New Aspects of InkML for Pen-Based Computing

Stephen M. Watt
Department of Computer Science
University of Western Ontario
London, Ontario, Canada N6A 5B7
`watt@csd.uwo.ca`

Abstract

As pen-based computing becomes more prevalent, it is increasingly important to be able to share ink across applications and across platforms. The emerging standard Ink Markup Language (InkML) is intended for this purpose. Four drafts of the proposed standard have been produced since 2003, resulting in a specification with broad input from industry and invited experts. The last working draft has simplified and generalized the specification in some important ways and the specification is now in "last call" status. This paper presents an overview of the main features of InkML, with an emphasis on the rationale for what has changed for the last call version.

1 Introduction

Ink Markup Language (InkML) is an XML format under development for pen-based applications that store, manipulate or exchange digital ink. It provides a range of features to support real-time ink streaming, multi-party interactions and annotated ink archival. Applications may make use of as much or as little information as required, from minimalist applications using only simple traces to more complex problems, such as signature verification or calligraphic animation, that require full dynamic information. As a platform-neutral format, InkML supports collaborative or distributed applications in heterogeneous environments, such as courier signature verification or distance education.

The InkML specification is the product of four years of work by a cross-sector working group under the ægis of the World Wide Web Consortium (W3C), with input from Apple, Corel, HP, IBM and Motorola, as well as invited experts from academia and other sources. The "last call" working draft for InkML [1] was published on October 23, 2006, and has now undergone a period of public comment. At present various trial applications are under development. This paper presents an overview of InkML, with an emphasis on what is new and its rationale.

2 An Overview of InkML

InkML is an XML data format for the representation of digital ink. In its simplest form, ink is represented as a series of traces given by sequences of (x, y) coordinate pairs. More sophisticated use of InkML can include a variety of measurements for each point and can organize the ink into higher-level structures.

2.1 Traces and Trace Formats

The most basic concept of InkML is the *trace*, consisting of a sequence of comma separated points given inside a `<trace>` element. A simple InkML fragment might contain only traces, as Figure 1, an example from the InkML specification, shows. This example represents the five traces for the letters of the word "hello" Each point comprises an x and y coordinate in some device-dependent units. This is the default format for traces.

More generally, an application may specify coordinates and units using a `<traceFormat>` element. The source of the values a coordinate assumes over a trace is called a *channel*. Trace formats are specified by naming the ordered set of channels. The pre-defined channels include x, y and z positions, pen tip force, switch and button states, stylus orientation coordinates (x and y tilt, azimuth, elevation and rotation), color coordinates (RGB, CMYK), stroke width and time. Additionally, application-defined channels are supported.

Regular channels provide a coordinate value at each point and *intermittent* channels do not necessarily provide values at all points. Trace formats specify which channels are intermittent and they are given as optional additional point coordinates. Coordinate values may be boolean or numerical, with numbers given in base 10 or 16. Decimal numbers may have a fractional part. Numerical coordinates may be given as values or via first or second differences, allowing a more compact representation. To support streaming applications, where devices may have limited memory

```
<ink>
   <trace>
     10 0,9 14,8 28,7 42,6 56,6 70,8 84,8 98,8 112,9 126,10 140,13 154,14 168,17 182,18 188,
     23 174,30 160,38 147,49 135,58 124,72 121,77 135,80 149,82 163,84 177,87 191,93 205
   </trace>
   <trace>
     130 155,144 159,158 160,170 154,179 143,179 129,166 125,152 128,140 136,131 149,
     126 163,124 177,128 190,137 200,150 208,163 210,178 208,192 201,205 192,214 180
   </trace>
   <trace>
     227 50,226 64,225 78,227 92,228 106,228 120,229 134,230 148,234 162,235 176,
     238 190, 241 204
   </trace>
   <trace>
     282 45,281 59,284 73,285 87,287 101,288 115,290 129,291 143,294 157,294 171,294 185,
     296 199, 300 213
   </trace>
   <trace>
     366 130,359 143,354 157,349 171,352 185,359 197,371 204,385 205,398 202,408 191,413 177,413 163,
     405 150,392 143,378 141,365 150
   </trace>
</ink>
```

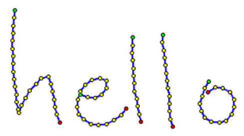

Figure 1. A simple example of InkML and a possible rendering

or where real-time performance is required, traces may be split across several `<trace>` elements, with continuations explicitly indicated.

These aspects are illustrated by the following example from the specification:

```
<traceFormat>
   <channel name="X" type="decimal"/>
   <channel name="Y" type="decimal"/>
   <intermittentChannels>
      <channel name="B1" type="boolean" default="F"/>
      <channel name="B2" type="boolean" default="F"/>
   </intermittentChannels>
</traceFormat>

<trace>
   1125 18432,'23'43,"7"-8,3-5,7 -3,6 2,6 8,3 6 T,
   2 4*T,3 6,3-6 F F
</trace>
```

After giving the coordinates of the starting point, subsequent points are given here by first and second differences denoted by prime (apostrophe `'`) and double prime (quote `"`) characters. The letters `T` and `F` represent true and false values for the boolean state channels. The asterisk ($*$) indicates an intermittent channel not reporting a value.

2.2 Contextual Information

A variety of contextual information may be provided for the interpretation of traces. This includes detailed properties of the ink source, such as channel thresholds, resolution, accuracy, quantization, channel cross-coupling, distortion and noise properties. Other ink source properties include latency, sampling frequency and the active area.

Other contextual information includes information about the virtual brush in use, reference time stamps and possibly mappings to and from standard or shared canvases.

2.3 Time

The time of points in traces may either be given explicitly, through the use of a time channel, or implicitly, as implied by an ink source sampling frequency and a reference time stamp for the start of the trace.

Applications that re-sample ink traces by interpolating some points and discarding others may wish to add a time channel if the new points are not equally spaced in time.

2.4 Grouping of Traces

Traces may be grouped together to share contextual information, to provide logical segmentation of the data, for annotation or to allow collections of ink strokes to be referred to as a unit. For this, the `<trace>` elements are placed within a `<traceGroup>` element.

Traces that already exist in other groups or documents may also be grouped virtually by referring to them via references using the `<traceView>` element. This allows the same ink data to be viewed in different ways, based on different criteria.

2.5 Streaming and Archival Ink

While InkML provides general mechanisms for each ink trace to refer to its own contextual information, there are two styles of referring to contextual information that deserve special mention. These are the *archival* and *streaming* styles of using InkML.

Archival InkML places all contextual information in one section of declarations and groups ink traces according to some logical structure, with traces or trace groups referring to the declarations as necessary.

Streaming InkML intersperses ink traces with declarative information that alters the current context. Traces are given in time order and are typically not grouped.

2.6 Use of XML *vs* Compactness

Although an XML representation of digital ink will be larger than a compressed binary format, using this representation allows a broader range of applications to handle ink data. Standard utilities, such as `gzip`, do a very effective job of compressing InkML data and can be used on complete files or data streams. Moreover, a compact binary representation of XML is of general interest and under design so, when available, will apply directly to InkML.

3 What's New in InkML

Perhaps the best way to summarize what is new in the current version of InkML is that it supports a broader range of possible higher-level applications without compromising its suitability for low-level data representation. The main new aspects of the current specification may be grouped into the following categories:

3.1 Structured Ink

There is greater support for applications to use InkML as a representation for their own application-defined structures. Now `<traceGroup>`s can be nested, allowing applications to organize ink into logical units if desired. This allows construction of hierarchical objects, such as letters, words sentences and paragraphs, or components of diagrams, equations, music, etc. This organization might be by construction, using the ink collection application, or as the result of analysis. This also allows more meaningful annotation of aggregate ink structures by the manipulating application.

InkML also now more robustly supports the construction of groupings of existing ink, either in the same document or from external sources. The `<traceView>` element can refer to ink via general URIs to form structured collections. This may be used, for example, to annotate existing ink documents with new layers or to provide multiple views of annotation on one ink document. Additionally, it is possible to put trace data in an out-of-stream `<definition>` element and then to refer to this ink via a `<traceView>`.

3.2 Abstract Ink Sources

The treatment of ink capturing devices has been abstracted to the concept of an *ink source*. This can still be a physical device, with a detailed description of hardware-level properties, or it can be a virtual device that arises as the result of data transformation or processing.

3.3 Optical *vs* Digitizing Tablets

Since the original drafts of InkML were written, the range of devices that may be used to capture digital ink has multiplied. No longer is it the case that almost all ink data is captured by a digitizing tablet. It is important to support camera-based devices of various sorts, mechanical contact devices and devices based on new sensing technologies. Consequently, InkML must now support a wider range of devices. This support takes two forms: First, the set of ink source properties has been abstracted and retains those qualities that make sense for a range of input devices, as well as for the outputs of data transformations. Secondly, a range of standard channels have been named to support the observations that these devices might report. In particular, optical devices might directly report stroke width or color information. Thirdly, the new permissible canvas transformations allows more general transformations from device coordinates to a standard canvas.

3.4 Enhanced Support for Annotation

There is now enhanced support for annotation, allowing any textual or XML-based information, providing sufficient hooks for rich semantic annotation of ink while keeping the standard simple. These annotations may be applied to structured ink collections and use attributes to name the (external) standard to which the annotations adhere. This allows InkML to be a small specification that is readily extended by applications that require annotations using other standards.

3.5 Support for Data Transformation

It is anticipated that InkML data will be used by many different sorts of applications and that original ink data may be transformed in various ways by a series of programs. To give these programs the freedom to reorganize and restructure the ink data as desired, InkML documents are no longer required to rely on the order of the ink traces within the document. Ink strokes can be re-organized within InkML groups for the convenience of the application. This has required adding certain attributes to InkML elements so that the relation between elements can be by explicit reference, rather than by relative position within the data file.

3.6 Streaming *vs* Archival

InkML now more robustly supports streaming, allowing a more general set of ink sources and application-dependent annotations. This has been achieved by simplifying the top-level content model of the `<ink>` element and making the treatment of contexts more uniform.

3.7 Shared Canvases and Mappings

The support for shared canvases has been enhanced, allowing multi-party applications to map to shared spaces recording all aspects of digital ink. The multiple different notions of mappings in the specification have been uniformized and converged. There is now only one mapping notion that is used everywhere. Now all of the previous different sorts of mapping specification (and new ones) can be used in all contexts.

3.8 Enhanced Support for Rendering

The new channels for color, width and brush rotation may be used by applications that wish to finely specify how digital ink should be rendered.

4 History and Related Work

The first draft of InkML [4] was produced following a working meeting a the 2002 International Workshop on Frontiers in Handwriting Recognition. This draft was based in a large part on contributions from IBM, with input from several other sources. The second and third drafts [3, 2] in 2004 and contained many refinements to the proposed standard. The fourth working draft underwent extensive internal review in 2005 and the remaining open issues were discussed at a working group meeting in October 2005. The last call working draft [1] was published in October 2006. Demonstration implementations of the standard are currently underway by working group participants and several should be available by the time of ICDAR 2007.

Earlier formats for digital ink notably include Jot [5], Unipen [6, 7] and Microsoft's `.isf` Ink Serialized Format [8]. Both Jot and Unipen were formats with open definitions that, in principle, could be supported on any platform. In practice, Jot was a binary format used in particular applications. Unipen was designed to suit the needs of those testing handwriting recognition algorithms on large amounts of data. In practice, it has been used to record databases of handwriting samples but is not well suited for a broad range of applications. Microsoft's Ink Serialized Format is a proprietary representation of digital ink as a compressed vector graphics format. Of interest is the International Telecommunications Union reference for telewriting [9].

5 Conclusions

Handwriting can be a powerful input for computing and communication, and InkML offers cross-platform medium to support it.

A number of simplifications and generalizations have been made to the InkML specification. The guiding philosophy has been to remove arbitrary restrictions so long as they neither complicate the specification nor compromise its ability to efficiently support low-level devices. Indeed, many of these generalizations have both increased the applicability of InkML as well as making it simpler to define and implement.

References

[1] *Ink Markup Language (InkML)*, W3C Working Draft 23 October 2006 `http://www.w3.org/TR/2006/WD-InkML-20061023`, Y-M. Chee, M. Froumentin, S.M. Watt (editors), Y-M. Chee, IBM K. Franke, M. Froumentin, S. Madhvanath, J-A. Magaña, G. Russell, G. Seni, C. Tremblay, S.M. Watt, L. Apple.

[2] *Ink Markup Language*, W3C Working Draft 28 September 2004 `http://www.w3.org/TR/2004/WD-InkML-20040928`, Y-M. Chee, M. Froumentin (editors), Y-M. Chee, J-A. Magaña, K. Franke, M. Froumentin, G. Russell, S. Madhvanath, G. Seni, C. Tremblay, L. Yaeger.

[3] *Ink Markup Language*, W3C Working Draft 23 February 2004 `http://www.w3.org/TR/2004/WD-InkML-20040223`, G. Russell, Y-M. Chee (editor), G. Seni L. Yaeger, C. Tremblay, K. Franke, S. Madhvanath, M. Froumentin.

[4] *Ink Markup Language*, W3C Working Draft 6 August 2003 `http://www.w3.org/TR/2003/WD-InkML-20030806`, G. Russell (editor), Y-M. Chee, G. Seni, L. Yaeger, C. Tremblay, K. Franke, S. Madhvanath, M. Froumentin.

[5] *Jot – A specification of an Ink Storage and Interchange Format*, Slate Corporation, 1993.

[6] *Unipen 1.0 Format Definition*, The Unipen Consortium, 1994. `http://www.unipen.org/dataformats.html`.

[7] *UNIPEN project of on-line data exchange and recognizer benchmarks*, I. Guyon, L. Schomaker, R Plamondon, M. Liberman and S. Janet, Proc 12th International Conference on Pattern Recognition, 1994, IAPR-IEEE, pp. 29-33.

[8] *Ink Serialized Format (ISF)*, Microsoft Corporation, 1994. Proprietary. Mentioned in `http://msdn.microsoft.com/msdnmag/issues/04/12/TabletPC/default.aspx`.

[9] *Telewriting terminal equipment* (ITU-T 150), International Telecommunication Union, 11/1998.