Webmapping mit SVG:
Werkzeuge, Arbeitsabläufe, aktuelle Tendenzen

Tobias DAHINDEN, Andreas NEUMANN und Andréas M. WINTER

Institut für Kartographie, ETH Zürich
email:[dahinden,neumann]@karto.baug.ethz.ch
http://www.karto.ethz.ch/
http://www.carto.net/

Freytag & Berndt KG
email:winter@freytagberndt.at
http://www.freytagberndt.at/
http://www.carto.net/

Einleitung

SVG (Scalable Vector Graphics) ist seit September 2001 offizielle Empfehlung des W3C (World Wide Web Consortiums) und ist derzeit der »state−of−the−art« Vektorgraphik−standard für das WWW. SVG ist derzeit die am besten geeignetste Methode um hoch−qualitative und interaktive Kartographie im WWW zu betreiben. SVG wurde bereits letztes Jahr am Webmapping−Symposium Karlsruhe 2000 und an diversen anderen Konferenzen vorgestellt. Aus diesem Umstand heraus, und unter der Annahme dass ein Grossteil des Webmapping 2001 Publikums sich stärker mit dem Publikum des Vorjahres überschneidet, soll hier keine Einführung mehr für SVG erfolgen. Dieser Artikel soll die aktuellen Tendenzen und Möglichkeiten, sowie Arbeitsabläufe für die Präsentation kartographischer Daten im WWW aufzeigen. Für eine Einführung in SVG im Allgemeinen und SVG für Kartographie−Zwecke sei auf [14, WINTER/NEUMANN , 2000] und [5, FERRAIOLO, 2001] verwiesen.

Für und Wider: Kartographie im Internet

Da der SVG−Standard nicht für sich alleine funktioniert, sondern in ein ganzes Standard/Technologie−Gebäude des W3C eingebettet ist, soll SVG hier in einen grösseren Kontext gestellt werden: Dass das WWW eine interessante und kostengünstige Möglichkeit darstellt um Kartographie im Internet zu betreiben ist ja keine Neuigkeit. Folgende Gründe sprechen für einen Einsatz des WWW als einen Vertriebskanal kartographischer Darstellungen:

- WWW−Browser ist Standardapplikation in allen Betriebssystemen
- Prinzipiell sind die WWW Standards Browser−/Plattformunabhängig ...
- Webtechniken entwickeln sich schnell weiter, sind state−of−the−art
- Die meisten Techniken basieren auf offenen Standards, offenes Consortium (W3C)
- Viele Open−Source Lösungen vorhanden
- Kostengünstiges Medium, wenig bis keine Lizenzkosten
- Einfach, kostengünstig und schnell zu aktualisieren

- Voll multimedia−fähig
- Weltweite 24x7 Verfügbarkeit
- Cross−Media fähig: Saubere Datenaufbreitung erlaubt mehrere Produkte aus der gleichen Datenquelle: Web, CD−Rom, Print

Folgende Gründe können als Hemmschuh oder Hinderungsgrund betrachtet werden:

- Untersch. Browserimplementationen, Standards werden untersch. gut umgesetzt
- Braucht beträchtliches technisches Know−How
- Ständige Weiterbildung zwingend
- Betriebssysteme unterschiedlich gut unterstützt
- Standards z.T. unvollständig implementiert
- Offene vs. proprietäre Standards
- Unsicherheit beim Schutz der Grunddaten
- Kunden möchten f. Online−Services kaum bezahlen (ungekl. Vertriebskanäle)
- Datenpflege: Daten müssen à jours gehalten werden; Geschwindigkeit des Mediums WWW zwingt zu höherer Aktualität der Daten.
- Konkurrenzdruck von nicht−kartographischen Firmen (Informatiker, Graphiker, etc.)

(Weiter)entwicklung der WWW−Architektur

Das WWW hat sich in mehreren Phasen von einer reinen Wissenschaftsplattform zum Austausch von Artikeln und Forschungsergebnissen hin zu einem modernen Informati−ons− und Kommunikationsmedium gewandelt. Wie bei der Einführung neuer Techniken häufig der Fall, hat sich das Medium WWW jedoch phasenweise gewandelt und hat sich nicht, wie in der Presse häufig dargestellt, revolutionär und plötzlich etabliert. Folgende Phasen sind dabei auszumachen: (Die Zusammenstellung enthält Auszüge aus: [15, ZA−KON, 2001])

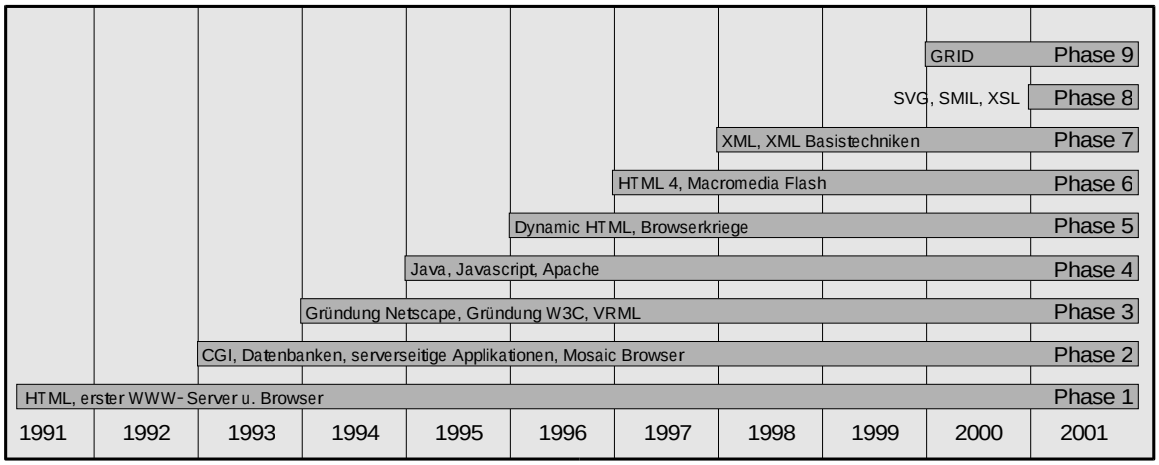

Abbildung 1: Entwicklungsphasen im WWW, Einführung v. neuen Techniken

Phase 1 (ab 1991) [2, vergl. BERNERS−LEE, 2001]:
Das WWW entstand 1990/91 am CERN (Genf), als **Tim Berners Lee** den **ersten Webserver (httpd) und graphischen Client (World Wide Web)** auf einem NEXT−

System schrieb. Das Web wurde darauf hin rege genutzt für den Austausch von wissen−schaftlichen Artikeln und Forschungsergebnissen.

Phase 2 (ab 1993) [4, vergl. CONOLLY, 1999]:
Das **CGI (Common Gateway Interface)** als Schnittstelle zwischen Webservern und Programmen wird definiert. CGI basierte Programme dienen v.a. Der Formulardaten−auswertung und Datenbankanbindung. Prinzipiell kann aber jede Programmiersprache und jedes scriptingfähige Programm via CGI eingebunden werden. 1993 wurde der po−puläre Browser **NCSA Mosaic unter X** geschrieben. Der CERN Direktor deklariert die WWW Technologie frei zugänglich für jedermann und bestätigt keine Gebühren dafür einheben zu wollen.

Phase 3 (ab 1994):
Netscape Communications wurde 1994 gegründet; die EU präsentiert ihre Pläne für den Information Superhighway. Der kommerzielle Browserhersteller Netscape beginnt ei−genwillig neue HTML−Tags einzuführen. Als Folge davon wird das **W3C gegründet**, um die Entwicklung neuer Standards kontrolliert und in einem öffentlich zugänglichen Prozess einzuführen. An der ersten WWW Konferenz am CERN wird das **VRML**−Format vorgestellt. Das Internet wird 1994 25 Jahre alt. Das weltweit erste Werbebanner erscheint auf hotwired.com.

Phase 4 (ab 1995):
Firmen und Privatpersonen entdecken das Internet um Informationen nachzuschla−gen und Produkte zu präsentieren. Obwohl die Unternehmen mit dem Anspruch auftra−ten, Ordnung in das »kreative Chaos« des WWW zu bringen, war die Kommerzialisie−rung des WWW meist kontraproduktiv: Nichteinhaltung von Standards, Einführung proprietärer Lösungen und eine einseitig gewinnorientierte Haltung war vielfach die Folge. Erst 1995 (als Netscapes Browser Lösungen bereits auf mehreren Plattformen etabliert waren) beginnt sich Microsoft für das WWW zu interessieren. Die Werbebran−che entdeckt das Internet und überschwemmt das Web mit blinkenden **Werbebannern** (GIF Animationen) → Filtersoftware wird geschrieben um diese zu unterdrücken. Sun lanciert **Java** um Webseiten mit interaktiven Inhalten versehen zu können. Netscape lanciert **Javascript** als clientseitige Scriptingsprache. Das **Apache−Projekt**, der weltweit populärste Webserver, wird geboren.

Phase 5: (ab 1996):
1996 war der Beginn der sog. **Browserkriege**: Netscape und Microsoft begannen sich mit proprietären Erweiterungen gegenseitig (V.a. aber den Web−Administratoren) das Leben schwer zu machen. Javascript−Implementierungen und der DOM−Zugriff (zwei Schlüsseltechnologien für **dynamisches HTML**) waren häufig inkompatibel zueinander. Webdesigner mussten deshalb zwei Versionen für die 2 meistverbreiteten Browser her−stellen. Das W3C stellt erstmals XML vor.

Phase 6: (ab 1997) [vergl. 11, MACROMEDIA, 2001]:
Macromedia führt die **Flash−Technologie** ein, um Webdesignern das Erstellen von Animationen zu erleichtert. Leider wird diese proprietäre Technologie zumeist für das Erstellen von Werbeeinschaltungen und multimedialen Zwangsbeglückungen (z.B.

Webpage−Intros) missbraucht, obwohl diese Technik für sinnvollere Dinge einsetzbar wäre ... Macromedia Flash hat das neue SVG−Format sicherlich massgeblich beeinflusst. Im gleichen Jahr gibt das W3C den **HTML 4.0** (den heute noch gültigen HTML−Stan−dard) frei. Damit ist auch dynamisches HTML erstmals clientseitig in einem sinnvollen Rahmen einsetzbar.

Phase 7: (ab1998):
XML als neue Metasprache zur Definierung weiterer Web−Formate wird offizielle W3C Recommendation. XML findet breite Akzeptanz in Forschung, Industrie und Open Source Kreisen. XML verlangt eine strikte Trennung zwischen den Daten, der Syntax−Beschreibung (DTD und Schema) und der Formattierung. Dadurch können aus dem gleichen Datensatz verschiedene Repräsentationen auf den verschiedensten Ausgabege−räten generiert werden. Mit XML können Organisationen, Firmen und Individuen eigene Datenformate kreieren, bei der Syntax, Datenbeschreibung aber auf zentral verfügbare XML−Techniken zurückgreifen. **XML Basistechniken** beinhalten DTD/Schema (Da−tenbeschreibung und Validierung), XSL (Formattierung), XLINK/XPointer (Verknüp−fungen), XPath (DOM Zugriff), XML Query (Abfragen), etc. 1998 wird auch das Nets−cape Web−Browser Projekt der **Open Source** Gemeinde übergeben und in **Mozilla** umbenannt.

Phase 8: (ab 2001):
Mit der W3C Recommendation von **SMIL** und **SVG** halten auch Vektorgraphik, Ani−mation und Multimedia offiziell Einzug unter den Webstandards. Gemeinsam mit DOM, Scripting und Datenbanken sind damit sehr einfach dynamische Seiten realisierbar, so−wohl am Client, als auch am Server.

Phase 9: (ab 1999):
Mit dem **GRID** (http://www.gridforum.org/) wird es möglich über das Internet nicht nur Daten und Informationen auszutauschen, zu präsentieren − es ist nun auch möglich, ver−teilte Rechner−Ressourcen und Dienste zu nutzen. So können brachliegende Rechenleis−tungen weltweit genutzt werden.

Interessant ist, dass während der verschiedenen Entwicklungsphasen verschiedene Ziel−gruppen als Webautoren angesprochen waren: waren es in der ersten Phase v.a. die Wissenschaftler, so stiegen mit dem Aufkommen der ersten Browserfirmen und der Einführung von Java die ersten Informatiker ein. Java ermöglichte auch die ersten In−ternet−Banking Anwendungen. Dynamic HTML und Flash sprach v.a. die Graphiker und Werbebranche an. XML und die verwandten Techniken schliesslich spricht ein sehr breites Publikum an: Wissenschaftler, Graphiker (Cross−Media Publishing, SVG SMIL), Multimedia−Entwickler (SMIL und SVG), Wirtschaftsinformatiker (Datenaustausch v. Geschäftsdaten), u.v.m. Das GRID wird zunächst erst einmal v.a. von Wissenschaftlern genutzt, und wird erst später den Weg in die Wirtschaft und bis hin zu den Privat−An−wendern finden.

Es ist auch wichtig, anzumerken, dass sämtliche Web−Techniken ein Konglomerat von wiederverwendbaren und ausbaubaren Modulen bildet. Kaum eine Technik ist für sich alleine zu gebrauchen, zusammen kann man sie erst für sinnvolle Anwendungen einset−zen. Eine Nebenwirkung der »Evolution in Phasen« ist das gleichzeitige Existieren

mehrerer de−facto Standards. Diese Umstände zwingen zu einer profunden Kenntnis des Mediums wenn man es zielgerecht und kartographischen Ansprüchen entsprechend einsetzen will. Eine klare Trennung in Graphikdesigner, Programmierer (GUI, Funktionalität und Datenbanken) und Redakteur ist in diesem Sinne nicht immer praktikabel, keiner der drei Bereiche kann ohne Kenntnis der Grundsätze der anderen bearbeitet werden.

Designprinzipien des WWW

Tim Berners Lee, der Direktor des W3C, hat folgende Design−Richtlinien für gegenwärtige und zukünftige Webstandards/−architekturen herausgegeben: [3, Berners−Lee, 1998]

- **Einfachheit**: KISS (Keep it Simple Stupid) ist das Motto der Unix−Betriebsysteme: Definiere eine kleinere Anzahl von flexibleren Tools und kombiniere sie um mächtigere Tools zu erhalten. »Einfachheit« darf nicht mit »einfach zu verstehen« verwechselt werden ...
- **Modularität**: Modulares Design, kombiniert mit klar definierten und dokumentierten Schnittstellen erlaubt in der Zukunft einzelne Komponenten auszutauschen, ohne das ganze »Gebäude« restrukturieren zu müssen. Entwickler können damit einzelne Module entwickeln, ohne andere Module im Detail verstehen zu müssen. Das Motto eines guten Modules sollte sein: »Do one thing well, and leave other things to other modules ...«
- **Toleranz**: *"Be liberal in what you require but conservative in what you do"* ... Für den Webdesigner hiesse dies etwa, sich für die eigenen Projekte sehr strikt an die Standards zu halten, für die Browserhersteller tolerant auch falsch geschriebene HTML−Seiten zu akzeptieren und einigermassen »richtig« darzustellen. Das »Toleranzprinzip« soll jedoch keine Entschuldigung für die unterwandern von Standards sein.
- **Dezentralisierung**: Das Konzept der »Dezentralisierung« ist dem Web per Definition eigen. »Single Points of Failure« und »Flaschenhälse« sollten jedoch dennoch vermieden werden. Die Dezentralisierung im WWW »mischt die Karten neu« und gibt auch Einzelpersonen und Randgruppen die Möglichkeit sich zu artikulieren und an gesellschaftspolitischen Prozessen teilzuhaben.
- **Test of Independent Invention**: Wenn zwei Entwickler(gemeinden) gleichzeitig, aber unabhängig voneinander, an Projekten mit gleichen Zielsetzungen arbeiten − werden die beiden Systeme/Implementationen so flexibel sein, um miteinander kommunizieren und zusammenarbeiten zu können? Wird man die beiden Systeme ineinander überführen können? Die Vergangenheit hat gezeigt, dass in vielen Fällen nicht klappte (z.B. inkompatible WP−Formate) ... das WWW mit seiner XML−Basis bringt hier wesentlich bessere Voraussetzungen bereits mit.
- **Principle of Least Power**: Oft hat sich herausgestellt, dass einfachere und wenig mächtigere Ansätze (z.B. Programmiersprachen und Datenformate) ein weiteres Nutzen der Daten wesentlich vereinfachen, im Vergleich zu »Black−Boxes« oder komplizierten Konstrukten. Als Veranschaulichung soll die Veröffentlichung von Daten im WWW dienen: Wenn jemand seine Daten in einer HTML Tabelle publiziert, kann jemand anderes die Daten einfach auswerten, Diagramme davon generieren, mathematische Berechnungen damit anstellen, etc. Komplexe Darstellungen, etwa durch Java−Applets limitieren den Zugang auf die vom Herausgeber/Entwickler vorgesehene Nutzung, obwohl sie vorderhand als kräftiger erscheinen mögen.

Merkmale von SVG

Nach dem »Ausflug« in allgemeine Webtechniken zurück zu SVG. SVG ist ein XML−Format mit all den Vorteilen, die XML bietet. Folgende Eigenschaften machen den Einsatz von SVG interessant:

- XML basiert
- Nutzung aller XML−Basistechnologien: XSL, XPATH, XMLQuery, RDF, etc.
- Les−, schreib− und interpretierbar von Mensch und Maschine
- Plattform−, Software− und Herstellerunabhängig
- DOM− und Scripting−transparent
- Gut m. serverseitigen Techniken kombinierbar: z.b. PHP, Perl, JSP/Servlets, ASP, etc.
- Gut aus Datenbanken und GIS heraus generierbar
- Erweiterbar
- Einbetten v. Metadaten und Attribut−Daten möglich

- Basisgeometrie: Kreise, Rechtecke, Ellipsen, Linien, Polylinien, Polygone
- Komplexe Pfade m. kubischen u. quadrat. Bezierkurven; Löcher und disjunkte Pfade
- Transformations: Translation, Scaling, Rotation, Skewing, Matrix
- Clipping, Masking and Compositing
- Füllungen: Raster, Vektor, Farbverläufe
- Transparenz
- Strich: Strichlierungen, transparente Striche, versch. Linienendentypen
- Symbole: können zentral definiert werden
- Filter: z.B. Lichteffekte, Schatten, Schärfe, Unschärfe, etc.
- Text: v. Suchmaschine indizierbar
- Textformattierung: volle Möglichkeiten wie bei DTP/Layout Programmen
- Fonts: Einbetten v. einzelnen Glyphen und Font−Sets
- Graphik−Stile: CSS und XSL transparent

- Interaktivität: Maus−, Keyboard− und Status− und DOMTree−Ereignisse
- Scripting: Support von ECMA Javascript
- Programmierung: Support v. IDL, ActiveX/Com
- Cursor: frei definierbar
- Linking: both external and internal

- Animation: almost every attribute (f.e. color, position, geometry, transformation, etc.)
- Animation eines Objektes entlang eines Pfades, mit automatischer Orientierung
- Zeitkontrolle: Zeitdauer, Anzahl der Wiederholungen, relative Zeitevents, etc.
- Interpolationen: diskret, linear, gestuft, spline−basiert
- Animation ist kompatibel mit SMIL

Für eine genauere Beschreibung des SVG−Formats siehe die SVG Spezifikation [5, Ferraiolo, 2001], [6, Fibinger, 2001] und [14,Winter/Neumann , 2000]. Für einen Ver−gleich mit dem Macromedia .SWF Format (Flash) und dem W3C WebCGM siehe [12, Neumann, 2001] und [13, Neumann/Winter, 2001]. Zusammenfassend kann gesagt wer−den, dass SVG ein klares Superset der Möglichkeiten von Macromedia Flash darstellt, SVG im Gegensatz zu .SWF auf offenen Standards basiert und wesentlich besser mit

anderen Technologien integrierbar ist.

Werkzeuge zum Erstellen und Bearbeiten von SVG

Im Folgenden weisen wir auf die uns bekannten Werkzeuge hin. Es kommen dabei Programme aus dem Macintosh, Windows und UNIX/Linux−Bereich zur Sprache.

ASCII−Editoren

SVG ist ein XML−Dialekt. Das bedeutet SVG ist textbasiert und kann somit von jedem beliebigen Text−Editor (z.B. Emacs, vi, Notepad, nedit) bearbeitet werden. Von Vorteil ist ein Editor, der die SVG−Tags farbig darstellen kann. Dazu eignen sich XML−Editoren (z.B. **XML−Spy**). Einigen Editoren mit Unterstützung für Syntax−Highlighting kann SVG auch beigebracht werden (z.B. **Xemacs, UltaEdit32**). Falls eine Eingabe repetitiv erfolgen muss, eignen sich dafür v.a. Editoren mit guten Such− und Ersetzfunktionen (z.B. **sed & awk**). Kleine Scripte (z.B. **Perl, PHP, Python**) können sich ebenfalls gut für solche Arbeiten eignen.

Autorensysteme

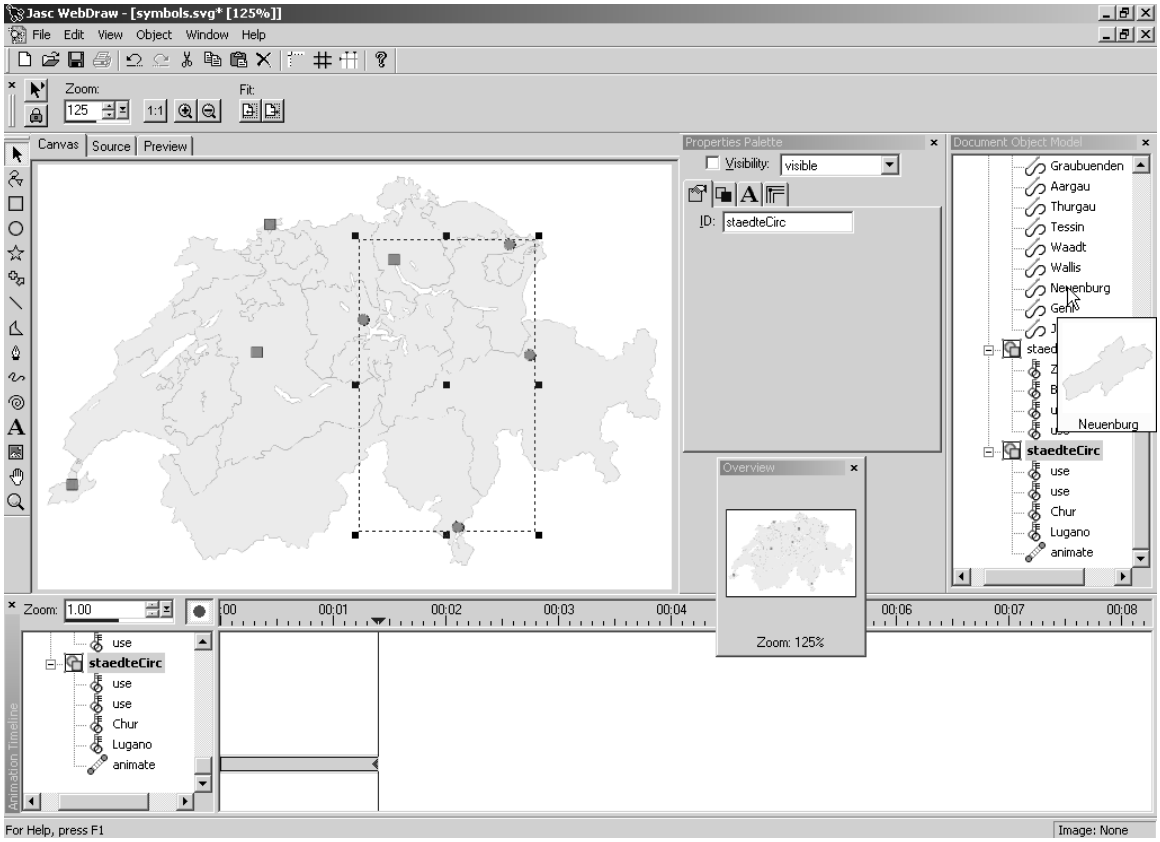

Abb 2: Jasc WebDRAW, ein interaktives SVG−Autorensystem

SVG−Autorensysteme sind häufig noch in einer frühen Entwicklungsstufe. Am weitesten fortgeschritten ist sicher "**Jasc WebDraw**". Bei diesem Produkt ist es möglich jederzeit von einem WYSIWYG−Modus in einen Quelltext−Modus umzuschalten und umgekehrt.

Zudem ist der Adobe−SVG−Viewer als Vorschaumodus eingebaut. Damit wird es einfach möglich SVG einfach zu editieren und sofort Fehler zu finden. Es gibt auch Unterstützung für Animation und Filter. Das Produkt befindet sich noch in der Betaphase. Es zeigt sich aber jetzt schon, dass damit sehr angenehm gearbeitet werden kann.

Vektorzeichenprogramme
Im Moment kennen wir folgende Zeichenprogramme, die SVG **Import−** und/oder **Exportfilter** anbieten: Adobe Illustrator, Corel Draw, Mayura Draw, Sphinx Open Editor, Sketch (OpenSource), KDE Kontour (ehem. Killustrator). Exportfilter sind einfach zu handhaben. Die meisten Programme können zudem noch andere Formate als ihr eigenes lesen. Damit ist es möglich eine Vielzahl von schon bestehenden Graphiken und Karten in SVG umzuwandeln.

Beim Export ist jedoch **Vorsicht** geboten. Flächen und Linien werden meist gut exportiert. Symbole jedoch benötigen unnötig viel Speicherplatz. Sie werden häufig als Pfade exportiert. Zudem wird oft auch nicht erkannt, dass es sich um Wiederholungen des selben Objektes handelt. Es ist daher ratsam, Symbole im Nachhinein im Textmodus in der Karte zu platzieren. Zu beachten gilt auch, dass mit den Exportfiltern noch keine Interaktion in die Karte eingebaut wurde. Auch da muss im Nachhinein im Textmodus gearbeitet werden. Macromedia (FreeHand) unterstützt das hauseigene Flash. Man kann jedoch über kleine Umwege aus einem FreeHand− oder Flash−File ein SVG−File herstellen. Ältere Programme (z.B. Xfig, Idraw) kennen SVG (noch) nicht. Sie stützen sich meist auf eigene (binäre) Formate. Es ist jedoch zu hoffen (und zu erwarten), dass SVG von der Graphikindustrie noch mehr Unterstützung erhält.

Konverter
Es existieren spezialisierte SVG−Konverter (komerziell und als OpenSource) um andere Graphikformate nach SVG zu konvertieren. Für offene, dokumentierte Formate können solche Filter viel einfacher hergestellt werden, als für undokumentierte und proprietäre Formate. Wir kennen Konvertierungstools von **PostScript, Encapsulated PostScript, pdf, wml** und **swf** (Macromedia Flash) nach SVG.

Für ArcView−Anwender gibt es den Konverter **shp2svg**. Er wandelt (wie der Name antönt) Shapefiles ins SVG−Format um. Der Konverter erzeugt ein SVG−File mit den Daten und ein HTML File, um das SVG−Dokument anzuzeigen. Das Programm ist unter (http://www.carto.net/) erhältlich. Unter der selben Adresse findet man auch Angaben zur Anwendung und zu den Limiten des Programmes.

Programmierhilfen
Das SVG−Format ist offen und gut dokumentiert. Es existieren XML−Parser und Bibliotheken mit denen Sie einfach einen eigenen Konverter schreiben können. Es gibt eigene **SVG−Libraries** für C/C++, Java und Perl−Programmierer. Diese helfen das Lesen und Schreiben in SVG−Files zu vereinfachen. Damit können Sie einfach Arbeitsabläufe automatisieren. Der Arbeitsfluss in SVG wird dann Ihren eigenen Bedürfnissen angepasst. (Bitte denken Sie auch daran, dass auch andere an ihren Programmen Interesse haben könnten.)

Drucker−Treiber
»Software Mechanics« entwickelte einen SVG Druckertreiber (SVGMaker), der ähnlich wie Adobe Distiller funktioniert. Er erlaubt Applikationen direkt in SVG−Files zu drucken. Dies ist eine gute Idee um SVG−Unterstützung für Applikationen hinzuzufügen, die SVG nicht direkt unterstützen. In der Praxis hat sich jedoch gezeigt, dass dadurch die **Dokumentenstruktur verloren** geht. Viele Optimierungen und erweiterte Funktionalitäten, die selber geschriebene Konverter durchführen können, kommen bei einem Druckertreiber nicht zur Anwendung. Dies dürfte v.a. eine Möglichkeit zum Erzeugen von statischen SVG−Graphiken sein.

Server− und Clientseitig erzeugte SVG−Dateien
Auf den Bezug von SVG zu XML wurde bereits hingewiesen. Serverseitig ist es daher **einfach** möglich mit Techniken wie Perl, PHP und Java und XML−Daten mit dem dazugehörigen Stylesheet (XSLT) in SVG− oder entsprechend strukturierte Dokumente serverseitig herzustellen. Anstelle der XML−Daten kann auch eine entsprechende Datenbank (z.B. PostgresGIS, Oracle Spatial oder IBM DB2) treten.

Clientseitig kann SVG in Verbindung mit JavaScript **schnell erzeugt** bzw. abgeändert werden. Zukünftige Browser werden zudem XSL (Extensible Style Language), das Verbindungsstück zwischen XML (SVG) und HTML, unterstützen. Das bedeutet, dass die Browser SVG direkt anzeigen können. Die Daten können dabei in eigenen XML−Dateien, als JavaScript−Arrays oder in versteckten Feldern übertragen werden.

Arbeitsabläufe und Optimieren von SVG Code

Im folgenden Abschnitt soll nur auf Probleme im Arbeitsablauf hingewiesen werden, die im Zusammenhang mit SVG entstehen. Ganz allgemein lässt sich ein Arbeitsablauf jedoch nicht darstellen. Wir empfehlen darum, beim Durchführen eines grossen Kartenprojekts den ganzen Ablauf zuerst an einem einfachen Beispiel auszuprobieren.

Grobkonzept
Bevor mit der Herstellung der Karte begonnen wird, muss das **Zielpublikum** identifiziert werden. Daraus resultiert wie die Karte etwa aussehen soll und welche Interaktionen integriert werden sollten. Es sollte auch klar werden, ob dynamische Elemente nötig sind. Schliesslich muss festgelegt werden ob und wie das SVG−File in eine HTML−Seite eingebunden werden soll. Dabei wird auch das Layout der HTML mit CSS−Layern, Tabellen und Frames festgelegt. Als Beispielkarte dient etwa (http://www.karto.ethz.ch/teaching/dip_2001_joray/).

Erstellen einer Grundlagenkarte
Es ist theoretisch möglich, eine Karte in SVG mit einem Texteditor zu erstellen. Die SVG−Daten werden natürlich aus bereits bestehenden Vektordaten konvertiert. Es ist möglich, dass die Grundlagenkarte in einem Format vorhanden ist, das nicht direkt in SVG konvertiert werden kann. In diesem Fall lohnt sich der Versuch, die Karte in einem alternativen Vektorprogramm zu öffnen. ’.xfig’ kann z.B. mit Kontour gelesen werden. Kontour selbst enthält einen SVG−Export. Ebenso kann Illustrator FreeHand−Formate lesen. Sketch kennt u.a. ’.ai’, ’.wmf’ und Corels ’.cmx’.

Die Grundlagenkarte lässt sich in diesen Vektorzeichenprogrammen gut bearbeiten. Dabei sollte darauf geachtet werden, dass bereits hier die Karte in verschiedene, **sinnvolle Layer** unterteilt wird. Diese Unterteilung zeigt sich später auch im SVG−Code. Das Nachbearbeiten des SVG−Codes sowie das Einfügen von Interaktion und dynamischen Elementen kann so erheblich erleichtert werden. Es lohnt sich auch die Frage zu stellen, ob eine **Kartenlegende** nicht als separates SVG−File behandelt werden soll. Eine Verknüpfungen zur Karte via HTML/DOM ist ohne weiteres möglich.

Symbole werden häufig als einzelne Pfade exportiert. Es lohnt sich daher kaum in dieser Arbeitsstufe Symbole in die Karte einzubauen. Im Gegenteil, bereits vorhandene Symbole werden besser entfernt. Sie werden **später** eingesetzt. Es besteht die Möglichkeit verschiedene Schriften in ein SVG−File als PostScript einzubauen. Es ist jedoch erwünscht, möglichst wenig Schriften einzubauen, da sonst das SVG−File unnötig gross wird. Text wird auch mit Vorteil zu einem späteren Zeitpunkt in die Karte integriert. Die bearbeitete Grundlagenkarte sollte den gängigen (web−)**kartographischen Regeln** entsprechen. Erst dann sollte sie nach SVG exportiert werden.

Erstellen von Symbolvorlagen
Symbolvorlagen können einfach unabhängig von der Karte mit einem Autorentool wie ’Jasc WebDraw’ hergestellt werden. Ein Symbol wird mit den Malwerkzeugen gezeichnet. Anschliessend separiert man den SVG−Code des Symbols und fügt ihn mit <symbol name=’Symbolname’>’Symbolpfad’</symbol> in der Grundlagenkarte ein. Typischerweise kommt die Symbolbeschreibung kurz nach dem Header des SVG−Files. Es gilt dann zu beachten, dass der Bildmassstab der Symbolvorlage dem Bildmassstab der Grundlagenkarte entsprechen sollte. Sonst wird das Symbol zu gross oder zu klein abgebildet. Nachträgliches skalieren ist jedoch einfach möglich.

Setzen einzelner Symbole und Texte in der Karte
Beachtet werden muss, dass die SVG−Viewer das File von oben zu lesen beginnen. Was zuerst kommt, wird zuerst gezeichnet. Der Code für die Symbolplatzierung muss deshalb erst **zuunterst** im File stehen, damit die Symbole sichtbar sind. Wenn man ein Symbol nur wenige Male in der Karte platziert werden, so lohnt es sich oft, dieses ’von Hand’ einzufügen. Kommt ein Symbol öfters vor und sind die Bildkoordinaten bereits in digitaler Form vorhanden, so lohnt es sich mit sed/awk, Perl oder PHP ein kleines Script zu schreiben. Wer lieber eine höhere Programmiersprache verwendet, kann mit C/C++ oder Java ein kleines Programm schreiben. Das Script bzw. das Programm kann zumeist für andere Projekte wiederverwendet werden. In diesem Arbeitsschritt kann man auch noch weiteren **Text** einfügen. SVG stellt dazu einen eigen <text>−Tag zur Verfügung. Dadurch, dass der Text nicht als Pfad in der Karte eingefügt ist, besteht die Möglichkeit eine Vielzahl von Interaktionen auf den Text anzuwenden.

Einfügen von dynamischen Elemente und Interaktionsmöglichkeiten
Es sei hier noch einmal darauf hingewiesen, dass **nicht** möglichst viel Interaktion und viele dynamische Elemente automatisch eine gute Karte/Graphik geben! Es besteht die Möglichkeit in SVG jedem Element Dynamik zu verleihen. Man kann Flächentöne

beliegig zeitlich variieren. Symbole können sich entlang einer Linie bewegen. Es können ganze Ebenen ein− und ausgeblendet werden. Die **Dynamik/Animation** ist häufig auf genau ein Element angepasst. Es lohnt sich daher eher selten diesen Prozess zu automatisieren. Meist arbeitet man mit einem geeigneten Editor oder einem Autorensystem. Sollte der Prozess trotzdem automatisch ablaufen, können hier wieder Perl− oder PHP−Scripte und/oder kleine Java−oder C/C++ Programme zur Anwendung kommen.

Interaktion in SVG kann mit Mauseffekten geschehen. Es besteht aber auch die Möglichkeit Elemente (Pfade, Symbole, Text) miteinander, mit HTML−Menüs oder mit anderen SVG−Files und deren Elementen zu verbinden. Vielfach betrifft eine Interaktionsmöglichkeit mehrere Elemente. Gerade hier ist der Einsatz von sed/awk−, Perl−, PHP−Scripten oder Java−, C/C++Programmen sinnvoll. Die Interaktion kann als JavaScript realisiert werden. Funktionen müssen dabei nur einmal geschrieben werden, bzw. sie können aus dem Netz kopiert werden. Manchmal ist es programmiertechnisch einfacher Interaktion und Dynamik bereits zu einem früheren Zeitpunkt in den SVG−Code einzuführen. Sie müssen das allerdings von Fall zu Fall selber entscheiden.

Erstellen einer HTML−Seite
Die Techniken zum Erstellen einer HTML−Seite sollten bekannt sein. Im Sinne einer guten Lesbarkeit sollte man die Seite aber nicht überfüllen. Auch die Farb− und Schriftwahl sollte sich der Karte anpassen und sorgfältig erfolgen. Weniger ist oft mehr!

Ausgewählte kartographische Beispielprojekte

Social Patterns in Vienna
Erstes wirklich interaktives Kartographie−Beispiel implementiert in SVG. Der Benutzer kann sozioökonomische Variablen wählen, Anzahl der Klassen (Quantile) festlegen und interaktiv Werte abfragen. Zur Navigation ist eine Übersichtskarte verknüpft. Autor: Andreas Neumann, URL: http://www.karto.ethz.ch/neumann/cartography/vienna/.

Prototyp eines OECD Europa−Atlas
Diplomarbeit an ETH Zürich/Universität Wien. Interaktive Wirtschaftskarten mit inte−grierten Diagrammen (Kreisdiagramme, Kreissektorendiagramme). Die Diagramme werden »on the fly« aus den Daten generiert. Der Benutzer kann Farbgebung und Dia−grammgrösse beeinflussen. Zusätzlich können Schatten den Diagrammen hinterlegt werden. Sortierbare statistische Tabellen sind mit den Ländern verknüpft. Autor: Andréas M. Winter, URL: http://www.carto.net/papers/svg/eu/.

Migration zwischen der Schweiz und Europa
Umfangreichere SVG−Karte, die Migrationsdaten zwischen der Schweiz und Europa mit Flächenfarben und Diagrammen kartographisch darstellt. Es können viele Parameter (z.B. Geschlecht, Zeitraum, Richtung, etc.) und verschiedene Klassierungsmethoden gewählt werden. Verlaufskurven werden direkt aus den Daten generiert, die Darstellun−gen können auch animiert werden. Autor: Christian Joray, URL: http://www.karto.ethz.ch/teaching/dip_2001_joray/.

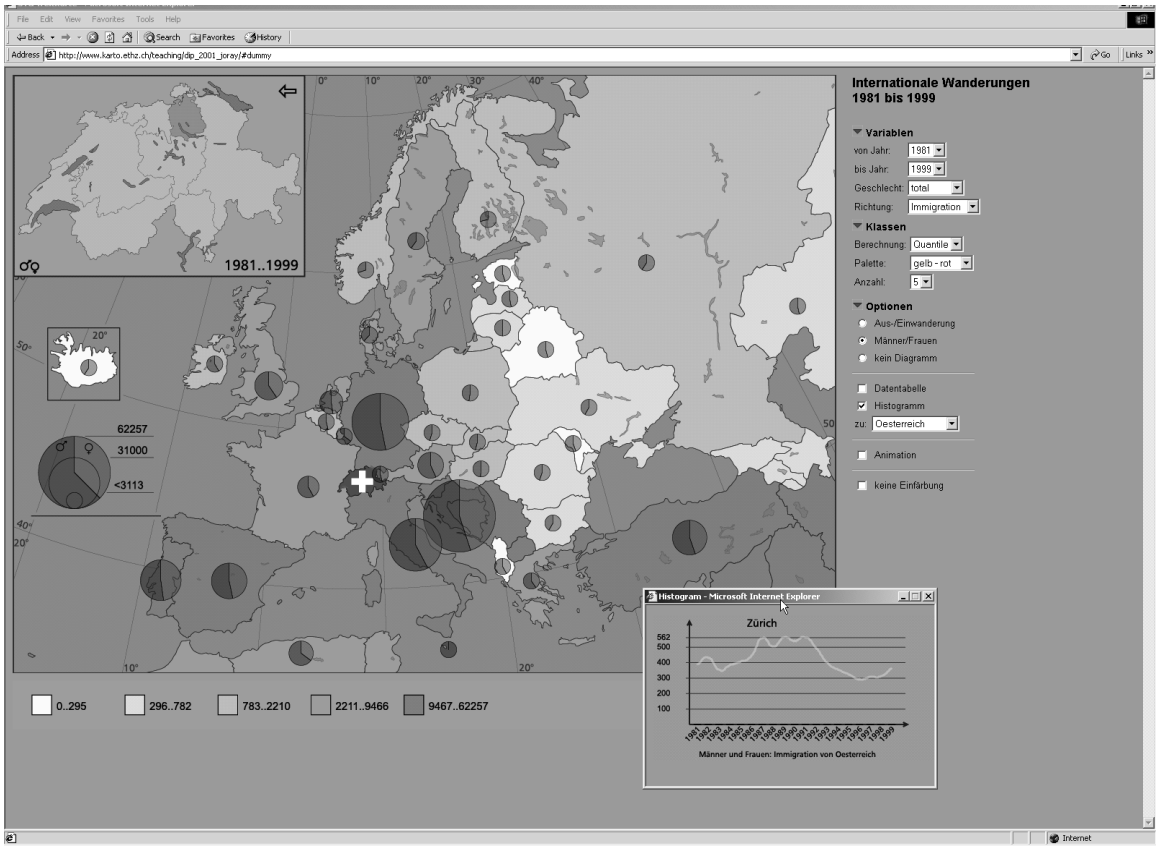

Abb. 3: Migration zwischen der Schweiz und Europa;
interaktive kartographische SVG Anwendung

Taxes in Zurich, Zug and Schwyz
Interaktive thematische Karte zur Steuerbelastung in drei Kantonen der Schweiz. Mit Hilfe von »mouse−over«−Effekten werden dynamisch Kreisdiagramme beim überfahren einer Fläche angezeigt. Daneben können die administrativen Einheiten (Gemeinde, Be−zirk, Kanton), die Variablen und die Klassierungsmethoden spezifiziert werden. Zusätz−lich können mit Hilfe von Attribut−Daten Selektionen durchgeführt werden. Autoren: Christian Hilber, Stefan Ziegler, URL: http://www.karto.ethz.ch/teaching/vtb_2001_hilber_ziegler/.

Aktuelle und zukünftige Entwicklungen

Die Einführungsphase eines wohlüberlegten Standards wie SVG (er hat ca. 2 Jahre Entwicklungszeit hinter sich und wurde von den »Cracks« der Computergraphik entworfen) ist eine spannende Zeit. Man kann mit eigenen innovativen Beiträgen zur Community beitragen und Einfluss auf Weiterentwicklungen nehmen. Es ist heute nur schwer abzuschätzen wie schnell sich SVG weiterentwickeln wird. Klar ist, dass wir bezüglich der Anwendungs− und Integrationsmöglichkeiten erst die »Spitze des Eisberges« kennen. Offene Standards und die solide XML−Basis sowie Scripting− und Programmiermöglichkeiten garantieren für die Zukunft maximale Flexibilität und Integration. Gut ist auch, dass Sie mit SVG von keiner Firma abhängig sind und mit keinen bis geringen Lizenzkosten bei der Implementierung rechnen können.

Aus kartographischer Sicht ist es natürlich spannend die interaktiven Möglichkeiten

weiter auszuloten/auszubauen. So könnten Benutzer beispielsweise online Selektionen und Datenerfassungen vornehmen und serverseitige Datenbanken abfragen/ergänzen. Grössere Datenbestände bindet man serverseitig am besten mit räumlichen Datenbanken und GIS−Systemen an. Einfache GIS−Analysen kann man zukünftig direkt in das Client−System einbauen, komplexere Aufgaben delegiert man an einen oder mehrere Server. »Distributed Mapping on Demand« kommt mit SVG, PDF, Datenbanken und der Web−Infrastruktur jedenfalls um entscheidende Schritte näher. In welche Richtung clientseitige Interaktion in Zukunft gehen könnte, zeigen Adobe [1, ADOBE SYSTEMS, 2001] mit einem Prototyp eines Online−SVG−Zeichenprogrammes (SVGDraw) und Kevin Lindsey [10, Lindsey, 2001] mit seinen Experimenten mit SVG und GUI (Graphical User Interface).

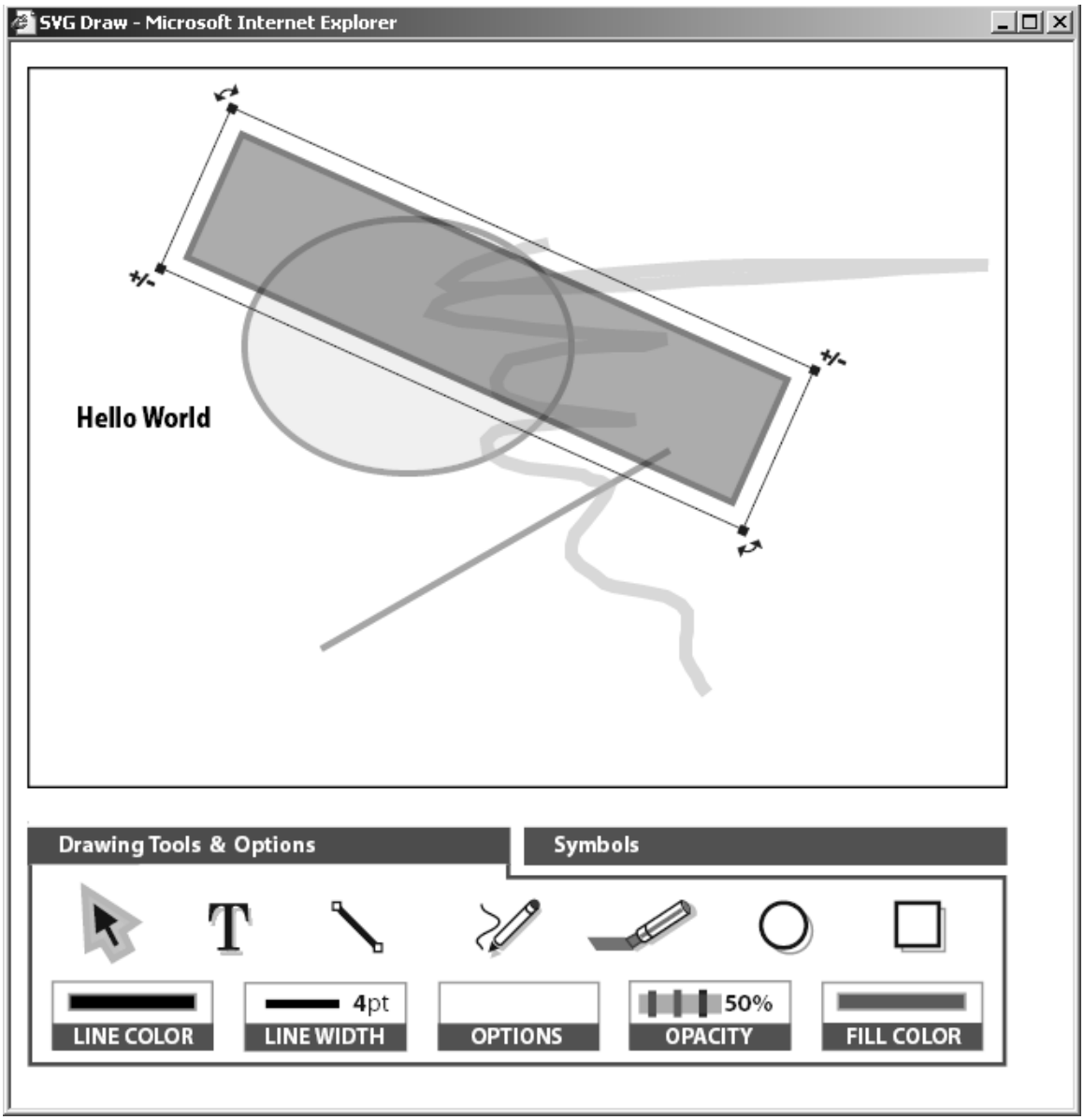

Abb 4: Adobe SVGDraw, ein interaktives SVG−basiertes Online−Zeichentool

Für den Einsatz in der Schulkartographie bietet sich SVG derzeit besonders an: Kleinmassstäbige Daten sind weniger sensitiv hinsichtlich Datenschutz und die ohnehin häufig unter »Sparmassnahmen« leidende Schulbuchbranche könnte mit webbasierter Distribution erhebliche Kosten sparen. SVG bietet attraktive Möglichkeiten hinsichtlich Interaktion und individueller Kartengestaltung. Lehrer wie Schüler könnten selber gestalterisch eingreifen und online explorative Visualisierung und Analysen betreiben. Da kleinmassstäbigere Karten meist auch kleinere Datensätze impliziert, können die gesamten Projekte auch zum Download angeboten werden und den Schülern zur Weiterverarbeitung/Anpassung zur Verfügung gestellt werden. Korrekturen und Ergänzungen sind mit geringem Aufwand und Kosten zu realisieren.

SVG Mobile Working Group, SVG 1.1

Diese neu etablierte W3C Arbeitsgruppe diskutiert die Anforderungen für SVG auf mobilen Geräten, wie Handys und PDA's. Die Gruppe schlägt vor, zwei separate Profile einzuführen, die ein Subset der derzeitigen SVG 1.0 Spezifikation darstellen: SVGB (SVG Basic) für stärkere PDA's und Mini−Computer und SVGT (SVG Tiny) für »low−level« Geräte mit noch geringerer Rechenleistung, Hauptspeicher und Netzwerkbandbreite. Der derzeit diskutierte Vorschlag (requirement analysis) mit einer Auswahl der Features, Elemente und Attribute für SVG Mobile kann unter [7, GRAHAM/CAPIN, 2001] nachgelesen werden. Die Gruppe sieht für SVG Mobile die folgenden Anwendungsbereiche:

- Location−Based Services
- Kartographie und Verortung
- Versand animierter Botschaften und Grüsse
- Multimedia Nachrichten
- Unterhaltung
- Industrielle Anwendungen mit Bedarf für mobile Geräte
- eCommerce
- Benutzerschnittstellen

SVG 1.1/2.0 Working Group

Obwohl die SVG 1.0 Spezifikation erst kürzlich eine W3C−Empfehlung wurde, diskutiert eine weitere neue Arbeitsgruppe bereits mögliche Erweiterungen für das nächste kleinere Release (SVG 1.1) und den grösseren Upgrade der Spezifikation hin zur Version 2.0 (siehe [9, JACKSON, 2001]). Für SVG 1.1 werden im wesentlichen die Bedürfnisse von SVG für mobile Geräte eingearbeitet. Es wird die modulare Struktur der Profile eingeführt, ein Weg der bereits auch bei der X3D Spezifikation begangen wurde. Zusätzlich zum SVGT und SVGB Profil wird wahrscheinlich auch ein SVG Printing Profil ausgearbeitet. SVG 2.0 wird ein grösseres Upgrade. Da die Resultate der W3C Arbeitsgruppen (Bedarfsanalyse, Entwurf und Spezifikation) öffentlich diskutiert werden, wird das Feedback von der Öffentlichkeit wie auch von eingeladenen Spezialisten, wie auch Mitgliedern der W3C Arbeitsgruppen gerne entgegengenommen und nach Möglichkeit eingearbeitet.

Unter den Zielen der Arbeitsgruppe ist auch die Plazierung von SVG als Standard−Applikation auf den Desktops der Benutzer (innerhalb von Web−Browsern, Graphik−Applikationen, Autorensysteme und auch als Austauschfomat), mobilen Geräten (in deren Browsern, als User−Interface und auch in der Autonavigation), in Druckern und industriellen Geräten mit graphischen Anwendungen. Obwohl dieses Ziel ehrgeizig klingen mag, bietet es eine grosse Chance für die Software−Industrie: sämtliche graphikorientierten Applikationen können erstmals auf einem durchdachten Standard aufbauen, der bereits die meisten Anforderungen an strukturierte interaktive Graphik erfüllt, bei Bedarf aber sehr leicht erweiterbar ist. Die Arbeitsgruppe will SVG vor allem aber auch als Austauchformat für die verschiedensten vektorbasierten Graphik−Applikationen etablieren. Das W3C wird zwar die Aufsplitterung in verschiedene Profile wegen der unterschiedlichsten Bedürfnisse der Ausgabegeräte zulassen, will aber für sämtliche Profile umfassende Testsuiten bereitstellen um zukünftige SVG−Software auf deren Kompatibilität und Standard−Konformität zu testen.

Es ist wohl wenig sinnvoll an dieser Stelle sämtliche Vorschläge im Detail zu diskutieren. Die Verbesserungen/Erweiterungen sollen v.a. verschiedene Anwendungsbereiche aus dem Kartographie, GIS, CAD und Design−Umfeld betreffen. Sie können die Bedarfsanalyse unter [6, JACKSON, 2001] studieren und selber sinnvolle Vorschläge der W3C−Arbeitsgruppe übermitteln. Wir denken, dass die Chance, an der Verbesserung offener Standards mitzuwirken von der interessierten Öffentlichkeit auch wahrgenommen werden sollte!

Wir wünschen allen Kartographen/Graphikern viel Spass beim »Entdecken/ Implementieren« des SVG−Standards und den neuen Möglichkeiten die sich dadurch für ihre Web−Applikationen ergeben! Die untenstehenden Ressourcen werden eine wertvolle Hilfe sein. An Zusendung von Links zu ihren fertigen Projekten sind wir interessiert!

Resourcen

SVG Spezifikation:
http://www.w3.org/TR/SVG/
http://www.w3.org/TR/SVGMobileReqs (Working Draft!)
http://www.w3.org/TR/SVG2Reqs (Working Draft!)

Ausgewählte SVG News−, Tutorial−, Links− und Demo−Seiten:
http://www.w3.org/Graphics/SVG/Overview.htm8 (News und Links)
http://www.adobe.com/svg/community/external.html (Links)
http://www.adobe.com/svg/demos/main.html (Dynamische SVG Beispiele)
http://www.adobe.com/svg/basics/intro.html (SVG Entwickler tutorial)
http://www.kevlindev.com/ (SVG tutorial und Beispiele, SVG und GUI)
http://www.carto.net/papers/svg/ (SVG tutorial, Beispiele, Artikel)
http://www.pinkjuice.com/SVG/SVGlinks.htm (SVG links)
http://www.sun.com/software/xml/developers/svg/ (SVG bei Sun)
http://www.svgopen.org/ (Erste internationale SVG Entwickler−Konferenz, Zürich 2002)

Ausgewählte SVG Implementationen:
http://www.adobe.com/svg/ (Heimat des Adobe SVG Viewer)
http://xml.apache.org/batik/ (Heimat des Batik, Java2 basierten SVG Viewer)

http://www.alphaworks.ibm.com/tech/svgview (IBM SVG Viewer)
http://sis.cmis.csiro.au/svg/ (CSIRO, SVG toolkit)
http://www.croczilla.com/svg/index.html (Mozilla SVG)
http://www.in−gmbh.de/en/Products/in−gmbh/sphinxopen/editor.htm (SVG Editor)
http://www.jasc.com/products/webdraw/ (Kräftiges SVG Autorensystem)
http://broadway.cs.nott.ac.uk/projects/SVG/svgpl/ (SVG Perl Module)
http://sketch.sourceforge.net/ (Vektorgraphik−Software für Linux, Open Source, SVG exp.)
http://koffice.kde.org/kontour/ (Vektorgraphiksoftware f. Linux, Open Source, SVG im/exp.)
http://www.svgmaker.com/ (SVG Druckertreiber, ähnlich Acrobat Distiller)
http://www.mayura.com/ (Graphik Software mit SVG Export)

SVG Mailinglisten und Discussions−Gruppen:
http://groups.yahoo.com/group/svg−developers
http://www.carto.net/papers/svg/mail_e.html (Kartographie spezifisch)
http://www.adobe.com/support/forums/main.html (siehe SVG Forum)
http://lists.w3.org/Archives/Public/www−svg/

Literatur

[1] Adobe Systems (2001): »Adobe SVG Zone − SVGDraw and other Demos«, http://www.adobe.com/svg/demos/main.html

[2] Berners−Lee, Tim (2001): »Longer Bio for Tim Berners−Lee«, http://www.w3.org/People/Berners−Lee/Longer.html

[3] Berners−Lee, Tim (1998): »Axioms of Web−Architecture«, http://www.w3.org/De−signIssues/Principles.html

[4] Conolly, Dan (1999): »CGI: Common Gateway Interface«, http://www.w3.org/CGI/

[5] Ferraiolo, Jon (2001): »Scalable Vector Graphics (SVG) 1.0 Specification«, http://www.w3.org/TR/SVG/

[6] Fibinger, Iris (2001): »Scalable Vector Graphics (SVG) − Untersuchung des XML−Standards für zweidimensionale Vektorgraphiken und Erstellung eines SVG−Tutorials«, Diplomarbeit an der Fachhochschule Karlsruhe, Fachbereich Sozialwissenschaften, Studiengang Technische Redaktion, Karlsruhe. (http://www.fbwi.fh−karlsruhe.de/lin02/xmldirectory/ver/diplom_if.pdf)

[7] Graham Rick und Tolga Capin (2001): »SVG Mobile Requirements«, http://www.w3.org/TR/SVGMobileReqs

[8] Hurni Lorenz, Neumann Andreas und Andréas M. Winter (2001): »Aktuelle Web−techniken und deren Anwendung in der thematischen Kartographie und der Hochge−birgskartographie«, in: Buzin/Wintges, »Kartographie 2001 − multidisziplinär und mul−tidimensional«, Beiträge zum 50. Deutschen Kartographentag; Wichmann−Verlag, Hei−

delberg. (http://www.carto.net/papers/svg/articles/paper_dgfk_berchtesgaden_2001.pdf)

[9] JACKSON, Dean (2001): »SVG 1.1/2.0 Requirements«, http://www.w3.org/TR/SVG2Reqs

[10] LINDSEY, Kevin (2001): »KevLinDev − SVG Resources«, http://www.kevlindev.com/

[11] MACROMEDIA CORP. (2001): »Macromedia History Timeline«, http://www.macro−media.com/macromedia/proom/corpinfo/docs/mm_timeline.pdf

[12] NEUMANN Andreas (2001): »Comparing .SWF and .SVG file format specifications«, http://www.carto.net/papers/svg/comparison_flash_svg.html

[13] NEUMANN Andreas und Andréas M. WINTER (2001): »SVG − Scalable Vector Gra−phics; Ein zukünfiger Eckstein der WWW−Infrastruktur«, in: Tagungsunterlagen zur Internationale Internet− und Multimedia−Tagung (UGRA), WTC Zürich. (http://www.carto.net/papers/svg/articles/paper_ugra_zurich_2001.pdf)

[14] WINTER, Andréas und Andreas NEUMANN (2000): »Vector−based Web Cartography: Enabler SVG«, http://www.carto.net/papers/svg/index_e.html

[15] ZAKON, Robert H'obbes' (2001):, »Hobbes' Internet Timeline v5.4«, http://www.zakon.org/robert/internet/timeline/