In: Tagungsband zu den 3. Münsteraner GI-Tagen, Juli 2004.

Visualisierung und interaktive Bearbeitung von Geodaten mit SVG$^{\pm geo}$

Thomas Brinkhoff [1], Jürgen Weitkämper [2]

Institut für Angewandte Photogrammetrie und Geoinformatik (IAPG)
Fachhochschule Oldenburg/Ostfriesland/Wilhelmshaven
Ofener Str. 16/19, D-26121 Oldenburg

[1] thomas.brinkhoff@fh-oldenburg.de [2] weitkaemper@fh-oldenburg.de

ZUSAMMENFASSUNG

Der vom W3C entwickelte Vektorgrafikstandard SVG bietet sich wegen seines rasch zunehmenden Verbreitungsgrades auch für die Visualisierung von Geodaten an. Für den Einsatz im Geodatenbereich gibt es eine Reihe von Anforderungen, die sich mit den vorhandenen Sprachmitteln von SVG nicht adäquat lösen lassen. Dies betrifft neben der Visualisierung vor allem auch die Erfassung von Sach- und Geometriedaten. Zur Lösung dieser Problematik wird in diesem Beitrag SVG$^{\pm geo}$ als Erweiterung und Ergänzung von SVG zur besseren Unterstützung von GIS-Anwendungen, speziell auch mit Blick auf mobile Endgeräte, vorgestellt.

EINLEITUNG

Im Bereich der Geoinformationssysteme lässt sich zur Zeit unter dem Schlagwort *Mobile GIS* ein wichtiger Trend beobachten: Immer mehr GIS-Anwendungen werden auf mobile Endgeräte wie PDAs verlagert. Für solche mobilen Anwendungen lassen sich zwei Kernaufgaben nennen:

1. Der Einsatz als mobiles Auskunftssystem: Hierbei steht die *Visualisierung* von räumlichen Daten mit ihren zugehörigen Sachinformationen im Vordergrund. Nutzen solche Anwendungen die aktuelle Position des Geräts, so spricht man auch von *ortsbezogenen Diensten* (*Location-Based Services*, *LBS*).

2. Der Einsatz als mobiles Erfassungssystem: Durch mobile Endgeräte lassen sich vor Ort raumbezogene Informationen aufnehmen und ohne Medienbruch (sofort oder später) weiterleiten und verarbeiten. Typische Aufgaben hier sind die Auswahl oder Bewertung vorhandener Daten oder die Aufnahme einfacher Geoobjekte (*Redlining*).

In den letzten zehn Jahren hat im Rahmen der Open-GIS-Initiativen des OGC und des ISO/TC 211 die Standardisierung von Schnittstellen, Daten und Dienstabläufen eine herausragende Bedeutung in der Geoinformatik gewonnen. Für einen reibungslosen Datenaustausch wird hierbei immer mehr auf Standards zurückgegriffen, die auf Basis der *Extensible Markup Language (XML)* definiert sind. Im Bereich der Repräsentation von Geodaten gewinnt insbesondere die *Geography Markup Language (GML)* (OGC 2003) stark an Bedeutung, die seit der Version 3 das gesamte OGC Feature Geometry Model (ISO 19107 Spatial Schema) abdeckt. GML dient in erster Linie der Beschreibung von Daten im Rahmen eines Datenaustausches oder für Geodatendienste wie dem Web Feature Service.

Der Visualisierung von grafischen Daten dient der vom World Wide Web Consortium (W3C) entwickelte XML-basierte Vektorgrafikstandard *Scalable Vector Graphics (SVG)* (W3C 2003a). SVG bietet sich wegen seines rasch zunehmenden Verbreitungs- und Unterstützungsgrades insbesondere auch für die Visualisierung von Geodaten an. So stellen Neumann und Winter (2000) fest, dass SVG zahlreiche Merkmale besitzt, die die Visualisierung von kartografisch hochwertigen Daten erlauben.

Für die Datenerfassung stellt *XForms* stellt eine Weiterentwicklung der von HTML bekannten Formulartechnik dar, die vom W3C als universeller Standard für XML-Dokumente definiert wurde (W3C 2003c). Eine Erfassung von geometrischen Objekten ist zur Zeit in XForms nicht vorgesehen.

Ziel der vorliegenden Arbeit ist die Spezifikation einer XML-Anwendung und deren Umsetzung zur Visualisierung und interaktiven Bearbeitung von Geodaten. Dabei soll insbesondere die Nutzung auf mobilen Endgeräten wie PDAs und Smartphones im Vordergrund stehen, aber nicht der ausschließliche Einsatzzweck sein. Als Basis dieser Entwicklung dient SVG. Für die effektive Nutzung von SVG auf mobilen Endgeräten ist einerseits ein Anforderungskatalog von Eigenschaften zu definieren, die für den Einsatz im Geodatenbereich zwingend erforderlich sind, und andererseits Erweiterungen zu beschreiben, die die Verwendung von SVG für Geodaten stark vereinfachen. Der daraus resultierende SVG-„Dialekt“ wird im Folgenden $SVG^{\pm geo}$ genannt. Daneben ist die Darstellung von Eigenschaften von Geodaten und insbesondere die Erfassung von Geodaten mit Hilfe von XForms ein wesentliches Ziel der vorliegenden Arbeit. Hierzu zählt auch die Einbindung von Sensordaten wie zum Beispiel GPS-Positionsangaben.

Nachfolgend werden die wesentlichen Merkmale von $SVG^{\pm geo}$ und der XForms-Erweiterungen beschrieben. Den Schlussteil des Beitrags bildet die Vorstellung einer prototyphaften Implementierung und ein Ausblick auf zukünftige Arbeiten.

SCALABLE VECTOR GRAPHICS UND SVG$^{\pm GEO}$

Der Vektorgrafikstandard *Scalable Vector Graphics (SVG)* (W3C 2003a) dient der Visualisierung von grafischen Daten. SVG ist XML-basiert und unterstützt neben Vektordaten auch die Einbettung von Rasterbildern, Audio- und Videodaten. SVG bietet alle erforderlichen zweidimensionalen Geometrieobjekte an und stellt Operationen wie Transformationen, Gruppierung, Clipping und Filterung zur Verfügung. Die Formatierung über CSS wird genauso unterstützt wie die Ereignisbehandlung über Skripte, die Zugriff auf das zugehörige Document Object Model (DOM) haben.

SVG bietet sich für die Visualisierung von Geodaten an. Die Anzahl der Werkzeuge, die einen Export von SVG-Dateien aus Geoinformationssystemen erlauben, nimmt zur Zeit stetig zu.

Allerdings wurde SVG für einen breiten Verwendungsbereich entworfen, insbesondere mit einem starken Fokus auf Unterstützung multimedialer Präsentationen. Für die Visualisierung von Geodaten bedeutet dies, dass zahlreiche Möglichkeiten angeboten werden, die in diesem Umfeld nicht benötigt werden. Dies ist insbesondere für die Implementierung von SVG-Viewern auf mobilen Endgeräten ein Hindernis. Für diese Zielrichtung hat das W3C (2003b) *SVG Tiny* und *SVG Basic* als funktional eingeschränkte *SVG-Profile* entwickelt. Hierbei stand die Visualisierung von Geodaten allerdings nicht im Vordergrund.

Ein weiteres Problem von SVG ist, dass es für den Einsatz im Geodatenbereich eine Reihe von Anforderungen gibt, die sich mit den vorhandenen Sprachmitteln von SVG nicht ad hoc lösen lassen, sondern im Regelfall durch Skript-Erweiterungen gelöst werden. Als Beispiele seien hier genannt:

- Aspekte der Generalisierung (z.B. maßstabsabhängige Sichtbarkeiten und Objektrepräsentationen),
- die dynamische Integration von Daten,
- die Integration von Sensordaten und des Ortsbezugs,
- die Anzeige von Sachattributen,
- die Erfassung von Sach- und Geodaten.

Ziel dieser Arbeit ist es, einerseits einen Anforderungskatalog (also ein SVG-Profil) von SVG-Eigenschaften zu definieren, die für den Einsatz im Geodatenbereich zwingend erforderlich sind, und andererseits Erweiterungen zu beschreiben, die die Verwendung von SVG für Geodaten stark vereinfachen. Der daraus resultierende SVG-„Dialekt“ wird SVG$^{\pm geo}$ genannt.

VISUALISIERUNG UND DYNAMISCHE INTEGRATION VON DATEN

Die Visualisierung von kartografischen Informationen mit einem SVG-Viewer ermöglicht und erfordert – insbesondere auf mobilen Kleingeräten – den Wechsel zwischen verschiedenen Zoom-Stufen. Für eine Benutzergerechte Darstellung der Geodaten bedeutet dies, dass die Darstellung einer kartografischen Generalisierung folgen muss. Um Methoden wie Vereinfachung, Typisierung, Zusammenfassung, Betonung und Verdrängung in einem SVG-Dokument umsetzen zu können, bedarf es der maßstabsabhängigen Darstellung von SVG-Elementen, so dass Kartenelemente bei einem Maßstabswechsel ggf. ein- oder weggeblendet werden, deren Darstellungsattribute geändert werden oder durch eine alternative Repräsentation ersetzt werden.

Abb. 1: Einblenden von Elementen in Abhängigkeit vom Maßstab.

Eng gekoppelt mit der maßstababhängigen Sichtbarkeit ist das Nachladen und das Entladen von Daten. In einer Übersichtsdarstellung sollte im Regelfall nur ein Bruchteil aller vorhandenen Kartenelemente eines SVG-Dokuments dargestellt werden. Damit ist ein nahe liegender Gedanke, SVG-Elemente erst später bei Bedarf (also wenn ein Nutzer in ein entsprechendes Gebiet hineinzoomt) zu laden (Held et al. 2003). Die Daten können also sowohl hinsichtlich des Maßstabbereichs als auch hinsichtlich räumlicher Aspekte gekachelt werden. Dies ist insbesondere für mobile Anwendungen, die Daten über eine langsame Datenverbindung erhalten, von großer Bedeutung. Auf mobilen Kleingeräten ist auch die inverse Operation – das Entladen von Daten – wichtig, da typischerweise dort der zur Verfügung stehende Speicherplatz begrenzt ist.

Deklarative Notation der Visualisierung

SVG bietet für grafische Elemente zwei Attribute, die die Sichtbarkeit von Elementen beeinflussen: `visibility` und `display`. `visibilitity` steuert die Sichtbarkeit von Elementen, die vom SVG-Viewer bereits interpretiert worden sind und über das DOM des SVG-Dokuments ansprechbar sind.

Mit `display` können hingegen Teilbäume aus dem Dokument herausgenommen werden, so dass sie von einem Viewer (zunächst) nicht interpretiert und repräsentiert werden müssen.

Zur Steuerung der maßstabsabhängigen Darstellung von SVG-Elementen bietet sich somit eine Erweiterung des Elements `visibility` an. Dazu wurde für SVG$^{\pm geo}$ ein neues Element `lod-visibility` eingeführt, mit dem Maßstabsbereiche definiert werden können. Ausprägungen dieses Elements werden – wie auch andere SVG-Deklarationen – im SVG-Element `defs` eingebettet. Ein `lod-visibility`-Element besitzt neben einer obligatorischen `id` Attribute, die die Grenzen des Maßstabsbereichs definieren.

Die Referenzierung des Sichtbarkeitsbereiches erfolgt objekt- bzw. gruppenbezogen über das SVG$^{\pm geo}$-Attribut `lod-visibility`, das allen Attributen, die das `visibility`-Attribut besitzen dürfen, hinzugefügt werden kann. Das Attribut verweist über ein URI auf das entsprechende `lod-visibility`-Element. Über die Definition in einer CSS-Datei, kann die Sichtbarkeit klassenweise gesteuert werden.

Deklarative Notation des Ladens und Entladens

Für das Laden und Entladen von Kartenausschnitten werden zwei Gebiete voneinander unterschieden:

- Der *Preload-Bereich* definiert ein Gebiet und einen Maßstabsbereich. Sobald der dargestellte Kartenausschnitt mit dem Preload-Bereich überlappt, wird das zugehörige (Gruppen-) Element geladen. Um Wartezeiten für den Nutzer zu vermeiden, sollte der Preload-Bereich räumlich und im Maßstabsbereich größer sein als das betroffene Element.
- Analog wird ein geladenes (Gruppen-) Element freigegeben, sobald der *Preserve-Bereich* verlassen wird. Der Preserve-Bereich sollte größer als der entsprechende Preload-Bereich sein, um ein wiederholtes Laden und Entladen des gleichen Kartenausschnitts zu vermeiden.

Beide Bereiche werden in dem SVG$^{\pm geo}$-Element `lod-display` definiert und über ein gleichnamiges Attribut referenziert. Die räumliche Definition kann entweder in absoluten Koordinaten oder als Distanzangabe bezüglich des Basiselements erfolgen. Das Attribut `lod-display` kann allen Elementen hinzugefügt werden, die das SVG-Attribut `display` besitzen dürfen, vorausgesetzt ein URI auf das zu ladende Dokument ist vorhanden.

SVG$^{\pm geo}$ unterstützt weitere, hier aus Platzgründen nicht erläuterte Konzepte, die die Visualisierung von Geodaten betreffen.

Nutzerinteraktionen zur Datenmanipulation in GIS-Anwendungen umfassen alphanumerische und geometrische Aspekte.

Erfassung von alphanumerischen Daten mit XForms

Bei der Eingabe alphanumerischer Daten finden typischerweise Formulare Anwendung. *XForms* stellt eine Weiterentwicklung der von HTML bekannten Formulartechnik dar, die vom W3C aus XHTML als eigenständiger Standard herausgelöst wurde, um universell einsetzbar zu sein (W3C 2003c). Mittels XForms lassen sich z.B. innerhalb von SVG – oder auch in jedem anderen XML-Dialekt – Formulare und die Weiterleitung der darin gesammelten Daten beschreiben. Charakteristisch für XForms ist die Trennung des Datenmodells von der Darstellung. Die konkrete Ausprägung von Eingabefeldern (z.B. als Listenfeld) wird nicht explizit festgelegt.

```
<foreignobject x="10" y="10" width="70" height=
  <xfm:input ref="/daten/name">
    <xfm:label>Name:</xfm:label>
  </xfm:input>
</foreignobject>

<foreignobject x="10" y="10" width="70" height=
  <xfm:select1 ref="/daten/status">
    <xfm:label>Status:</xfm:label>
    <xfm:item>
      <xfm:label>Student</xfm:label>
      <xfm:value>student</xfm:value>
    </xfm:item>
    <xfm:item>
      <xfm:label>Sekretär(in)</xfm:label>
      <xfm:value>secretary</xfm:value>
    </xfm:item>
    <xfm:item>
      <xfm:label>Dozent(in)</xfm:label>
      <xfm:value>prof</xfm:value>
    </xfm:item>
  </xfm:select1>
</foreignobject>
```

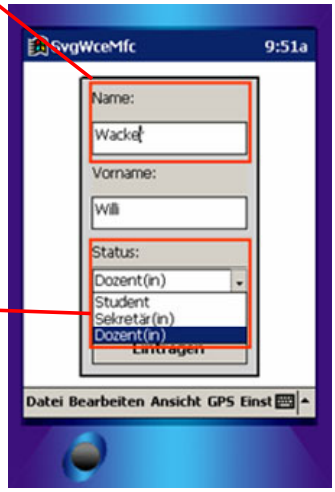

Abb. 2: Beispiel eines in SVG eingebetteten XForms-Formulars.

XForms Controls

Formulare sind aus Eingabefeldern, sogenannten *Kontrollelementen (Controls)*, aufgebaut. XForms spezifiziert verschiedene Basistypen, z. B. `input`, `textarea`, `select1` oder `choices`-Controls zur Eingabe ein- oder mehrzeiliger Texte bzw. zur Auswahl von Alternativen. Welche konkreten

Oberflächenelemente zum Einsatz kommen, bleibt dem Client überlassen, der je nach Geräteklasse unterschiedliche Repräsentationen wählen kann.

SVG erlaubt die Einbettung von Controls in `foreignobject`-Elemente. Die grafische Darstellung eines `foreignobject`-Elements wird aber nicht vom SVG-Client, sondern von einem *XForms-Prozessor* durchgeführt.

XForms Model

Das `model`-Element von XForms beschreibt über das `instance`-Element die Struktur der einzugebenen Daten. Zusätzlich wird im `submission`-Element die Weiterverarbeitung durch Angabe des Übertragungsprotokolls, des Datenformats und des Verarbeitungsagenten analog zum `action`-Attribut von HTML-Formularen definiert.

```
<model id="form1">
  <submission method="post" action="http://www.X.de/x.php"/>
  <schema/>
  <instance>
    <daten>
        <name>Wacker</name>
        <firstname>Willi</firstname>
        <status>prof</status>
    </daten>
  </instance>
</model>

<xfm:input model="form1" ref="/daten/name">
  <xfm:label>Name:</xfm:label>
</xfm:input>
```

Abb. 3: Model und Binding zum Formular aus Abbildung 2.

XForms Binding

Über das Binding wird der Zusammenhang zwischen den in den Controls eingetragenen Daten und den Daten im `instance`-Element hergestellt. Jedes Eingabefeld kann über ein `ref`-Attribut auf ein Element der Instanzdaten verweisen: Dazu wird ein XPath-Ausdruck (hier: `/daten/name`) eingesetzt.

Bei der erstmaligen Anzeige werden die Instanzdaten zur Vorbelegung der Controls verwendet. Während der Eingabe werden die Daten synchron im DOM-Baum abgelegt.

Die Erfassung von Geometrien über XForms: XForms-GI

SVG$^{\pm geo}$ sieht als Erweiterung von XForms die Einführung weiterer Kontrollelemente vor, um Standardinteraktionsmöglichkeiten zur Erfassung

von Geometrien deklarativ in eine SVG-Karte einbinden zu können *(„XForms-GI“)*. Die Weiterleitung der Daten erfolgt über die herkömmlichen XForms-Mechanismen.

Zur Zeit umfasst XForms-GI Kontrollelemente zur

- Selektion von Objekten
- Definition neuer geometrischer Elemente
- Modifikation vorhandener Elemente
- Verarbeitung von Sensordaten, insbesondere GPS.

Für die Elemente von XForms-GI wurde ein Namensraum mit dem URI `"http://www.fh-oldenburg.de/xformsgi/2004"` eingeführt.

Koordinatensysteme und geodätische Transformationen

Die von den Controls erzeugten Geometrien werden z. Zt. in Form von Listen von Koordinaten und Radien etc. in den zugehörigen Referenzelementen abgelegt.

Zur Weiterverarbeitung interaktiv erzeugter Geodaten ist es wünschenswert, die Koordinaten nicht nur in SVG-Koordinaten, sondern direkt im verwendeten Weltkoordinatensystem zu ermitteln. Damit wird ein erweitertes Binding erforderlich: bei einer Geometrie müssen zusätzlich zu den SVG-Koordinaten die Weltkoordinaten in den Instanzdaten abgelegt werden. XForms-GI führt dazu ein zweites Referenzattribut namens `ref-globalcoords` ein.

Der SVG-Standard 1.1 schlägt vor, das Referenzsystem und die Transformation in einem `metadata`-Element zu verwalten und gemäß der OGC-Spezifikation „Coordinate Transformation Services“ zu notieren. Für den Einsatz in mobilen Endgeräten wäre die Umsetzung dieses Vorschlags zu aufwändig. In XForms-GI erfolgt daher die Angabe des erforderlichen Bezugssystems stattdessen durch das neu definierte Element

```
<xfmgi:coordinatereferencesystem
  crs-id="EPSG:31467"
  transform="scale(1000,10000) translate(-5308.5,-812)"
/>
```

Das Attribut `crs-id` gibt den EPSG-Code (EPSG 2002) des Referenzsystems an, während das SVG-Standardattribut `transform` die Umrechnung der geodätischen in SVG-Koordinaten beschreibt.

XForms-GI Controls: Geometrische Erfassungsoperationen

Gegenüber herkömmlichen Formularelementen besitzen geometrische Eingaben eine größere Komplexität. Die Definition eines Kreises mittels Maus oder Stift besteht aus verschiedenen Schritten. Zunächst muss die grafische Eingabe gestartet werden, der Mittelpunkt muss gesetzt und dann der Radius bestimmt werden. Bei anderen Eingabeoperationen wiederum muss die grafische Eingabesequenz explizit vom Nutzer beendet werden (z. B. beim Polygon, dessen Punktanzahl zu Eingabebeginn nicht bekannt ist).

Darüber hinaus können während einer Operation weitere grafische Interaktionen erfolgen, wie Zoom- oder Pan-Operationen. Im SVG$^{\pm geo}$-Prototypen werden die jeweils unterbrochenen Aktionen durch einen Stack verwaltet.

XForms-GI realisiert das Starten und Stoppen geometrischer Eingabesequenzen durch ein `status`-Attribut mit dem Anfangswert `"stopped"`. Bei Beginn der Eingabe wird der Status durch einen Trigger auf den Wert `"running"` gesetzt. Analog wird die Eingabe durch Ändern des Werts auf `"stopped"` beendet. Um einen Verarbeitungsschritt direkt im Anschluss an die Beendigung einer Eingabe durchführen zu können, wurde das Ereignis `onterminate` eingeführt.

Stellvertretend für ein XForms-GI-Control ist nachfolgend das `circle`-Element (einschließlich Instanzdaten) dargestellt, das die Eingabe eines Kreises ermöglicht. Die Daten des konstruierten Kreises werden in der Form x, y, r in den zugehörigen `ref`-Elementen abgelegt.

```
<circle
  ref="flurstueck/data"
  ref-globalcoords="flurstueck/data_wc"
  state="stopped"
  onterminate="startPolygonInput()">
    <label>neuer Kreis:</label>
    <trigger>Eingabe starten</trigger>
</circle>

<model id="form1" >
  ...
  <instance id="instance1">
    <flurstueck>
        <data>25.5 14.7 13</data>
        <data_wc>70.8007 180.0537 0.0034</data_wc>
    </flurstueck>
  </instance>
</model>
```

Sensoreingaben

Sensoreingaben wie die GPS-Position können durch ein `sensor`-Element in ein SVG$^{\pm geo}$-Dokument eingebunden werden. Über ein `source`-Attribut

wird die Schnittstelle für den Sensor gewählt. Das `onchange`-Ereignis ermöglicht das Nachführen von SVG-Elementen entsprechend der Position.

```
<sensor type="gps"
  source = "nmea2.TXT"
  model  = "#form1"
  ref    = "/daten/flurstueck"
  ref-globalcoords = "/daten/flurstueck_global"
  state  = "running"
  onchange = "gpsChange()"
/>
```

PROTOTYP

Wegen nicht ausreichender Leistungsfähigkeit und der geringen Erweiterungsmöglichkeiten kommerzieller Viewer wurde ein eigener SVG-Viewer als Basis für die prototypische Umsetzung von $SVG^{\pm geo}$ entwickelt. Aufgrund der Erfahrungen aus früheren Projekten (Brinkhoff, 2003) wurde C++ anstelle von Java oder C# als Programmiersprache gewählt. Die deutlich höhere Leistungsfähigkeit von C++ gab den Ausschlag.

Der Prototyp wurde unter der Maßgabe entwickelt, möglichst weitgehend plattformunabhängig zu sein, ohne jedoch dafür die Performanz zu opfern (mit entsprechenden Konsequenzen für das Design). Lauffähige Implementierungen liegen auf folgenden Plattformen vor:

- PocketPC, als eigenständiges Programm oder ActiveX-Control
- Win32 (Desktop), als eigenständiges Programm oder ActiveX-Control
- Linux (mit der qt-Bibliothek von TrollTech, www.trolltech.com)
- Symbian OS 6.1

Aufgrund der aktuellen Marktsituation im Bereich der PDAs ist allerdings das Microsoft-Betriebssytem PocketPC die hauptsächliche Zielplattform.

Es werden folgende Open-Source-Bibliotheken genutzt: als Parser können *libxml2* (www.xmlsoft.org) und *expat* (www.jclark.com/xml/expat.html) gewählt werden, als Scripting Engine wird die Bibliothek *Spidermonkey* des Mozilla-Projekts (www.mozilla.org/js/spidermonkey) verwendet.

Es ist beabsichtigt, den Kern des SVG-Viewers als Open Source zur Verfügung zu stellen. Abbildung 4 zeigt die Architektur des Viewers und Abbildung 5 einige vom Viewer dargestellte SVG-Karten.

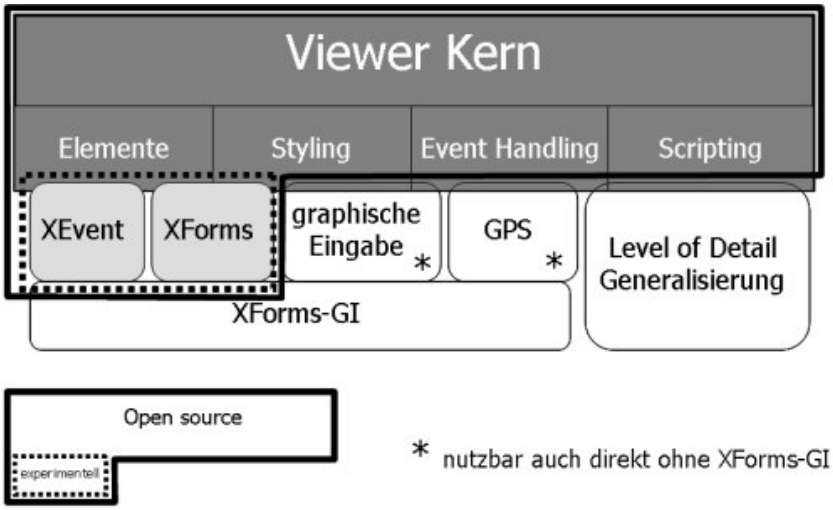

Abb. 4: Architektur des Viewers.

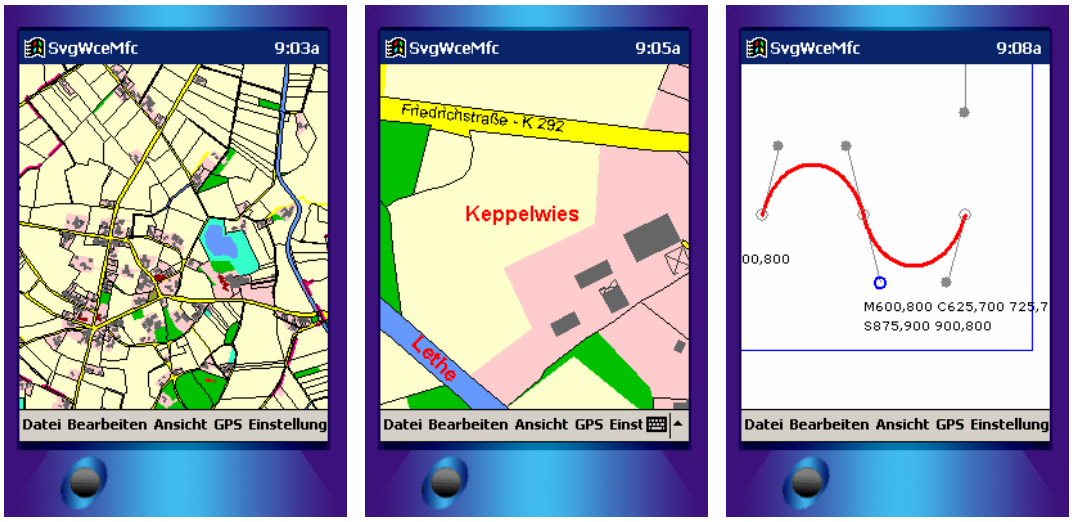

Abb. 5: Darstellung von SVG-Karten durch den Viewer.

SCHLUSSBEMERKUNGEN

In diesem Beitrag wurde SVG$^{\pm geo}$ als ein Vorschlag zur Erweiterung von SVG zur besseren Unterstützung von Geodaten und GIS-Anwendungen vorgestellt. Hauptaugenmerk wurde dabei auf die speziellen Visualisierungsanforderungen und die Erfassung von Daten gerichtet. Dabei fanden die Einschränkungen mobiler Endgeräte besondere Berücksichtigung.

Sowohl hinsichtlich der Konzeption von SVG$^{\pm geo}$ als auch der Implementierung stehen noch einige wichtige Aufgaben an:

- formale Spezifikation eines SVG-Profils für SVG$^{\pm geo}$

- formale Spezifikation von XForms-GI
- Fertigstellung der Implementierung des SVG$^{\pm geo}$-Viewers

Diese Aufgaben sollen im Rahmen des laufenden Projektes „SVG-Viewer für mobile Endgeräte“, das von der „Arbeitsgruppe Innovative Projekte (AGIP) beim Ministerium für Wissenschaft und Kultur des Landes Niedersachsen“ gefördert wird, bearbeitet werden. Darüber hinaus soll eine Integration in eine Architektur von Geodatendiensten untersucht werden.

LITERATUR

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.

EPSG (2002): http://www.epsg.org

Held G., Neumann A., Ueberschär N., Winter A. (2003): *SVG für die Webkartographie - Aktuelles und Zukünftiges*, carto:net, http://www.carto.net/papers/svg/

Neumann A., Winter A. (2000), *Kartographie im Internet auf Vektorbasis, mit Hilfe von SVG nun möglich*, carto:net, http://www.carto.net/papers/svg/

OGC (2003) OpenGIS *Geography Markup Language (GML)* Implementation Specification, Version 3.0.0, OpenGIS Implementation Specification, 29 January 2003, http://www.opengis.org/docs/02-023r4.pdf

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): *Mobile SVG Profiles: SVG Tiny and SVG Basic*, W3C Recommendation, 14 January 2003, http://www.w3.org/TR/2003/REC-SVGMobile-20030114/

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

W3C (2004): *Scalable Vector Graphics (SVG) 1.2*, W3C Working Draft 10 May 2004, http://www.w3.org/TR/2004/WD-SVG12-20040510/