Mobile Viewers based on SVG$^{\pm geo}$ and XFormsGI

Thomas Brinkhoff[1], Jürgen Weitkämper[2]
Institut für Angewandte Photogrammetrie und Geoinformatik (IAPG)
Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven
Ofener Str. 16/19, D-26121 Oldenburg, Germany
[1]thomas.brinkhoff@fh-oldenburg.de [2]weitkaemper@fh-oldenburg.de

SUMMARY

Compared to raster data, vector formats show several advantages for the representation of maps. Especially in the domain of LBS, the reduced storage requirements and the "intelligence" of vector data with respect to user interaction make vector formats an ideal choice for a broad range of applications. The XML-based "Scalable Vector Graphics (SVG)" plays a growing role in vector graphic representations on the internet. As it is mainly developed with a focus on multimedia presentations, there are several aspects of SVG which are not required for the representation of geographic data (i.e. SVG is too powerful). On the other hand, typical features of GIS only can be modeled by large scripts or not at all in SVG. We propose a restriction/extension SVG$^{\pm geo}$ which addresses the special needs of SVG for GIS. Furthermore, LBS interact with the user to enter, edit or query spatial or non-spatial data. The XForms standard of the World Wide Web consortium addresses non-spatial user input embedded in documents (much like form elements in HTML-documents). We extend XForms to support interactive manipulation and input of geometric features by the user or by sensors. The concepts are demonstrated in a prototype running on PocketPC- based PDAs.

KEYWORDS: *SVG, SVG-Extension, XForms, Location Based Services, SVG-Viewer, SVG-Editing*

INTRODUCTION

Recently more and more GIS applications are adapted or transferred to mobile devices. Such applications are used as mobile information systems displaying geographic and related non-geographic information. For systems, which integrate the current position, the term *Location-based Services* (LBS) was coined. A second type of application is used to capture data which depends on location or to manipulate the geographic data itself.

Using a wide variety of devices with different operating systems demands for a data representation that is indifferent to the underlying hardware or software. In nearly all branches of computer science, XML-based standardized data representations are used in such scenarios. Also for GIS there exist several XML dialects to describe services or represent data, like e.g. the Geography Markup Language (GML) of the Open Geospatial Consortium (OGC, 2003).

The XML-based vector graphic standard SVG (W3C, 2003a) of the World Wide Web Consortium (W3C) is used to visualize vector graphics. In the World Wide Web, SVG becomes more and more widespread. Several authors remark that SVG allows high quality rendering of geographic data, see e.g. (Neumann & Winter, 2000).

Furthermore, we need means to realize user input as well for non-spatial as for spatial data and for presenting non-spatial data. For non-spatial data, XForms presents a solution. The W3C Standard XForms (W3C, 2003b) extends the forms mechanism of HTML and can be seamlessly integrated with SVG. XForms can be used to display or collect non-geographical data to or from the user.

In this paper, we propose an extension to XForms called XFormsGI that allows the realization of user interfaces for geometric interaction. The following features are discussed:

- Dynamic integration of location information and sensor data
- Interactive definition of geometrical entities
- Editing of existing geometrical data

The extensions are formulated with special focus on simplicity to allow easy implementation on mobile devices with restricted resources.

EXTENSIONS/RESTRICTIONS TO SVG

The SVG standard is very powerful and defines many "complicated" features that are required mainly for multimedia presentations. This makes SVG viewers large, unreliable and too resource demanding on mobile devices. Many of these features are not required for GIS.

On the other hand, there exist several requirements in the domain of GIS that can not be adequately solved using SVG. Mechanisms like generalization or rendering of content depending on view depth, scale etc. can only be realized – if at all – by relatively complicated scripts.

Location-based services introduce further needs. The position has to be represented and the view window and view direction have to be adjusted depending on the current movement and location of the user. The movement of the user in space makes it necessary to reload new map data over the internet and unload it when it is no longer needed.

The definition of a suitable extension/restriction (called $SVG^{\pm geo}$) of SVG is described in (Brinkhoff, 2003) and (Brinkhoff & Weitkämper, 2004). Details of the styling and generalization aspects are discussed in (Brinkhoff, 2005).

EXTENSIONS TO XFORMS

In this paper we propose an extension to XForms called XFormsGI that permits handling of geometrical data.

The user interface of XForms is defined by controls that can be embedded in SVG by the `foreignobject` element (as example see Fig. 1).

```
<foreignobject x="10" y="10" width="70" height="100">
      <xfm:input ref="/data/fsid">
            <xfm:label>Element-ID:</xfm:label>
      </xfm:input>
</foreignobject>
```

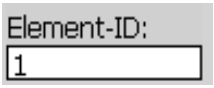

Figure 1: Embedding of XForms controls in SVG with a possible visual representation.

The XForms standard allows the extension by *proprietary user interfaces* (see Fig. 2).

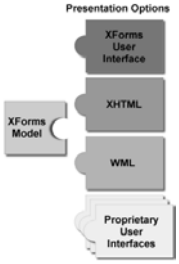

Figure 2: Interplay between the XForms model and different user interfaces (W3C, 2003c).

XFormsGI is an implementation of a user interface specialized to editing of geometrical entities in the host language SVG.

XFormsGI uses the standard method of XForms to process forms data. The so called *instance data* is collected in a subtree of the document (see Fig. 7). On activation of a submit trigger the data is serialized and submitted by one of several communication protocols. As an example the data could be sent over HTTP serialized as *application/xml* (as example see the *submission* element in Fig. 7).

XFormsGI provides the following controls:

- Definition of geometric entities:
  The *input* control (as an example see Fig. 3) is used to construct a geometrical entity like a line, circle, etc. interactively.
- Input from sensors:
  The *sensor* element represents a GPS sensor that is attached directly to the device (see. Fig 4). Extensions to other types of sensors are possible.
- Construction of new elements:
  In addition to the *input* control, a *construct* control uses the constructed geometry to add a corresponding SVG element to the document (see Fig. 5). The non-geometric attributes of the element are copied from a *template* element in the `defs` section of the document. The new SVG element is appended as child to the element given by the *parent* attribute.
- Selection of elements:
  The *selection* control is used to collect the ids of elements selected by the user.
- Editing of geometric properties of existing elements.

```
<xfmgi:input
      type              = "circle"
      id                = "polyinput"
      ref               = "/data "
      state             = "stopped"
/>
```

Figure 3: A fragment of XFormsGI defining a user operation “circle input”.

```
<xfmgi:sensor
      type              = "gps"
      source            = "default"
      ref               = "/data-gps"
      cursor            = "#gps-cursor"
      state             = "running"
/>
```

Figure 4: A fragment of XFormsGI defining a GPS sensor that automatically displays the current location by the SVG element referenced by `#gps-cursor`.

```
<defs>
      <image id="image-template" width="60" height="40"
             xlink:href="poi-symbol.svg" />
</defs>

<xfmgi:construct
      type              = "image"
      template          = "#image-template"
      parent            = "#new-elements"
      state             = "stopped"
/>
```

Figure 5: A fragment of XFormsGI defining the construction of a new image element.

INSTANCE DATA

The form data (also called *instance data*) is stored inside the document as a subtree of an XForms *instance* element (for an example see Fig. 7).

In GIS applications and LBS, we have to consider three coordinate systems:

- GPS-coordinates (referenced to WGS 84)
- The geodetic reference system of interest to the application (e.g. Gauss-Kruger)
- The SVG world coordinate system

The standard SVG 1.1 proposes a method to describe these systems that would be very expensive to implement on mobile devices. XFormsGI uses the very simple *coordinatereferencesystem* instead, see Fig 6.

```
<xformsgi:coordinatereferencesystem
       crs-id     = "EPSG:4326"
       transform  = "scale(1,-1)"
/>
```

Figure 6: XFormsGI definition of reference systems.

The EPSG code (EPSG, 2004) defines the geodetic reference system, and the SVG transform attribute describes the transformation from geodetic to SVG coordinates.

If a *coordinatereferencesystem* element is present in the document, all coordinate information is represented independently in SVG world coordinates as well as in geodetic coordinates. For a GPS sensor element, the overall structure of the instance data is shown in Fig. 7.

```
<xfm:model id="form1">
 <xfm:instance>
       <data-gps>
         <svgcoords>          73.4       125.6       </svgcoords>
         <globalcoords>       8.18968500 53.144997   </globalcoords>
         <gps-ext>
               Satellite infos
               Velocity, direction, ...
         </gps-ext>
       </data-gps>
 </xfm:instance>
 <xfm:submission method="post" action="http://www.xxx.de/xxx" />
</xfm:model>
```

Figure 7: Structure of instance data corresponding to a GPS sensor element

PROTOTYPE

There exist several freely available and commercial SVG viewers for mobile devices (for a list see W3C (2004), a Java-based viewer by one of the authors is described in (Brinkhoff, 2003)). They are either based on Java or implemented as an ActiveX control (only for PocketPC devices). In our tests, the Java viewers were much too slow to be used for geographic data. The evaluated commercial ActiveX viewers lacked the necessary extensibility to implement the XFormsGI interface. Thus, we decided to implement an own viewer in C++. The main target operating system is PocketPC/Windows Mobile, but operating system dependencies are kept to a minimum. Early versions have been ported to Symbian OS for smartphones and to Linux. The architecture of the viewer is depicted in Fig. 8.

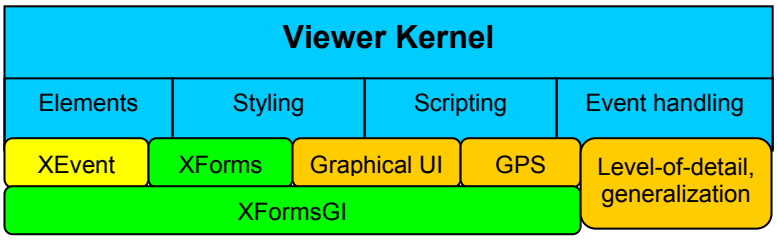

Figure 8: Architecture of the viewer.

Several of the described extensions are realized in the prototype. There exists an ActiveX Control which can be used with embedded Visual Basic. The source code will be made freely available (see Figs. 9 and 10 for some screenshots of the viewer).

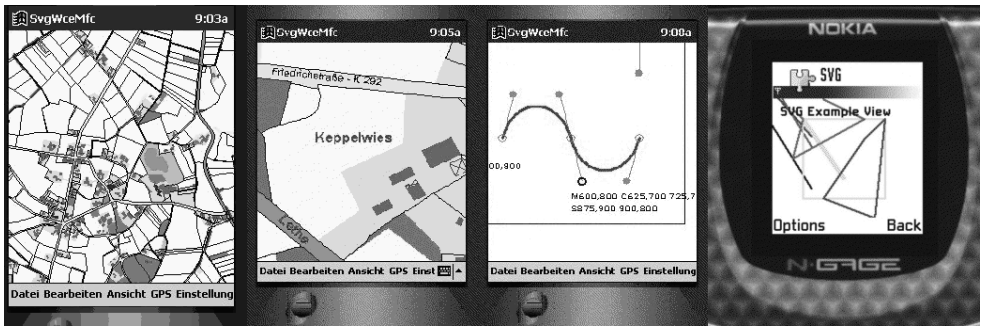

Figure 9: Some screenshots of the prototypes.

Figure 10: A complete graphical editor defined declaratively using SVG$^{\pm geo}$ and XFormsGI.

CONCLUSIONS AND FUTURE WORK

In this paper, we propose a first step towards the definition of an extension of XForms to integrate geometric user interaction into SVG. A set of controls is defined that allows the definition of

geometric entities, integration of sensor data and manipulation of geometrical properties of existing SVG elements. The feasibility of the ideas is demonstrated by a prototype running on PocketPC devices.

A lot of open problems remain for further investigation:

- Geometric restrictions and validation: XFormsGI needs to be extended to allow the definition of geometric restrictions for geometric input or manipulation. As examples consider the specification of a position on a road or inside a given piece of land.
- Completion of the event model of XFormsGI
- Referencing of existing geometric objects: During geometric constructions existing points or edges are often used to locate a new point precisely. Extensions to SVG are needed to describe the set of existing elements that are active for "snapping" or selection.
- Missing object model in SVG: Several problems arise due to the absence of an object model in SVG. Consider e.g. a land parcel that is represented in SVG by a group element containing a path and a text element (for the parcel number). In SVG each of the three elements may have an id attribute whereas in the application domain the "object" has the parcel number as single identifier. A natural solution would be to set the parcel number as id of the group element. Now consider a selection operation by the user. An interactive selection is bound to visible SVG elements, i.e. in our example text and/or path element(s) are selected. The question remains how to relate the selected elements to the object id which is needed for further application processing.
- Generation of SVG$^{\pm geo}$: The server side architecture required to generate SVG$^{\pm geo}$ from given data sources has to be systematically investigated.

BIBLIOGRAPHY

Brinkhoff Th. (2003): A Portable SVG Viewer on Mobile Devices for Supporting Geographic Applications. Proceedings 6th AGILE Conference on Geographic Information Science, Lyon, France, 2003, Presses Polytechniques et Universitaires Romandes, pp. 87-96.

Brinkhoff Th., J. Weitkämper (2004): Visualisierung und interaktive Bearbeitung von Geodaten mit SVG$^{\pm geo}$, Proceedings 3. Münsteraner GI-Tage 2004, 133-145.

Brinkhoff Th. (2005) Towards a Declarative Portrayal and Interaction Model for GIS and LBS. Submitted to AGILE 2005.

EPSG (2004) European Petroleum Survey Group (EPSG) Geodesy Parameters, 20.10.2004, http://www.epsg.org.

OGC (2003) Geography Markup Language (GML). Implementation Specification, Version 3.0.0, OpenGIS Implementation Specification, 29.1.2003, http://www.opengis.org/docs/02-023r4.pdf

Neumann, A., A. Winter. Kartographie im Internet auf Vektorbasis mit Hilfe von SVG nun möglich, carto:net, http://www.carto.net/papers/svg

W3C (2003a): Scalable Vector Graphics (SVG) 1.1 Specification, W3C Recommendation 14 January 2003, http://www.w3.org/TR/2003/REC-SVG11-20030114/.

W3C (2003b): XForms 1.0, W3C Recommendation, 14 October 2003, http://www.w3.org/TR/2003/REC-xforms-20031014/.

W3C (2003c): XForms - The Next Generation of Web Forms, 25 October 2003, http://www.w3.org/MarkUp/Forms/.

W3C (2004): Mobile SVG Viewers, 10 December 2004, http://www.w3.org/Graphics/SVG/SVG-Implementations.htm8#mviewer.