TIME FOR SVG[1] – TOWARDS HIGH QUALITY INTERACTIVE WEB-MAPS

Andreas Neumann
and Andréas M. Winter
Institute of Cartography, ETH Zurich
ETH Hoenggerberg, CH-8093 Zurich
Freytag & Berndt, Austria
Brunnerstr. 69, A-1231 Vienna

Fax: ++41-1-6331153
neumann@karto.baug.ethz.ch
http://www.karto.ethz.ch/neumann/

Fax: ++43-1-8699090-38
andre.mw@gmx.net
http://www.carto.net/
http://www.freytagberndt.at

topic 28: web mapping

Keywords: SVG, webmapping, interactive thematic mapping

Abstract

Needless to say that there is a high demand for maps on the internet. Several papers had dealt with issues on internet map use and the exploding numbers of hits on busy cartographic websites [7]. However, most of the presented maps are of low (carto)-graphical quality and none or little degree of interactivity. This fact is due to a lack of standardized technology and limited know-how in programming, graphics design and additional web-techniques. With the rise of SVG there is for the first time a technology at hand that allows to represent all graphical elements producable by graphics and cartographic-software with the additional advantages of high interaction possibilities and animation, all based on open and standardized file formats and programming languages. This contribution summarizes the most important techniques involved in high-quality interactive web-mapping, presents SVG's capabilities and shows possible workflows. Several prototype implementations presented at the congress show that the SVG-technology is ready for deployment and SVG-based web-maps are relatively easy to implement. The examples are referenced at http://www.carto.net/.

Introduction

Before the rise of vector-based graphics formats most of the web-maps had been quite static or with minimal interaction possibilities. Webmapping services that work with vector data, f.e. for routing and address matching, had to first convert the vector data on the server-side to raster formats, thus limiting the clients capabilities in interaction and graphical quality. When trying to implement interactive maps based on raster layers, cartographers had to work out complex layer- and Javascript constructs that have by no means been browser-independent. SVG was not the first try to establish a vector-standard for the web, but is today the most promising one. The first attempts go back to 1993. Today there are mainly two alternatives: SVG and Flash. Both have quite similar capabilities concerning graphical objects, effects and animation. However, Flash is proprietary, depending on one single company, not very well integrateable into complex web-projects and not easy for the cartographer to generate and edit.

In order to understand SVG's architecture and to learn about accompanying technologies we have to explain the most basic terms and components that collaborate with SVG in order to fully provide interactive, high-quality web-maps:

Technical Terms - Components of a web-mapping system

XML

XML is the future base-technology for all coming web-standards. The basic idea is not really new, but derived and

[1]SVG is an acronym for Scalable Vector Graphics

simplified from SGML[2], with a strict separation of the content, a syntax description and validation, the formatting and output. XML is like HTML a tag based text format, easily readable by humans, platform- and application independent and ready for archiving. Based on XML there are several extended base-technologies, f.e. for file format translation (XSLT), for formatting (XSLFO), for syntax description and validation (DTD and Schema), for querying data (XQL - not fully implemented yet) , for hyperlinking (XLL and XPOINTER). Building on this base technology, different XML-dialects had been introduced, often related to specific domains. Examples include CML (Chemical Markup Language), GML (Geography Markup Language), SMIL (Multimedia), SVG (2D Vector Graphics) and X3D (3D Graphics, similar to VRML). One of the advantages of using XML is that all further implementations and dialects are based on the same syntax rules and can use similar parsers that are available for all of the popular programming languages. This speeds up the programming and content creation process and helps to leverage knowledge.

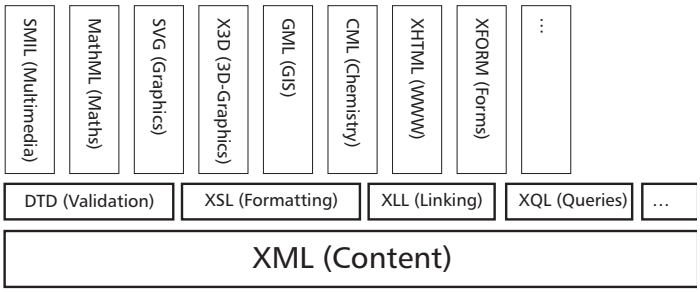

Figure 1: XML architecture: Basic techniques and domain-specific file-formats.

DOM

The DOM (Document Object Model) provides a unique identifier and access to all objects of a webpage, including tables, images, graphical objects, etc. This is implemented in a hierarchical way so that a script can identify and modify each single object. The main issue of a clean DOM is that web-based applications usually use a lot of different formats, engines and plugins. However, communication often stops at applications, based on proprietary plugins.

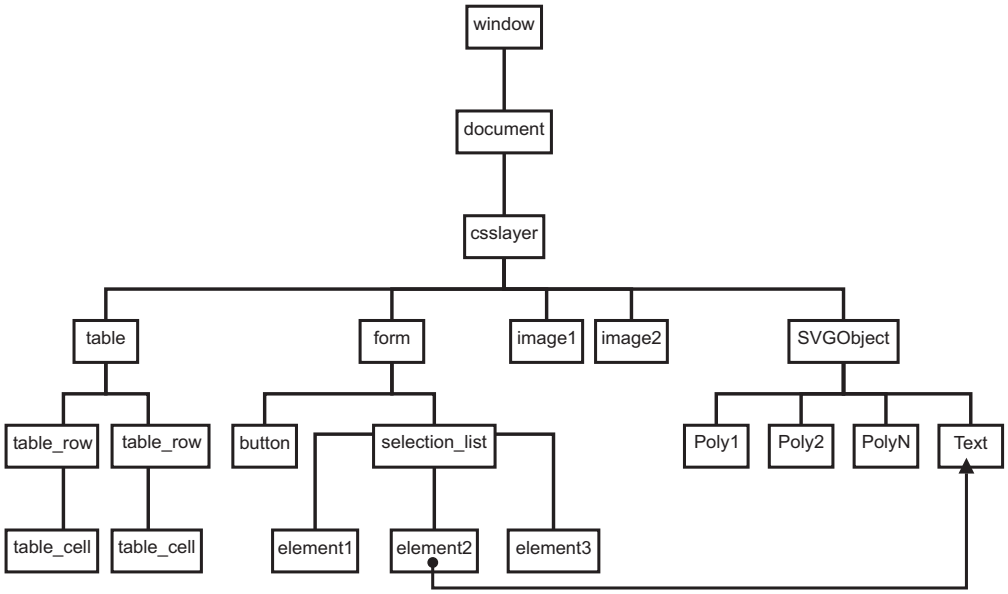

Figure 2: Example for a simple DOM of a typical webpage.

One way for those to communicate with each other is to address them by their hierarchical position. In Figure 2 the element "text" of the svg-graphics can access the value of "element2" in the selection-list of the form-element and set the text in the SVG-file. A javascript-code could look like the following:

```
window.document.csslayer.mySVG.text = window.document.csslayer.form.selection_list.element2.value;
```

[2]SGML is an acronym for Structured Generalized Markup Language, introduced in the late 1980ies for documentation issues and

Javascript

Javascript, a scripting language that is validated, compiled and executed on the fly, was introduced by Netscape Corporation in the beginning of the 90ies. Meanwhile it had been accepted by webmasters and programmers, and implemented in all of the major web-browsers. The syntax is closely related to C++ and Java. Though it is a language relatively easy to learn, it is a quite complex and powerful programming language. One of the not well-known but existing features is object-orientation, incl. inheritance and encapsulation. One should not mix up Javascript and Java – those are two completely different implementations. We use Javascript for client-side dynamic content creation, interactivity, event handling and animation, mainly through the DOM.

Relational Databases and SQL

Relational Databases are used to store tabular and geographic data in an efficient way while allowing easy updating and querying. Interaction with the database is done by using SQL, a standardized querying language. Available data-types include various number formats, monetary types, boolean types, character and text strings with several length-limits, and BLOBS (Binary Large Objects) to store large amounts of binary data. Some databases, like Oracle, IBM DB2 and Postgres additionally offer geometric and/or spatial data-types for storing geographic-features, spatial queries and simple GIS-functions. Databases are perfect complements to SVG and XML. Databases are easier to query and update, XML is perfect for data-exchange and archiving. SVG can be generated out of SQL-Databases.

Serverside Scripts, Java Servlets and CGI

Serverside Scripts may be written in any scripting or programming language. Popular languages include Perl, PHP, Python, C and Java. They are a great way to use databases or any external scriptable application to deliver dynamic content to the client. Only the processed output is sent and not the original data. One of the advantages when using serverside-scripts, is the relative independence of a clients browser capabilities and version, because the interactive part is provided by the server. On the other hand the drawbacks are longer time delays, when doing requests, and the heavy load produced on the server-side. The architecture of server-side techniques is quite open and extensible.

Standards and Open Source Technology

It is important that cartographers stick to official and open standards, when presenting cartographic information on the Internet. Open standards guarantee a longer life-cycle and an easier change to other software products, both on client and server side. There are several standardization organizations, f.e. ISO, ANSI, ECMA, etc. Most relevant for web-publishers and internet software developers are the recommendations worked out and published by the W3 ORG (WWW consortium) [9] - a consortium consisting of universities, institutions and companies. Standards, published so far, include several XML and XML-related base-standards, web graphics, multimedia, payment, security, accessibility, internationalization, etc.

For almost all web-techniques, both client and server, one can find open-source solutions. Well-established open source projects not only save time and money but are in most cases better tested and documented. Recent studies have shown that security-related patches are usually quicker released for open-source software than for commercial one. Some open source packages such as Apache (Webserver), Tomcat (Java Servlets), Linux and FreeBSD (Operating-systems), Perl, PHP, Python (Scripting Languages), Progress and MySQL (databases), bind (DNS-Server), Gnu-Compilers, Routers, etc. are widely used and build the base of the Internet infrastructure. Our policy is to use and support open-source software wherever it is mature and well-documented.

Other vector-based approaches

As already mentioned, efforts to establish vector-based web-formats may be traced back to 1993. It begun with SVF (Simple Vector Format) and leads to nowadays frequently used standards, like Macromedia Flash and PDF, besides using Java-Applets. None of them are fully appropriate for cartographic requirements, which might be the reason of the usually quite poor (carto-)graphical quality of todays map-servers. In Table 1 you find an overview of most graphic formats in use on the Web. The currently most important ones are discussed below.

DWF: Drawing Web Format, to be visualised with AutoDesk's WHIP4 plugin and based on DWG. Has some interesting features like highlighting of hyperlinked elements, but is essentially designed for displaying technical drawings. Not useful for complex maps.

Flash: Currently the most widespread vector format that allows interaction. It runs as a Macromedia plugin within the browser. Before the rise of SVG this was the most up-to-date standard for the representation of vectors. However, this format has been primarily designed for advertising and multimedia, and has, from the cartographic point of view, some shortcomings that are better implemented in the SVG format. After export from Macromedia products there is no easy access to the source code. Scripting works since the last version but there is still no easy interaction handling offered to developers and cartographers.

PDF: Portable Document Format. In the pre-press stage it is more and more replacing PostScript and EPS. Not primarily designed for WWW, yet it may be used with the help of plugins. There are export filters for most common text editing, graphics and DTP programs. It's viewer, a free courtesy of Adobe, operates indepentently from a browser and is cross-platform. Documents may also be password-protected to prevent reading, printing, or copying. The binary format version is, unfortunately, not readable by WWW search enginges. PDF is basically designed for static graphics. Hyperlinking is possible, but does not provide sufficient interaction.

Java2D: This is not a graphics format but a very sophisticated graphics library for Java developers. Because Java is widely used on the web, this is an interesting way for skilled cartographers to present cartographic data. Any data-format (also encrypted) may be parsed and rendered. Interaction needs to be programmed, but offers a lot of flexibility. The results are of high graphical quality, including anti-aliasing features, transparency and complex text-rendering. Java2D is often used to render some of the above formats, incl. SVG. The Java2D library is available as source-code.

Format	visualisation module	use	interactivity level	internal format
SVF	plugin	out-dated	1	binary
DWF	plugin/applet	rare	2	binary
Flash	plugin	frequent	3	binary
PDF	plugin	frequent	1	binary/ascii
SVG	browser/plugin	rare (new)	4	ascii
PGML	[2]	[2]	3	ascii
WebCGM	browser/plugin	rare	2	binary
HGML	[2]	[2]	1	ascii
DrawML	[2]	[2]	0	binary
VML	browser	rare [3]	1	ascii
Java2D [4]	applet	rare (new)	4	binary
ActiveX [4]	browser	frequent [3]	4	binary

[2]) format specified but not implemented

[3]) only MSIE4.0+

[4]) not a Graphics format, but gra-phics library for programmers

*) 0: simple display

1: zoom, layers, links on objects
2: external scripts accessing graphics
3: animation
4: full control on objects and animations

Table 1: Overview on Vector formats on the Web.

SVG's capabilities

The following passages are based on the W3C SVG specification [11], Adobe SVG ressources [1] and the SVG-pages on Carto.net [6].

An open Standard and W3C recommendation

SVG is the only, of the above vector-standards, discussed and developed by members of open source and the leading graphics-softare development companies. Besides WebCGM, it is the only standard that has become an official W3C recommendation. Companies that joined development include Adobe, Apple, Autodesk, Corel, HP, IBM, Kodak, Macromedia, Microsoft, Netscape, Quark, Sun Microsystems, Visio and Xerox. This ensures a quick integration into above companies' products. Many of them, as well as a lot of open source projects, have already included filters into their products. GIS companies like GE-SMALLWORLD and SICAD include SVG into their web-servers.

General concepts

SVG is an XML-based graphics format, readable by humans. The file format starts with a header indicating the format-version, and a link to a DTD (Document Type Definition), provided by the W3C consortium. The header may contain style-sheet-definitions, describing graphical styles that can be referenced later, for symbolizing objects. External style-sheet-files, that can be used for a whole project, may be referenced with a hyperlink. The code-snippet below shows a

complete SVG-header with a viewbox-definition. The viewbox definition may of course be changed on the fly by using scripts.

```
<?xml version="1.0" standalone="no"?>

<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20001102//EN" "http://www.w3.org/TR/2000/CR-SVG-20001102/DTD/svg-20001102.dtd" [

 <!ENTITY fillBuilding "fill:#CCFFD4;stroke:none;"> <!-- green -->
 <!ENTITY fillNodata "fill:#E7E7E7;stroke:black;stroke-width:2;stroke-antialiasing:true;">

<svg id="viennaMap" width="5.14cm" height="4cm" viewBox="611800 -440468 29550 23000"
enableZoomAndPanControls="false">

...
```

A SVG-file may contain one or more viewport definitions, specifying a "viewport coordinate system" (output device oriented) and a "user coordinate system". In our example the "width" and "height"-attribute specifies the map size on paper and a real world coordinate-system in the "viewbox"-attribute. It is important to mention that the origin of the SVG-coordinate system is in the upper left. One can easily convert this with a transform-node or by using negative values along the y-axis.

SVG-Viewers render the graphics-elements in exactly the same order they appear within the file, from top to bottom. This "overpainting-method" of graphical elements is called the "painters model". However, this hierarchy (scene graph) may be changed later with scripting. SVG-Elements may be grouped and can be thus used like layers. Style-sheet definitions and visibility may be attached to a group as a whole. Each element and group should have a unique identifyer in order to later access and change their attributes with scripts. Elements may be once defined and then instantiated later several times. This is of particular interest for cartographic symbol-definitions, markers and the like, in order to keep filesizes small and to efficiently change them on demand.

Example 1: Simple viewing functionality, layers, concepts: http://www.carto.net/papers/svg/canvas_e.html

Example 2: Viewboxes, Zooming: http://www.carto.net/papers/svg/viewbox_e.html

Transformations

Each element and group may be transformed: translated, resized, rotated and skewed. Transformations may be nested (be careful to use the correct order!) or defined using transformation matrices. This way, it is possible to do all affine transformations.

Example 3: Transformations, Matrices: http://www.carto.net/papers/svg/matrix_e.html

Basic geometric elements

Basic SVG-geometry includes rectangles, circles, ellipses, lines, polylines and polygons. Of particular interest is the path-element. Its syntax is related to PostScript or PDF: through commands such as "moveTo" or "lineTo" the path is drawn up. In order to suffice to all demands, cubical and square bezier curves and elliptic curve elements may also form a path. It is possible to have holes in a path-object and to combine disjunct path-elements to one single path. For this purpose it is important to specify the right rule for complex polygons using holes: even-odd or nonzero-winding. One can use either absolute or relative coordinates - using relative coordinates often helps to keep file-sizes small. Path elements can also be used for generating pie-slice diagrams as it is often the case in thematic maps.

Example 4: Basic geometry: http://www.carto.net/papers/svg/shapes_e.html

Clipping, masking and compositing

Any path-object, basic geometry and text object may serve as clipping and masking path or may be clipped and masked (including raster images). Masks may be inserted along the margin of a map for instance, in order to exclude regions. Just as is the case with a path, here too, in case of complex clipping polygons the rule for inlying and/or cross-sectional polygons needs to be indicated.

Example 5: Clipping and masking: http://www.carto.net/papers/svg/clipmask_e.html

Colors, Fills, Color Gradients, Transparency, Stroke-Types, Markers

Colour values are defined in sRGB standard. This is a particular color domain for the Internet with parameters specific to display devices. As with HTML4, values are determined in hexadecimals. Each and every filling, even lines and texts, may take a transparency value. Lines may carry the following attributes: thickness, line end type, vertex markers, dashed or dotted line mode, dash offset, etc. On both ends, markers (such as arrows) may be attached. As a special feature, markers may also be attached to every vertex, which might be helpful for emphasizing minor angle changes.

As to color gradients, linear and radial gradients are supported, with any number of interval and stop values. The spreading method, too, can be controlled. A gradient may even be assigned to opacity. Patterns may be raster and vector based. Those patterns are tiled. A given tile has to be defined once and is repeated according to the tile size. Offset values can be defined as well.

Example 6: Interactive gradients and transparency: http://www.carto.net/papers/svg/fill_e.html

Symbols

The symbol element defines graphical template objects, which can be instantiated by a "use-element". The symbol-element itself is not rendered, only the instances. A symbol-element has the attributes "viewBox" and "preserveAspectRatio" to allow a scale-to-fit within a rectangular viewport defined by the instance-object. This object is of particular interest for us cartographers as we have to deal a lot with signatures and pictograms. The concept of "def" and "use" provides a way to keep file-sizes small and the use of templates helps to efficiently apply changes.

Text and text-along-path

SVG allows for just about any parameters in formatting text elements, such as: font family, font type, size, position, letter spacing and kerning, glyph orientation, inclination, etc. - actually all the parameters known by DTP and cartography. An interesting option for cartographers should be the possibility to align text along path elements. Unfortunately, SVG in its present specification does not know continuous text. In order to properly represent non-latin or self-made fonts as well (e.g. for aligning special symbols along linear elements), SVG allows for external font description files, or for replacement of missing unicode characters by the use of supplementary graphics or symbols. Entire font descriptions may be embedded into an SVG file.

Regarding text formatting, all CSS rules apply, just like with HTML4. Individual characters can be wrought in their orientation. Direction can be chosen, in order to view some non-latin alphabets. Also vertical rows, are represented correctly. Like HTML4, SVG supports Unicode, the international standard for coding all characters of this world. Western languages are part of the 8859-1 (Latin 1) Unicode family. An outstanding feature of the SVG text implementation is, that text elements are indexable by search engines, since the whole graphics is based on XML. This is in contrast to binary graphics, non-readable by search engines. Within the plugin, you can also search for text and highlight it. If the found text is situated outside the present viewport, the extent is adapted accordingly. Centering does not occur automatically, in order to minimize irritating viewport changes.

Example 7: Text and some of the formatting possibilities: http://www.carto.net/papers/svg/fill_e.html

Animation, Motion Path Animations

HTML4, enlarged with CSS and JavaScript, made it possible to create simple animations - this was not perceived in advance, however. For instance, if an object should be moved from A to B, all intermediate steps had to be pre-defined (scripted). In contrast, SVG offers interpolation, and a whole lot more. Syntax and SVG's animation component depends heavily on SMIL, resp. it is SMIL compatible. Amongst others, the following animation variables can be determined: start time of animation (usually relative to an event), duration and end time of an animation, repetition, number of repetitions, etc. Depending on object type, various parameters may be animated, such as: color value, position, position along a path, rotation, scale, and many more.

A time line metapher (as known from multimedia authoring systems) allows you to establish so-called "key frames", that display pre-defined values at a given time. In the meantime, either discrete values are used (key frames only), or interpolation is applied. Interpolation options are: step-by-step, linear, or spline. You may enter "key time values". If you don't, linear interpolation is applied between the intervals. Furthermore a specific motion path instruction is implemented, allowing position animation along path elements, and even accepting a distance-from-path argument. The objects orientation can be aligned automatically to the path. Animations may be combined as you please (cumulated or not), or inherited to other child elements.

Example 8: Simple Animation: http://www.carto.net/papers/svg/anim_e.html

Interaction and Scripting

Basic interaction features of an SVG viewer are zooming, panning, return to original view, and a display print option. Adobe's viewer implemented an additional text search function, an anti-aliasing switch option, a copy feature, and a source code viewer. An option to choose a mouse cursor from a predefined set is planned, resp. a way to add a self-made cursor image. Together with events, simple key functions may be implemented, e.g. drawing applications or distance and area measurement. Hyperlinks, as with HTML, may be used to refer to other files, or to other elements within a SVG document. As mentioned above one can also refer to pre-defined "viewBoxes". In SVG, three event categories are at disposition: mouse events, keyboard events, and state change events (concerning display and SVG file loading state).

Event handling occurs analogue to HTML4. Since all possible events may be combined, almost everything is possible. Complex applications may be realized, combining scripts and Java applets. By JavaScript, the developer is able to fully access SVG's and every browser feature's DOM.

Cartographic applications may cover a large range, such as: simple switching on and off of elements and layers, changing graphical attributes, reacting to mouse events (such as displaying object data on mouse-over), linked windows (combining various views, e.g. overview and main map), interactive moving, scaling and rotating elements (e.g. didactical puzzles), or small applications (e.g. a user option to digitize and to save the data on the server for further editing and storage). Finally, with the help of server CGI scripts, or Java applets, and/or Java servlets, databases may be linked as well. Countless applications may be envisioned, that are bound to enhance cartographer's creativity in future web projects. We are curious to know, just what will be realized in the wake of those new possibilities.

Metadata, Attaching Attribute Data and Extensibility

Metadata serve to store information on the document in a structured way. Data concerning authorship, date of publication, version, title, brief description, etc. may be inserted. There are several ways to integrate attribute data with graphical objects: attachment of the data to the SVG-element as an attribute, embedding of other XML-namespace objects, use of a javascript-array and link to the SVG-object via ID, use of external XML files with an ID-link or use of an external SQL database with an ID-link. As with VRML, SVG offers the possibility to re-use graphical objects defined earlier in the file, given that the original object can be identified explicitly by an ID. Thus you may pre-define more or less complicated graphical objects and groups, in order to re-use them in a modified way, e.g. after transformation.

Extensibility is one of the most important features of SVG, as it is the case with any DOM/XML compatible specification. Since SVG itself is defined in XML, any other standard which is equally defined in XML, such as MathML, XHTML, SMIL, and many others, can be embedded and accessed from within SVG. For example, one could deposit attribute data of polygons, which are part of a thematic map, in XML. Then a JavaScript function would read and process those values in an appropriate way, e.g. as a map diagram dynamically generated using SVG.

Cartographic SVG Implementations

The following, representative, examples are client-based SVG realizations, with emphasis on thematic mapping. They are listed in the chronological order of implementation.

Social Patterns of Vienna

This first example of a thematic map svg-implementation shows interactive choroplethe maps. Various socio-economic variables may be chosen, as well as the number of classes, calculated using quantiles. Interesting distribution and segregation patterns can be thus discovered. Object data and district/subdistrict names are displayed using mouse-over effects. Detailed data are presented on mouse-click using a tabular view. Layers may be switched on and off. An overview-map is linked to the main view in order to allow easy zooming, panning and orientation in the overview context. This capability was first introduced by Gaborit Gaëtan [4] in an SVG-map.

Example 9: Interactive Choroplethe-Map - Social Patterns in Vienna:
http://www.karto.ethz.ch/neumann/cartography/vienna/

Prototype of an OECD-Atlas of Europe

The master thesis of Andréas M. Winter dealt with economic data (mainly gross national product provided by OECD) in Europe based on countries. It was the first SVG-map example that integrated automatically generated pie-slice and wing-diagrams by using the DOM to integrate new elements into the scene-graph. Tabular data views have been advanced by offering sorting capabilities. Colors for the pie-slices may be chosen and filters are introduced for the diagrams to better contrast from each other and the background.

Example 10: Interactive diagram maps - Prototype for an OECD-Atlas in Europe:
http://www.carto.net/papers/svg/eu/

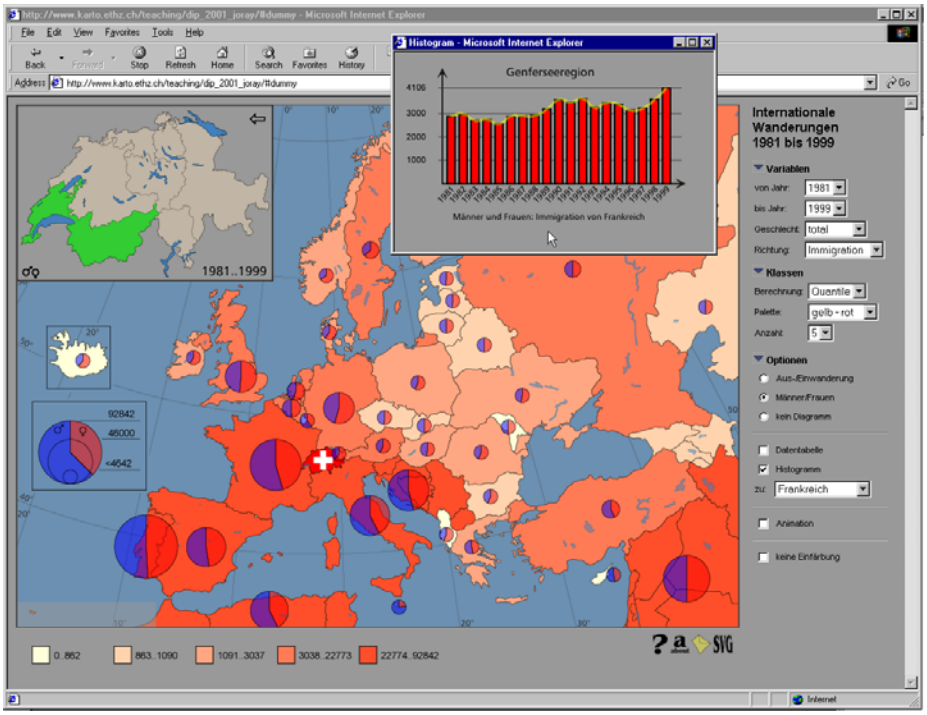

Figure 3: Integrated SVG example: Migration between Europe and Switzerland, by Christian Joray.

Migration between Europe and Switzerland

Christian Joray did his master thesis on an interactive map on migration patterns between Europe and Switzerlands main regions during the time-frame of 1981 and 1999. For this purpose the quite huge amount of data had to be stored into a complex javascript array-structure. The user can choose the time-frame, the variable (sex, immigration, emigration and saldo), the number of classes, type of classification, colors and some more options. The mapping itself is invoked by a mouse-click event in either one of the regions in Switzerland, or an European country. Additionally one can display diagrams, dynamically generated histograms and tabular views. An animation function shows the development from year to year over a longer period.

Example 11: Migration between Europe and Switzerland:
http://www.karto.ethz.ch/teaching/dip_2001_joray/

Tax burden in Zurich, Schwyz and Zug

The last example is a student's work of Raphael Hilber and Stefan Ziegler. It shows the distribution of tax-burden within the cantons Zurich, Schwyz and Zug. A scaled pie-slice diagram is rendered on a separate canvas invoked by mouse-over events in order to show the amount of taxes collected in an administrative unit and the distribution between legal and natural persons. Besides the functionality already described above (choroplethe mapping), one can search for and display specific units (choosing between municipalities, districts and cantons), or select by attribute data. The search may be restricted in an further step to a subset already selected in a first step. Due to time-restrictions the students could only implement this prototype for Internet-Explorer.

Example 12: Tax burden in the cantons Zurich, Schwyz and Zug:
http://www.karto.ethz.ch/teaching/vtb_2001_hilber_ziegler/

SVG Workflows

As SVG is an open and well-documented file-format, there are several ways to export or generate SVG-code. Sometimes a combination of different methods works best:

Export from Graphics software

At the time of writing this article (May 2001) Adobe Illustrator, Corel Draw, Jasc WebDraw, Killustrator, Mayura Draw, Sketch and some more products or open-source projects supported the import and/or export of SVG graphics. Because

SVG is a well documented format and supports all of the major possible graphical features, it will replace or complement todays established vector formats for graphics-exchange. Quark, Adobe, Sun (StarOffice) and others announced that they will support SVG throughout their products.

Export from GIS-Software, Internet Map Servers

GE-Smallworld and SICAD already recognized the value of SVG for web-mapping and integrated SVG as an export-filter or as part of their Map-Servers. More GIS-vendors are likely to follow. Open-source efforts are on the way to provide export filters from Arcview, Arc/Info and Mapinfo. For some GIS it is necessary to write self-made converters - as long as one has to deal with text-based formats this should not be too complicated.

SVG printer drivers

There are printer drivers (similar to Postscript or PDF) available to directly print from any application to SVG. The drawback of this approach is that those printer-drivers usually only support the graphical features of postscript, what means that a lot of the original file-structure (e.g. layers, groups and graphical primitives) is usually lost.

XML to SVG-conversion with XSL

XSL as a formatting and transformation language can provide rules how another XML-file should be rendered as SVG. XSL allows for control-structures, loops, simple queries, sorting, etc. as processing instructions during the conversion process. This process can be either done on the server-side, prior to delivery to the client, by using XSLT-Processors like Saxon [5] or Xalan [3], or by using client-side formatting and/or conversion. XSLT also supports conversion to text-based non-XML formats. The latter is currently only partly supported by a few browsers.

SVG-Converters

Converters exist to transform from and to documented graphics formats, such as eps/postscript, pdf, flash, cgm and wmf. An up-to-date list for available converters may be found at [10].

Serverside SVG-Generators, Databases

These are usually SVG-graphics libraries to generate SVG on the fly. An SVG-Perl library from the University of Nottingham exists (see [8]) and the Apache organization is working on a server-based SVG-Generator written in Java. The project is called Batik (see [2]) and includes a SVG-Viewer, rasterizer and generator, all based on the Java2D libraries.

Text-Editors and XML-Editors

To do corrections and additions it is quite handy to use Text-Editors that support SVG/XML with syntax highlighting and/or syntax check, or XML-Editors with SVG-support. This is a big advantage compared to binary-only formats.

Specific Cartographic Problems

Of course, SVG is not a "cure-all" for all (carto-)graphic problems. One of the biggest problems, that will likely prevent some cartographers from using SVG, is, that the source-code cannot be protected. This is not a SVG-specific problem but hits also other documented vector-based internet-file-formats, such as Macromedia Flash. As long as SVG-files are derived from more complex databases this should not be too much of a problem. This circumstance needs a rethinking for some cartographers. It is not only the graphical data, that makes a geodatabase valuable, but also its permanent maintenance and a complete attribute-database. More and more customers demand complete web-solutions and not only bare cartographic data. So how can one prevent copyright infringements? The vector-based nature of SVG also has its advantages: one can f.e. add hidden features or specific, non-visible, noise, that can serve as evidence to reveal copyright-infringements. It is thus possible to prove the origin of the data through mathematical methods.

One further disadvantage is the current situation of installed SVG-Plugins: Compared to Flash, the number of installed plugins is small. This is likely to change, because the Adobe SVG-plugin is currently bundled with different graphics software, incl. Acrobat-Reader. Several browser vendors or open-source projects are in the state of developing native SVG-support within their browser, so there won't be a need for installing additional plugins. Finally, there is the problem of missing topology and limitation in graphical symbolization: complex line-styles have to be reassembled from two or more lines (like most of the cartographers are used to from desktop-mapping). This drawback will perhaps be resolved in further SVG-versions. Topology is not available, unless one writes software to calculate it on the fly.

Conclusion

Up to now, all attempts to deliver high-quality cartography to the web, more or less failed, often due to technical restrictions. Most cartographers at this particular front have been busy creating emergency workarounds, preventing them from doing their actual job. For the first time, SVG, the new Internet vector standard, reduces this strain. It opens ways for cartographers to concentrate on content delivery and interactions, still typical for monitor cartography. But, with SVG, not only the visualization is optimized - SVG is an open, object-oriented file-format (not a software!). It is completely based on the XML-model. This guarantees a solid and long-term embedding into web-environments and compatibility to other data-formats, also in non-networking environments. Thanks to the object-hierarchy it is possible to implement all possible interactions. SVG-based projects can also be used offline - the Adobe SVG-viewer and the open-source Batik-project [2] offer programming interfaces and therefore embedding in offline multimedia environments. The combination of the high-quality rendering SVG-Viewers provide for both display and printing, and the architecture based on XML, allows a reduced development time, because developers can deliver for different media types based on the same data-source. The open character of the Internet, along with the success of the open source model, allows for new venues and possibilities for cartography, which are of no small significance for other fields as well, like geodata processing and sales/promotion.

References

[1]: Adobe.Com, "SVG resources at Adobe.com", http://www.adobe.com/svg/, 2001.

[2]: Apache.Org, "Batik - an open-source SVG viewer, rasterizer and generator", http://xml.apache.org/batik/, 2001.

[3]: Apache.Org, "Xalan - an open-source XSLT processor", http://xml.apache.org/xalan/, 2001.

[4]: Gaborit, Gaëtan, "La cartographie dynamique sur le Net avec SVG", http://svgmap.free.fr/, 2000.

[5]: Kay, Michael, "Saxon - an open-source XSLT Processor", http://users.iclway.co.uk/mhkay/saxon/, 2001.

[6]: Neumann, Andreas and Andréas M. Winter, "SVG on Carto.Net", http://www.carto.net/papers/svg/, 2001.

[7]: Peterson, Michael, "Trends in Internet Map Use", in: Proceedings of the 19th ICA Conference, Ottawa, 1, 10 p., 2001.

[8]: University of Nottingham, "SVG.PL - a perl library to generate SVG on the Fly", http://cs.nott.ac.uk/~jxm/SVG/netscape.html, 2001.

[9]: W3.Org, "The World-Wide-Web Consortium", http://www.w3.org/, 2001.

[10]: W3.Org, Chris Lilley, "SVG Converters", http://www.w3.org/Graphics/SVG/SVG-Implementations, 2001.

[11]: W3.Org, Chris Lilley, "SVG Resources and Specification", http://www.w3.org/Graphics/SVG/Overview.htm8, 2001.