Design of a framework using InkML for pen-based interaction in a collaborative environment

Solen Quiniou, Mohamed Cheriet, Eric Anquetil

Design of a Framework Using InkML for Pen-Based Interaction in a Collaborative Environment

Solen Quiniou[1], Mohamed Cheriet[1], and Eric Anquetil[2]

[1]Synchromédia Laboratory – École de Technologie Supérieure, 1100 rue Notre-Dame Ouest, Montréal (Québec) H3C 1K3, Canada
[2]IRISA—INSA, Campus de Beaulieu, 35042 Rennes Cedex, France
Solen.Quiniou@synchromedia.ca, Mohamed.Cheriet@etsmtl.ca, Eric.Anquetil@irisa.fr

Abstract. We present a framework based on the standard InkML format to represent digital ink in a collaborative environment using pen-based interaction functionalities. This framework includes the capture, the rendering and the interpretation of the digital ink. In the proposed framework, we focus more particularly on the representation of the contextual environment of the ink and used it for its interpretation (as drawing, for example) as well as on the representation of semantic information attached to the ink after its interpretation.

Keywords: Digital ink, InkML, annotations, collaborative environments.

1 Introduction

With the emergence of pen-based devices such as PDAs, Tablet PCs or whiteboards, pen-based input methods are becoming relevant in collaborative environments. Indeed, they allow users to input elements like text or drawings in a more natural way to communicate with each other. Pen-based interactions are thus useful in domains like remote healthcare, collaborative document annotation or active learning in classrooms where applications like whiteboard sharing allow several users to write or draw on a virtual shared whiteboard (blank or containing documents like images that can be annotated, for example).

The digital ink data used for this interaction thus needs to be represented in some format to be exchanged among the users. This format should also be able to represent other contextual information that may be needed to interpret the digital ink as text, drawing or annotations, for example, and should be able to represent the result of this interpretation process. To do so, the *Ink Markup Language* (InkML) [1] has been proposed by the W3C Multimodal Interaction Activity as an open standard for representing both the ink data (entered with some electronic device) and the inking environment. Furthermore, as InkML is an XML-based language, it is easily extensible to add specific information needed by any developed application.

The context of this paper is the addition of pen-based interaction functionalities into the collaborative environment Synchromedia. Synchromedia is a consortium regrouping Canadian Universities (ÉTS, Concordia, Waterloo, UQÀM or Teluq), companies like Inocybe Technologies Inc. and other partners such as the Hôpital du Sacré-Coeur in Montreal. The aim of this consortium is to develop a collaborative environment integrating different media (texts, sounds, videos…). The current phase of this project focuses on methods to improve the communication between users and to include functionalities that take advantage of pen-based interactions and more particularly on the interpretation of digital ink to attach semantic information to it. For this purpose, we propose a framework based on the InkML format, where we focus more particularly on the representation of the contextual environment of the digital ink as well as on the representation of its interpreted elements. Previous works [2] and [4] also use InkML to represent digital ink in a collaborative environment but they focused more on dealing with interoperability problems to send digital ink and to render it on heterogeneous devices and platforms.

The rest of this paper is organized as follows. In Section 2, we present the basic elements of InkML and, in Section 3, we describe the framework and more particularly the use of InkML to represent the aforementioned elements. Finally, we draw some conclusions in Section 4.

2 Overview of the Ink Markup Language (InkML)

InkML[1] is a markup language for representing digital ink which has been proposed by the W3C as an alternative to proprietary ink data formats (e.g. Microsoft's Ink Serializable Format (ISF)). Its specification defines a set of primitive elements for all basic ink applications. Furthermore, since it is an XML-based language, it is easily extensible to add application-specific information to suit the needs of the applications. We present here the basic elements of InkML (see [1] for further details).

The fundamental elements in InkML are traces (defined between `<trace>` elements, which are contained within a single `<ink>` element). A *trace* represents a sequence of contiguous ink points between a pen-down and a pen-up. Each point is represented by values depending on the pen's position, such as its X and Y coordinates (the `<traceFormat>` element describes the quantities that may be considered for the points and may also include the tip force or the angle). The following code shows an example of a simple InkML file containing the X and Y coordinates of the points of two traces:

```
<ink>
  <trace>10 0, 9 14, 8 28, 7 42, 6 56, 6 70</trace>
  <trace>282 45, 281 59, 284 73</trace>
</ink>
```

Information about the digitizer device can be recorded in the `<inkSource>` element and may include the sample rate, the resolution or the model of the device. The `<context>` element enables the ink context to be taken into account and may include information about the writer or about the graphical context (using the `<brush>`

element to record the color and the width of the ink). InkML also allows logical regrouping of ink using either `<traceGroup>` or `<traceView>` elements referring to `<trace>` elements. These groups of traces can also be semantically labeled using `<annotationXML>` elements.

Furthermore, InkML allows digital ink to be represented into two styles. The *streaming* style is more appropriate for applications that deal with transmission of ink because it allows the ink to be considered as it is entered, relatively to a current state (context modifications are given according to this state). The *archival* style is more relevant for applications that store ink or process stored ink because the contextual information is separated from the traces (organized and annotated hierarchically).

3 Framework of an Ink Processor Based on InkML

In this section, we present the framework that uses the previously presented InkML elements to process digital ink.The *ink processor* consists of three modules to capture, render and interpret the digital ink (see Fig. 1). In a collaborative environment, each user has its own ink processor both to deal with its own digital ink and with the ink from the other users. These three modules are presented in the following sub-sections.

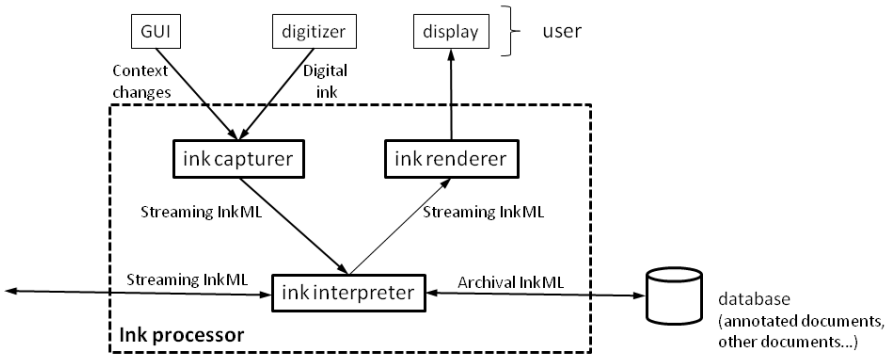

Fig. 1. Ink processor framework.

3.1 Ink Capturer

The aim of this module is to capture ink data from a digitizer and to represent these data in the Streaming InkML style by creating `<trace>` elements, where each point coordinates are generated according to the `<traceFormat>` element.

This module also receives context change events from the Graphical User Interface (GUI) and generates `<context>` elements. Among these context events are changes on the width or color of the ink (that may be specified by the `<brush>` element). Other information may be useful for the interpretation of the ink data and may be added to the InkML format, like a change of mode (a mode corresponds to the nature of the ink

which may correspond to gestures, annotations, texts, drawings or even to a search mode where the ink may be used as a request). Furthermore, in collaborative environments, useful information may include the ID of the ink owner or the protection level of the ink (eg public or private depending on if the data can be shared with other users). This InkML fragment shows examples of created context changes:

```
<context id=''context1''>
  <brush id=''black15pen''><width>15</width><color>#FFFFFF</color>
  </brush>
  <mode type=''drawing''/>
  <owner id=''writer1''/>  <protection>private</protection>
</context>
```

All these InkML generated elements are then sent to the ink interpreter module.

3.2 Ink Interpreter

The ink interpreter module receives ink data and context information both from the user and from the other users. It then sends its interpretation information to the renderer module as well as to the other users (which may depend on whether or not the ink is public).

The aim of this module is thus to interpret the received ink using the corresponding context information, received as well. The interpretation process mainly consists in associating semantic information to the ink data like, for example, the type of the ink (corresponding to the previously mentioned modes). To do so, the ink interpreter module may use specific interpreter sub-modules like handwriting, gesture or drawing recognizers. This overall interpretation process can be done in a *synchronous* way, thus generating context elements for changes of modes, before sending the corresponding ink, as shown in the given fragment code (a change to a gesture mode has been detected by the interpreter and is sent before sending the ink traces):

```
<context id=''context3''>
  <mode type=''gesture''/>
</context>
<trace>10 5, 10 7,...</trace>
```

The interpretation can also be done in an *asynchronous* way, as a background process, and the interpretation results will thus refer to previously entered and displayed ink, using `<traceView>` and `<annotationXML>` elements, as shown below (previous trace referenced by `''#L1''` has been interpreted as an annotation):

```
<traceView id=''L7'' traceDataRef=''#L1''>
  <annotationXML type=''annotation''/>
</traceView>
```

Moreover, in order to allow the ink and its interpretation to be further used, the corresponding information is stored in a database, after being converted to the Archival InkML style (using the conversion algorithm given in [3]). Thus, when the search mode is activated, the entered ink can be used as a request to retrieve corresponding documents, with ink documents being converted back to the Streaming style (the database may also contained documents without ink data).

3.3 Ink Renderer

The aim of this module is to render ink on the display area of the device, using the information given by the interpreter module, from the current user as well as the other ones. Ink rendering is an important aspect of pen-based applications because a non-satisfactory rendering, for a given user, may lead to a rejection of the corresponding application by this user.

To perform this rendering, `<trace>` elements sent by the interpreter module as well as `<context>` elements are processed; properties of the display device may also be taken into account to correctly render the ink. Furthermore, as the ink interpretation may be asynchronous, this module may also have to process `<traceView>` elements and thus change the rendering of already displayed ink data, according to the corresponding interpretation (ink data interpreted as annotations may be displayed with a different color, for example).

4 Conclusion and Future Works

In this paper, we have presented a framework based on the standard InkML format for representing digital ink in a collaborative context. We have focused more particularly on the representation of various context elements as well as on how to attach semantic information to the digital ink (the information being given by interpreter modules).

In future works, we will integrate the proposed framework in the Synchromédia collaborative environment to add pen-based functionalities into it. Given the domain applications, other extensions to InkML may also be needed. For example, we may want to further categorize annotations like in [5] as well as to specify different types of indexes to add search functionalities on documents (containing or not annotations).

References

1. Chee, Y.-M., Froumentin, M., Watt, S.M.: Ink Markup Language (InkML). http://www.w3.org/TR/InkML (October 2006)
2. Keshari, B., Madhvanath S., Prasad, M., Selvaraj, M., Watt, S.M.: Sharing Digital Ink in Heterogeneous Collaborative Environments. Proc. of the 11th International Conference on Frontiers in Handwriting Recognition. Montréal (2008)
3. Keshari, B, Watt, S.M.: Streaming-Archival InkML Conversion. Proc. of the 9th International Conference on Document Analysis and Recognition. Curitiba (2007)
4. Neddenriep, J., Griswold, W.G.: RiverInk - an Extensible Framework for Multimodal Interoperable Ink. Proc. of the 40th Annual Hawaii International Conference on System Sciences. Hawaii (2007)
5. Wang, X., Shilman, M., Raghupathy, S.: Parsing Ink Annotations on Heterogeneous Documents. Proc. of the Eurographics Workshop on Sketch-Based Interfaces and Modeling. Vienna (2006)