapter.

Once interpreted, the new element is created; it replaces the parameters of the applied rule in the document. New structural contexts are created to help interpreting the following elements; this is modelled by the DCC block. Once a rule is applied, the current iteration of the analysis process is finished. Then, a new iteration begins, in order to check if this new element interacts with other elements of the document to constitute a more complex symbol: we try to eventually apply a rule on the new element, and so on until stability; as a consequence, a stroke can imply a sequence of transformations. If no rule can be applied on a stroke (i.e., if the first iteration does not succeed), it is rejected and disappears from the editing window.

In order to explicit this interpretation process, Figure 9 presents its mechanism on one particular example, which is the interpretation of a sharp. In this example, the user first draws a vertical segment on the left of a filled-note head (1). This hand-drawn stroke is recognized by the system, and replaced by its neatly retranscribed symbol. This segment does not, for now, interact with other existing elements of the document to form a more complex one: the analysis process is over. The mechanism is the same when the user adds two horizontal segments on the left of this note head (2 and 3). Finally, the user draws a stroke which is supposed to be the second vertical segment that ends the sharp symbol. In the first iteration of the interpretation process, the stroke is interpreted as a vertical segment (4). This new element is then analyzed in the second iteration: it interacts with the three other segments to constitute a sharp (5). As this new symbol does not interact with other existing elements of the document to constitute a more complex symbol, the analysis process is over. We would like to remind the fact that the order in which these four segments are drawn is not constrained.

**User Validation**

As the recognition process is eager, the result of the analysis is displayed directly as the user is drawing. We can then exploit the human-computer interaction and integrate the user in the
interpretation process to validate or reject the results. Thus, if after the display of the answer, the user goes on with the drawing of the document, he implicitly accepts and validates it; on the contrary, if he does not agree, he can delete the new element with a deletion gesture and so explicitly reject it. The main consequence of this process is that it is not necessary for the analyzer to question a decision made beforehand because it has been validated by the user. We believe that it is pertinent because it could be perturbing for the user to see the interpretation of an element changing after drawing another one. It is a major advantage of the eager interpretation process over the lazy one: indeed, the user limits the ambiguities, which makes the system more robust and more efficient, increasing its user-friendliness and its usability. The system is also faster because it only has to find the sequence of rules to apply on the last hand-drawn stroke.

A Set of Graphical Functions and Pen-Based Editing Functions

The last component of the system is a set of graphical and pen-based editing functions: it deals with the human-computer interaction. In this section, we present these functions that are independent of the domain of the documents that are drawn, more in detail. Their principle is to enable the user to draw graphical gestures, which are not interpreted by the system as drawings of symbols, but as editing actions. Although we have developed our own graphical and pen-based editing functions, this component is not the main contribution of our method, because other authors have proposed similar functions (Lank, 2003).

Graphical Functions

Graphical functions correspond to the way the system displays its messages to the user. Their main functionality is the display of the results of the analysis process, i.e., the neatly retranscribed symbols, directly as the user is drawing his document. They exploit graphical information, such as images of these symbols, which are externalized of the system because they are dependent of the domain of the documents.

Graphical functions also propose solutions to guide the user in the drawing process. Indeed, to help him to have reference marks, rectangles giving an indication of the document structural contexts that are generated in DCC blocks can be displayed. We would like to note that whereas these displayed rectangles are strict, structural contexts are not: the user does not have to draw the elements exactly in the rectangles. To lighten the editing area, a context is visible provided that it is not already filled with an element. Thus, as presented on Figure 10, it is possible to switch between a novice mode, in which empty contexts are visible, and an expert mode, in which they are not. It is also possible to show only the contexts that are near the pen position, once again to lighten the editing area: it corresponds to a contextual mode. The experience shows that this last mode seems to be the most user-friendly because it allows to only focus on potentially interesting structural contexts according to the pen position. The authors would like to note that the filled-note head structural contexts presented on Figure 10 correspond to those declared in the FilledNote interpretation rule defined in Table 2.

Graphical functions also make it possible to display the neatly retranscribed symbols and/or handwritten strokes, as presented for instance in Figure 12 and Figure 13. Moreover, a document can be constituted of as many pages as possible; it is then, for instance, possible to switch form one page to another. Some classical functions,
such as zooming in or out, displaying an outline of some of the pages of the composed document or printing, are also already available.

Editing Functions

Editing functions correspond to the way the user expresses his requests to the system thanks to pen interaction. Concerned functions can be divided into two main categories, on the one hand the document functions that involve the management of the document, and on the other hand the element functions that involve the modification of the document elements.

Document functions are, for instance, adding or removing one page to the composed document, undoing the last action, saving or loading a document, and so forth. All these actions can be done using the pen, which offers an alternative to classical menu and button-based interaction.

Element functions are, for instance, selecting graphical symbols, moving them to another part of the document, deleting them, copying, cutting or pasting them, etc. For that purpose, every element of the document has a selection dot, which is a small red anchor point (which it is possible to make disappear, as presented in some figures of this chapter). In order to select an element, the user just has to circle its selection dot. This way, it is not necessary to draw a stroke as big as the element. Several elements can be selected or other elements can be added to the selection in the same manner. When an element is selected, the document elements associated to it are also selected; they actually correspond to the elements which have been drawn in the different contexts it has created. Once selected, elements can be moved to another part of the document by pointing to one of them and moving the pen to the appropriate place. It is also possible to move an element directly by pointing at its selection dot and moving the pen (by drag and drop). Moreover, selected elements can be copied or cut, and then pasted to another part of the document. To delete an element, the user can, for instance, move it outside the editing window. These mechanisms are illustrated by Figure 11: on the left, the user draws a stroke around the selection dot of a filled-note head; in the middle, the head and its associated elements (i.e., its stem and its flat) are selected, and the user moves them with the pen; on the right, he raises the pen to drop the elements. We would like to notice that the action of moving elements in the document requires the exploitation of the knowledge in the formalism, in order to check if the arrival position of the element is consistent with the rules associated to the document. Thus, the analysis process presented previously is also exploited for the moving of elements of the document.

Figure 11. Selection and moving mechanisms: Example on a quarter-note

Pen-Based Systems for Music Composition and Editing with the Presented Approach

Thanks to the methodology we present in this chapter, we have already developed various pen-based prototypes for the composition and the editing of structured documents from different domains:

- Pen-based editors adapted to classical musical score notations, plainchant score notations, drum tablature notations and finally stringed-instrument tablature notations. We are going to present these systems more in detail in this section.
- A pen-based graph editor, that allows to draw geometrical shapes and to connect them thanks to arcs; it also enables the user
to write text in the geometrical shapes; the top of Figure 12 presents two screenshots of this editing prototype, on the left the user hand-drawn strokes, and on the right the corresponding interpreted document.

- A pen-based UML class diagram editor, that makes it possible to draw classes, represented as rectangles, and to connect them to model associations; inheritance and aggregation are among already available symbols; the bottom of Figure 12 presents two screenshots of this editing prototype.

We would like to note that although it is not visible in Figure 12, the interpretation process of the systems that are developed thanks to the presented methodology is eager.

As indicated previously, the development of a composition and editing system based on pen interaction thanks to the presented methodology only requires:

- The writing of the interpretation rules thanks to the presented formalism.
- The design of the necessary hand-drawn shape recognizers (Plamondon & Srihari, 2000; Anquetil & Bouchereau, 2002).
- The specification of the graphical information, such as the images of the retranscribed elements.

In the next subsections, we focus on how this methodology can be exploited to develop pen-based composition and editing systems for various musical notations, whatever the desired associated composition conventions.

**Classical Musical Score Notations**

We first developed a pen-based musical score editor adapted to classical notations. In order to develop a system as usable and as intuitive as possible, professional musicians have guided us, in particular in the process of interpretation rule

---

*Figure 12. Screenshots of two pen-based composition and editing systems designed thanks to the presented methodology*
writing. Figure 13 presents two screenshots of the same musical score document: on the left only the user hand-drawn strokes are visible, and on the right the elements of the document are in their neatly retranscribed aspect (in order to lighten the screenshots, selection dots and document structural contexts are not shown).

Among the available symbols, the user can draw clefs (F-Clefs and G-Clefs), notes (from whole-notes to sixty-fourth notes) with uprising or descending stems, accidentals (flats, naturals, sharps, double-flats, double sharps) and durational dots. Beams are also available. Silences (from rests to sixty-fourth rests), bar lines (simple and double bar lines) can be drawn on the staves, and dynamics (crescendo, decrescendo, pppp, ppp, pp, p, mp, mf, f, ff, fff, ffff) can be drawn under them.

Table 3 presents some of these available symbols and some of their corresponding interpretation rules used in the developed system: it highlights in particular how the rules model that a flag or a beam can only be added to a quarter-note, and not to a half-note. The reader can also notice how the beam rule models that such a symbol must have its left and right points on the extremity (top or bottom) of two different quarter-notes of a same staff.

About 80 rules have been written in order to develop this system. The duration of the analysis process, that is, the duration between the moment when the user raises the pen and the moment when the corresponding neatly retranscribed symbol is displayed, is mostly less than 400 milliseconds, and always less than one second. We, as well as musicians, believe that it is short enough to be considered as quasi-instantaneous and to not disturb the user.

We would like to notice that our software does not constrain the user to write semantically correct notations. For instance, the user can draw any number of notes in any measure. Naturally, such functionality could be added.

Table 3. Examples of rules used in the pen-based musical score editor and the corresponding strokes and interpreted symbols

<table>
<thead>
<tr>
<th>Interpretation rules</th>
<th>Structural contexts before recognition</th>
<th>Structural contexts after recognition</th>
<th>Re-transcribed symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>UpStem (Stroke s)</td>
<td>DCV: HalfNote head [rightUp, all]</td>
<td>SCF:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GeometricalShapes, (VerticalSeg)</td>
<td>DCC:</td>
<td></td>
</tr>
<tr>
<td>UpStem (Stroke s)</td>
<td>DCV: FilledNoteHead head [rightUp, all]</td>
<td>SCF:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GeometricalShapes, (VerticalSeg)</td>
<td>DCC: this [top, extremity] Beam</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>this [top, extremity] Flag</td>
<td></td>
</tr>
<tr>
<td>DescStem (Stroke s)</td>
<td>DCV: FilledNoteHead head [leftDesc, all]</td>
<td>SCF:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GeometricalShapes, (VerticalSeg)</td>
<td>DCC: this [bottom, extremity] Beam</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>this [bottom, extremity] Flag</td>
<td></td>
</tr>
<tr>
<td>Beam (Stroke s)</td>
<td>DCV: UpStem extend [top, left],</td>
<td>SCF:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>UpStem extend [top, right],</td>
<td>SR: GeometricalShapes, (Segment)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>stemL = stemR, stemL.head.staff = stemR.head.staff</td>
<td>DCC:</td>
<td></td>
</tr>
<tr>
<td>Beam (Stroke s)</td>
<td>DCV: DescStem extend [bottom, left],</td>
<td>SCF:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DescStem extend [bottom, right],</td>
<td>SR: GeometricalShapes, (Segment)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>stemL = stemR, stemL.head.staff = stemR.head.staff</td>
<td>DCC:</td>
<td></td>
</tr>
</tbody>
</table>

Figure 13. Screenshots of the pen-based classical musical score composition and editing system: on the left the hand-drawn strokes and on the right the corresponding retranscribed document
Musical score notations used traditionally for classical and popular music is not the only one: other systems exist. Even though they are more specialized, they are often necessary for composers, interpreters, pedagogues, etc. We present three of these notations (for plainchant scores, drum tablatures and stringed-instrument tablatures), and highlight the genericity of the presented approach by showing how it is exploited to generate the corresponding composition and editing pen-based systems.

Plainchant Score Notations

The “Plainchant”, or “Gregorian singing”, is a type of occidental music, essentially based on a cappella singing. Generally, it is not written with the classical musical notations. Staves are constituted of only four lines and have one key. Notes are replaced by squares or diamonds isolated or gathered in neumes, which are short melodic figures executed in one breathing. Figure 14 presents an antiphonary of the middle of the XII century with such notations.

One of the main characteristics of the plainchant score notations is the diversity of the existing systems. Contrary to classical musical score notations, very standardized since the classical epoch, plainchant melodies have been written with very different notations. For instance, rhythms can be represented according to a modal system (i.e., based on the representation of short rhythmic cells), like for the “Ecole de Notre-Dame de Paris” in the XII century. On the opposite, the mensuralist notation, since last century, adopts a connection between the duration of a sound and the shape of the isolated note. This is the cause of lots of difficulties for usual musical composition software, which are therefore often adapted to very simplified plainchant notations. Only a formalism like the one we propose makes it possible to deal with such difficulties, by offering the possibility to easily modify composition conventions in order to adapt to different notations.

Among different variants for plainchant notations, some have started to be standardized since the XIVth century. In the continuation of this section, we focus on one example of these notations. It includes two kinds of simple notes: the punctum, which is a filled-square, and the virga, which is a filled-square with a stem. Table 4 presents three interpretation rules, which model the eager recognition of punctums, vertical segments (which can become, afterwards, stems) and virgas.

Then, punctums, and virgas can be joined to constitute various neumes. Most current neumes

Figure 14. Antiphonary of the Middle of the XII Century, with an Example of Plainchant Score

Table 4. Interpretation rules for basic plainchant notations

<table>
<thead>
<tr>
<th>Interpretation rules</th>
<th>Hand-drawn strokes</th>
<th>Re-transcribed symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Punctum (Stroke s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCV: Staff[in,all]s</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCF: GeometricalShapes, [Square]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SR:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCC:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Virga (Punctum punctum, VerticalSeg vSeg)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCV: Staff[in,all]punctum, staff[in,all]vSeg</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCF: vSeg[topLeft]punctum, vSeg</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SR:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DCC:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
are constituted of one to three notes. Table 5 introduces some of these symbols as well as the two existing clefs (the C-Clef and the F-Clef), and some interpretation rules which have been used to design the corresponding pen-based composition and editing system. The authors would like to highlight the fact that there are no constraints in the order of the drawings in the sub-elements of all the presented symbols. As a consequence, the resulting system is very user-friendly, because the user can draw the symbols in any order, as he does on a sheet of paper.

The developed system enables the user to draw as many staves as possible; it is possible to write the lyrics under these staves. Figure 15 presents two screenshots of this system, on the left the user strokes and on the right the corresponding interpreted document. As it can be seen on this figure, we have not yet coupled this prototype with a handwritten text recognizer: thanks to the context in which they are drawn, such strokes are identified as being text, but are not interpreted; as a consequence, we keep the strokes under their handwritten form.

About 20 rules model the eager interpretation process for this system. Its performances are even slightly better than the one for traditional musical score notations because the number of available symbols (and as a consequence of interpretation rules) is reduced.

**Table 5. Integration rules of more complex plainchant symbols; examples of keys and several neumes**

<table>
<thead>
<tr>
<th>Structure</th>
<th>Structural contexts before recognition</th>
<th>Structural contexts after recognition</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCV</td>
<td>Hand-drawn strokes</td>
<td>Re-transcribed symbols</td>
</tr>
<tr>
<td>SCV</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Drum Tablature Notations**

Notations for percussion instruments are less normalized than notations for other instruments. We focus here on drum tablatures. Contrary to traditional musical scores, which can represent music for any singer or instrument, tablatures are musical notations adapted to specific instruments. The graphical metaphor is no longer based on the pitch in the acoustic sense of the term, but on the physical constitution of the instrument: it is not about “what to play” any more, but about “how to play”.

A drum tablature is constituted of horizontal lines; each line is associated to one of the instruments of the drum set. On the left of each line, the name of the corresponding instrument is identified by a letter, for instance “C” for the cymbal, “HH” for the hi-hat, “SD” for the snare drum, “BD” for the bass drum, etc. Then, on each line, symbols depict how the corresponding instrument must be played: for...
the cymbal and the hi-hat, an “x” indicates that the player has to beat, like an “o” for the snare drum and the bass drum. A double stroke is denoted by “dd”. Bar lines can make it possible to delimitate the content of one tablature measure.

The developed system enables the user to draw as many tablatures as necessary, and each of these tablatures can be constituted of as many lines, that is, instruments, as necessary. All the symbols presented above are available and can be drawn in any order. Table 6 presents some of these symbols and the corresponding interpretation rules; in order to lighten the figures, the document structural contexts are not represented.

A screenshot of the developed system is given on Figure 16.

Less than 10 rules have been necessary to develop this prototype. Its performances are of the same order than the other ones. We would like to emphasize that, although this composition and editing system is so far quite simple and only adapted to drum, it can easily be completed with other symbols and adapted to other types of percussion instruments.

### Stringed-Instrument Tablature Notations

We now present another example of modern tablatures, adapted to fretted stringed-instruments, like for instance the guitar. As defined previously, a tablature does not represent “what to play”, but “how to play”; a stringed-instrument tablature tells the player where to place his fingers.

A tablature is constituted of horizontal lines representing the strings of the instrument. On the left of each line, the pitch of the corresponding opened string can be specified. Figures located on a line correspond to the fret of the instrument that the player must press down to produce the awaited sound, and “0” denotes an opened string. More complex notations can be introduced. Rhythm can be written under each figure thanks to symbols coming from the classical representation: a quarter-note is limited to its stem, two flags are indicated by a beam, etc. Other symbols make it possible to express particular playing modes, for instance “H” to represent a harmonic, “T” for a thumb, that is, a slap with the thumb, “P” for a pop, that is, a slap with another finger, and so forth.

We would like to note that, when limited to the most common symbols, stringed-instrument tablature notations are very close to drum tablature ones. As a consequence, it is easy to derive one from another, and to obtain a system with quite the same performances. Figure 17 presents a screenshot of a simple composition and editing.

---

**Table 6. Interpretation rules for drum tablature notations**

<table>
<thead>
<tr>
<th>Hand-drawn strokes</th>
<th>Re-transcribed symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>DrumTabLine (Stroke s)</td>
<td>DCV: Document[n,all].</td>
</tr>
<tr>
<td>SCV: .</td>
<td></td>
</tr>
<tr>
<td>SR: GeometricalShapes , HorizontalSeg .</td>
<td></td>
</tr>
<tr>
<td>DCC: this[all]Instrument* .</td>
<td></td>
</tr>
<tr>
<td>this[all]DrumStroke* .</td>
<td></td>
</tr>
<tr>
<td>this[below,all]LineBar* .</td>
<td></td>
</tr>
<tr>
<td>this[below,all]DrumTabLine* .</td>
<td></td>
</tr>
<tr>
<td>this[below,all]HorizontSeg* .</td>
<td></td>
</tr>
<tr>
<td>this[below,all]VerticalSeg* .</td>
<td></td>
</tr>
<tr>
<td>Instrument (Stroke s)</td>
<td>DCV: DrumTabLine[all,all].</td>
</tr>
<tr>
<td>SCV: .</td>
<td></td>
</tr>
<tr>
<td>DCC: .</td>
<td></td>
</tr>
<tr>
<td>DrumStroke (Stroke s)</td>
<td>DCV: DrumTabLine[all,all].</td>
</tr>
<tr>
<td>SCV: .</td>
<td></td>
</tr>
<tr>
<td>SR: Characters , x , o , d .</td>
<td></td>
</tr>
<tr>
<td>DCC: .</td>
<td></td>
</tr>
<tr>
<td>LineBar (Stroke s)</td>
<td>DCV: DrumTabLine[all,all].</td>
</tr>
<tr>
<td>SCV: .</td>
<td></td>
</tr>
<tr>
<td>SR: GeometricalShapes , VerticalSeg .</td>
<td></td>
</tr>
<tr>
<td>DCC: .</td>
<td></td>
</tr>
</tbody>
</table>

---

**Figure 16. Screenshot of the drum tablature composition and editing system**
system for stringed-instrument tablatures and whose interpretation rules are very close to the ones presented Table 6.

This basic prototype can be enriched with more symbols, for instance to model rhythm and particular playing modes. Table 7 presents some of the interpretation rules modelling the interpretation of such a document, and a screenshot of the corresponding system is on Figure 18.

**CONCLUSION AND FUTURE TRENDS**

In this chapter, we have faced the lack of software taking advantage of pen-interaction in order to compose and edit online structured documents with musical notations. We have presented a new generic approach to design such pen-based systems. It aims at interpreting the strokes eagerly, that is, directly as the user composes his documents. The main component of this approach is a formalism modelling how each symbol of the document can be interpreted: its originality is the coupling of a global vision of the analysed document in order to model in which structural context an element can be identified, with a local vision of this element in order to interpret its shape and that of its components. This formalism can be adapted to a large panel of structured documents. In order to emphasize its genericity, we have presented how it can be applied to interpret documents with various notations, such as traditional musical scores, plainchant scores, drum tablatures and finally stringed-instrument tablatures.

Future trends will aim at exploiting such pen-based systems at a larger scale, in order to offer an access to this technology to a larger public. People from various domains can potentially be interested in pen interaction. First, thanks to its aspect of play, pen-based software for pedagogue purposes will facilitate the initiation and learning of music and its notations. Pedagogy concerns the learning
of music theory, the intuitive and instantaneous composition (e.g., harmonisation exercises), ear formation exercises (e.g., musical dictations), and so forth. Examples of potentially interested targets are music schools, conservatoires, and universities. Secondly, pen-based music software is of great interest for composers and arrangers, whether they are amateurs or professionals. Such systems can be very useful in the process of composition itself (archiving of the drafts and of new ideas) as well as for the deposit of scores to author-composer society.

As far as the prototypes we have developed are concerned, many improvements can be achieved in order to make them more usable systems. Of course, we will have to add new symbols in the classical music notation system, for instance articulations (staccato, bow strokes, etc.), expressions, all the clefs, ties, and so forth. A composition on at least the two staves of the piano is often used: it will thus be conceivable. Thanks to the formalism associated to our method, such improvements will be straightforward. It will be essential to be able to hear the music and so for instance to propose an output in the MIDI format. The play of the music can be done, on the one hand, incrementally, that is, each time a new hand-drawn note is recognized, or, on the other hand, once the document is finished, that is, with the rhythm. A main step would be to couple pen-based systems with more traditional, keyboard and mouse-based musical editing software. Then, the user could exploit the advantages of both: on the one hand, he could draw musical notations from very different domains in a very intuitive and user-friendly way, as he usually does on paper, and, on the other hand, he could benefit of all the functionalities of classical software, such as the verification of the coherency of the notations, the formatting of the documents (e.g., the alignment of its elements), and so forth. Of course, pen-based systems could propose to save the documents in very widespread formats, such as MusicXML, in order to use them afterwards in any software able to deal with a file in such a format.

Finally, we will also have to define well-formalized test protocols in order to evaluate the usability and the user-friendliness of such pen-based prototypes.

ACKNOWLEDGMENT

The authors would like to thank Guy Lorette, Professor at the University of Rennes 1, from the IMADOC (Images and DOCuments) team of the IRISA (Institut de Recherche en Informatique et Systèmes Aléatoires) for his contribution to this work and his precious advices.

This project partially benefits from the financial support of the Brittany Region.

REFERENCES


Pen-Based Interaction for Intuitive Music Composition and Editing


