

Using SVG for Interactive Maps

Leon Stringer
(leon.stringer@ntlworld.com)

26 April 2006

Abstract

Geographical mapping systems are an obvious application of the SVG open vector graphics standard. Potentially heterogeneous client systems can render cartographical information as appropriate and client-side scripting allows the possibility of direct user interaction with the information in the same way we expect from Web pages today. However, implementation of this standard has been slow and its use restricted to niche applications. This paper examines the technology and its use for interactive mapping applications to examine the status of adoption so far and future directions.

Contents

1. Introduction.....	4
2. Technologies.....	4
2.1 SVG.....	4
2.1.1 Example.....	5
2.2 Vector and raster maps.....	6
3. Literature Review.....	7
3.1 “Time for SVG – Towards High Quality Interactive Web-maps”.....	7
3.2 “SVG & Smart Maps”.....	7
3.3 “Adaptive Methods for Mobile Cartography”.....	8
4. Investigation.....	9
4.1 SVG-capable clients.....	9
4.1.1 Mozilla Firefox.....	9
4.1.2 Adobe SVG Viewer.....	9
4.1.3 Amaya.....	9
4.1.4 Opera.....	10
4.2 Interactive web maps.....	10
4.2.1 Google Maps.....	10
4.2.2 Loreda City Council Map.....	10
5. Analysis.....	11
6. Conclusion.....	11
7. Bibliography.....	12

1. Introduction

This paper looks at the application of the Structured Vector Graphics (SVG) standard markup language in the field of interactive mapping systems such as might be used via the Internet. SVG's aim was to bring the same kind of heterogeneous access to vector graphics as HTML did to text and raster images, and an immediately obvious application of this is for rendering map data. But in the five years since its introduction, has it lived up to this expectation?

By way of historical context a survey of past work pertaining to SVG interactive maps is undertaken. This is followed by an examination of the current status of client technology and two interactive Internet mapping systems are described. This information is then analysed to attempt to explain the current degree of usage and likely future trends.

In investigating this report, the author hopes to gain increased knowledge of SVG and its applications, discovery of less exploited capabilities of current browsers and the general direction interactive web services are heading.

2. Technologies

2.1 SVG

SVG (Ferraiolo, Jun & Jackson 2003) is a textual markup language for describing 2D vector graphics. The first version of the standard, 1.0, was introduced in 2001 by the World Wide Web Consortium (W3C), the same body responsible for the HTML and XML standards. The current version, 1.1, was introduced in 2003.

SVG is an XML application: valid SVG files are valid XML files using a specific syntax defined by an XML Document Type Definition (DTD). The language contains ways to draw vector objects (lines, polygons), raster images and text in various colours and styles on a specified canvas area. These objects can be grouped for reuse in an image and there is also a mechanism for animation within diagrams. As well as the SVG elements and attributes, the standard also specifies CSS (Cascading Style Sheets) style properties and a DOM (Document Object Model) interface allowing an SVG document to be controlled programmatically.

After the initial publication of SVG 1.0, there was interest from industry in the applicability to mobile devices. However, the features available in SVG were considerably in excess of the requirements anticipated for mobile platforms. Consequently, the publication of SVG 1.1 "modularised" the language by collecting related features into units of functionality. This modularisation then led to the Mobile SVG profiles: SVG Tiny for very restricted devices such as cellphones and SVG Basic for more capable devices such as PDAs. SVG Basic is a subset of SVG 1.1; SVG Tiny is a subset of SVG Basic.

SVG can be used in a variety of scenarios, e.g. a common format for sharing vector data between applications or within applications requiring vector graphics via reusable SVG-aware libraries. However, as with the majority of the W3C's work, the key target is serving information via computer networks to suitably capable user agent software (e.g. browsers).

2.1.1 Example

By way of an example, listings 1, 2 and 3 contain the XHTML, SVG and ECMAScript¹ source for a simple “map” containing a road and a river; a single interactive function is included to hide or show the river.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="content-type"
        content="text/html; charset=iso-8859-1" />
  <title>Map Demo</title>
  <script type="text/javascript" src="./map.js">
  </script>
</head>

<body>
<h1>Map</h1>

<p><object id="mapobj" type="image/svg+xml" data="map.svg"
          width="200" height="200">
  SVG Map</object></p>

<p><input id="checkbox" type="checkbox" checked="checked"
          value="Show waterways" onclick="showWaterways()" />
Show waterways</p>
</body>
</html>
```

Listing 1: map.html

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
    "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="200px" height="200px" xmlns="http://www.w3.org/2000/svg"
      xmlns:xlink="http://www.w3.org/1999/xlink">
<path id="river" d="M0 60 C 50 55, 150 125, 200 110"
      style="stroke: blue; fill: none; stroke-width: 5px" />
<text x="80" y="70">B901 Main Road</text>
<path id="b901" d="M 0 100 L 80 100 L 120 80 L 200 80"
      style="stroke: black; fill: none; stroke-width: 3px" />
</svg>
```

Listing 2: map.svg

```
function showWaterways() {
  var svgObject = document.getElementById('mapobj');

  if (svgObject && svgObject.contentDocument)
    // All other browsers
    svgObject = svgObject.contentDocument;
  else
    // Internet Explorer + Adobe SVG Viewer
    svgObject = svgObject.getSVGDocument();

  var waterway = svgObject.getElementById('river');

  if (document.getElementById('checkbox').checked)
    waterway.setAttribute('visibility', 'inherit');
  else
    waterway.setAttribute('visibility', 'hidden');
}
```

Listing 3: map.js

¹ ECMAScript is sometimes referred to by its pre-standards names of “JavaScript” or “JScript”.

Figure 1 shows the results in the Firefox browser version 1.5.0.2. (This code has also been tested with Opera 8.52 and Internet Explorer 6 with Adobe SVG Viewer 3.03).

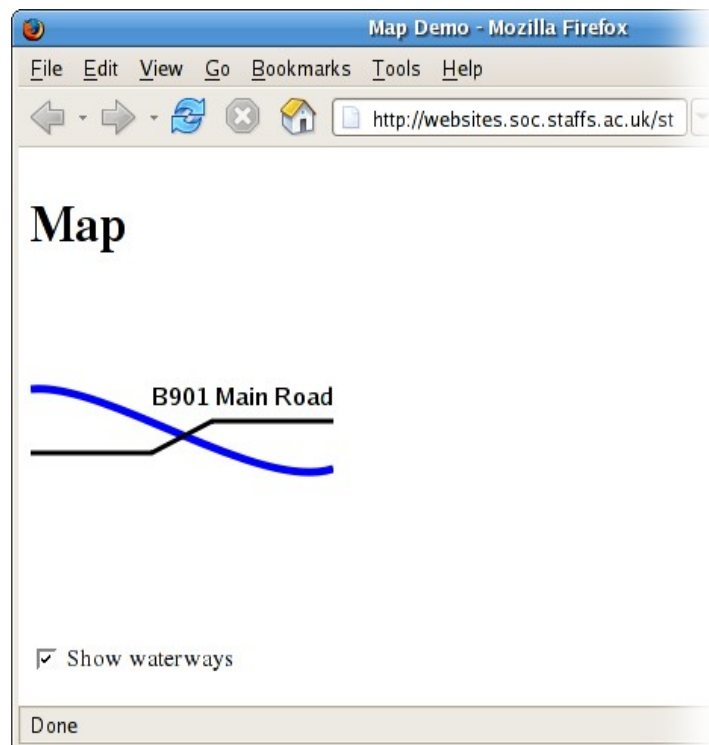


Figure 1: Map example

2.2 Vector and raster maps

Maps can be rendered from a GIS data source in one of two ways: raster or vector. Raster maps use pre-rendered images of the cartographical information – e.g. roads, lakes, place names – typically divided into tiles to decompose the map into manageable chunks for display. Vector maps retain cartographical features by rendering these as discrete objects: points (e.g. landmarks), lines (e.g. roads, rivers) and polygons (e.g. jurisdictional boundaries).

Raster maps tend to result in a more high quality display than vector maps but are constrained to those levels of detail where corresponding tiles exist. The tiles are typically transmitted over the network in a common format (e.g. PNG or GIF) and displayed directly by the client. Vector maps, on the other hand, support any variation in zoom level (within the range of the GIS data source) and any cartographical features can easily be hidden or shown as appropriate for the level of detail or user requirements. Point data for the required area is streamed over the network and rendered by the client using appropriate vector graphic primitive operations.

Either technique may be appropriate depending on the application. The structure of the GIS database may also affect the feasibility of using one or other method. For more advanced applications the two can be combined, e.g. by overlaying vector information on a raster image. In the browser-based scenarios we are interested in, it is apparent that any user agent capable of displaying bitmap raster graphics (e.g. compressed in PNG or GIF formats) is capable of displaying a raster map. There has been no equivalent uptake in standards or implementations for any vector graphics system between browser implementations.

3. Literature Review

This section reviews published papers pertaining to the field.

3.1 “Time for SVG – Towards High Quality Interactive Web-maps”

In their paper, Neuman & Winter (2001) highlight SVG as the key technology enabling “smart map” systems. As well as describing the SVG standard, the paper emphasises the complementary roles other client and server technologies have to play. On the client side, ECMAScript's ability to manipulate SVG through the DOM enables interactivity and event handling. On the server side, the presence of geographic spatial datatypes in modern relational database systems (e.g. Oracle, PostgreSQL) combined with common Web servers and scripting engines (e.g. Apache with PHP) are described as “perfect complements” to SVG.

A survey of alternative vector graphic technologies suitable for Web systems is presented, of which only SVG, Flash and PDF enjoy any significant use today. The fact that SVG is an open format that can be implemented by anyone without royalty payments or potential patent infringement is highlighted as an advantage over competing formats, and an overview of SVG's capabilities are presented. Listed are examples of cartographic uses of SVG and other areas where SVG is in use (e.g. as a common format between structured drawing applications).

SVG source code is available to end users in the same way that HTML source is available to users of Web pages. One interesting disadvantage described is that purveyors of cartographical data may not be happy for their geographical information to be available in this way, presumably due to the possibility of copyright infringement. It could be argued that the open nature of source on the Web has been a driver for adoption of HTML; techniques are quickly discovered and employed leading to greater competition. And despite this, copyright law still applies, actual data, trademarks and other copyrightable aspects are protected. Maybe GIS suppliers need to bring their business models up-to-date, open their systems to their users and start competing.

Lack of client support is the second disadvantage identified. When this paper was written, the only way to access SVG data with mainstream browsers was through a browser plug-in, the then relatively small install base for such plug-ins being a significant barrier to the uptake of SVG. The authors anticipated the improvement of this situation with the necessary plug-ins being bundled with other software, and the future native support of SVG in browsers negating the requirement for a plug-in altogether.

The focus on the open standards of SVG and attention to the importance of open source implementations are convincing arguments for the use of SVG to produce interactive, feature-rich, cross-platform smart maps. However, in the intervening years, emerging general purpose mapping services have eschewed SVG. The following contributory factors may be in some way responsible:

1. The persistently low install base of SVG plug-ins.
2. The subsequent lack of support for SVG in web browsers. Two browsers have recently included native support for SVG (Firefox 1.5 and Opera 8.5, both released in 2005) although so far these implementations are comparatively limited.
3. The desire of providers of cartographical data to keep their data in a proprietary form.

This paper is an excellent description of the promise of SVG in cartographical applications. The lack of progress in the intervening period is disappointing, though understandable.

3.2 “SVG & Smart Maps”

Bugg (2003) presents a number of techniques for building dynamic smart maps. The focus is on the client side, with attention on SVG, the DOM and ECMAScript.

Full source code for an example smart map is provided showing animation of objects (vehicles), pop-up

descriptions of locations (e.g. business premises), status indication through colour (e.g. showing whether a shop is open or closed) and the ability to hide or show map details (e.g. waterways, railways, etc.).

The article then goes on to discuss how dynamic updates from the server might be handled. It suggests using Java servlets, although whether this is managed by the client polling the servlet for information or a servlet-updated map being fetched from the server periodically depends on the requirements. At the time of the article's writing the only way to render and manipulate SVG with ECMAScript was using Adobe's SVG Viewer plug-in. The example smart map does not function with clients released subsequently.

This article uses a simple example to illustrate the potential of SVG in the GIS arena. One can imagine many applications using these techniques, and clearly this is only scratching the surface of the technologies. However, once again, in the intervening three years, little similar has materialized. Presumably, once again the lack of client support must be the major contributory factor.

3.3 “Adaptive Methods for Mobile Cartography”

Reichenbacher (2003) analyses the requirements for mobile users of GIS data, primarily Location-Based Services. A discussion of potential scenarios and usability issues leads to an attempt to formally categorise some of the HCI factors. The architecture of an entire GIS with mobile clients is described and the adaptation aspects of map visualisation are discussed. Finally client screenshots from a prototype SVG mapping system are presented.

The author applies the HCI concept of Activity Theory to user actions in the mobile scenarios. The contextual elements for the scenario are identified and functions with these elements as parameters used to derive patterns used in designing the system. Although this cannot capture and model all the required information for a given scenario it is sufficient such that missing information can be inferred.

The technical architecture describes a GIS system for mobile users using standard interfaces (e.g. SOAP) for potentially distributed nodes. Given the typically restricted capabilities of mobile devices, the computational load is biased towards the server. Contextual parameters are shown as annotations on the architecture diagram.

The adaptation section starts by defining adaptability as the extent to which system characteristics can be altered either by the user or automatically by the system in response to the user's needs. Aside from standard UI configuration such as font settings, this includes map-specific options such as zoom or level of detail. Usability analysis may simplify adaptation by identifying typical user roles thus allowing adaptation features to be grouped into profiles of user roles. Automatic adaptation is also discussed, such as potential threshold ranges for automatic changes in zoom and panning when in motion, based in the current speed of the device.

The prototype implementation presents map data from a GIS source passed to a PDA as SVG and rendered accordingly. Two different techniques are used for rendering location information. The first is a conventional use of multiple overlaid layers of information from a relational database. The second uses the database for the “base” information (e.g. streets) but takes points of interest from a GML (Geographical Markup Language, another XML application) source, included in the map through XSL (XML's Extensible Stylesheet Language) transformation to SVG. Although not explicitly stated, one can assume from the platform featured that the SVG Tiny profile is being used.

This paper offers an interesting insight into the issues and possibilities for SVG mobile maps. In the introduction the paper mentions in-car navigation devices as being costly and restricted in use. SVG allows similar functionality on relatively low cost mobile devices. Use of an open standard also allows multiple data sources to be combined in order to overlay information of specific interest to the user. Automatic adaptation can further improve the user experience; the importance of empirical usability research into adaptation in this area is highlighted by the author.

4. Investigation

4.1 SVG-capable clients

4.1.1 Mozilla Firefox

Mozilla Firefox (<http://www.mozilla.org/projects/firefox/>) is a Free/open source software Web browser available for Microsoft Windows, Apple Macintosh and GNU/Linux. The current version, 1.5, released in November 2005 introduced native SVG support. There are however two key drawbacks to this implementation.

1. Firstly, it is neither a complete implementation of SVG 1.1 (the current recommendation) nor any of the official subsets (e.g. Tiny SVG).
2. Secondly, SVG cannot be embedded directly within another document, e.g. a map within an XHTML page (although a similar result can be achieved by use of the `<object>` element).

The details of the current level of SVG support are extensively documented (Mozilla Foundation 2006) with respect to the W3C specification and other known issues and limitations (e.g. problems specific to a particular platform) are similarly explained. Script access to the DOM is also provided.

Full implementation of SVG 1.1 is a long way off and the browser's next release – 2.0 expected early 2007 – merely adds support for the `textPath` element. However, the work undertaken and that planned for the future shows a clear commitment to SVG which may be a cornerstone for an increase in its application.

4.1.2 Adobe SVG Viewer

Adobe's SVG Viewer (<http://www.adobe.com/svg/viewer/install/>) is a browser plug-in that interprets and displays SVG content seamlessly within the browser window. Supporting the majority of SVG 1.1, version 3.0 was released in 2001 and there have been only minor releases subsequently to address security issues. It is proprietary software, and versions for Microsoft Windows, Apple Macintosh, GNU/Linux and Solaris available for download free of charge. Non-Windows versions are less fully featured, notably lacking a way for scripts in HTML pages to access the SVG DOM.

The vast majority of SVG systems encountered researching this paper – and all of the cartographic systems – worked with SVG Viewer only. This was due to two factors: the comprehensiveness of the implementation and the scripting functionality available through the DOM. The level of support for SVG elements and the DOM interface are extensively documented (Adobe 2001). Consequently it would appear that the apparent lack of development in the past five years is due to a general level of satisfaction with the viewer. As indicated by the security fixes it is still actively supported.

SVG Viewer is the most complete implementation for SVG in a browser-based environment currently available and the majority of Web-based SVG systems use this as the reference implementation.

4.1.3 Amaya

Amaya (<http://www.w3.org/Amaya/>) is a web browser and editor supporting XHTML, MathML and SVG. Intended as a usable reference implementation for their standards, it is a Free/open source software project actively developed under the auspices of the W3C. It is available for Microsoft Windows, Apple Macintosh and UNIX-like systems.

A subset of SVG is supported concentrating on the core functionality of lines, shapes, text and images, and this can only be edited at source level. It also supports the direct embedding of SVG in XHTML (or vice versa). Although it does support SVG animation, there is no script engine.

Amaya's primary users are those interested in web development rather than end users. Those using SVG

for web systems will find it useful in testing against a standard implementation. However, the lack of a script engine limits its use for interactive systems.

4.1.4 Opera

The Opera desktop web browser (<http://www.opera.com/>) introduced support for SVG Tiny with the release of version 8.0 in 2005. Subsequent versions have built on this and the recently released beta test version (9.5b) added support for a superset of SVG Basic as part of the intended eventual implementation of SVG 1.1. As with Firefox and the Adobe SVG Viewer the level of support is documented in detail (Opera n.d.). Like Firefox, Opera does support the embedding of SVG content in HTML pages via the `<object>` element and, as a browser intended for mainstream use, includes a script engine.

Opera is proprietary software available for download free of charge and supports Microsoft Windows, Apple Macintosh, UNIX-like systems and others. Opera also produce a microbrowser for use on mobile devices. Although this doesn't have SVG support it does provide this through a third-party plug-in on the Symbian platform (Opera 2003).

4.2 Interactive web maps

4.2.1 Google Maps

Introduced in 2005, Google Maps (<http://maps.google.co.uk/>) is a Web-based service offering mapping information for many countries. This is combined with local business search information, route planning and even an interface for use by third party applications, and provides one of the most comprehensive general-purpose Internet GIS systems today. It also includes detailed terrestrial satellite imagery with the option of combining this with an overlaid map.

The maps are displayed using tiled PNG images and, although this is a raster-based rendering on the client, clearly are generated by a vector-based system. The background of these tiles is a transparent layer allowing this to be overlaid on the JPEG satellite photoimages to present the hybrid view.

Finding locations is simply a case of entering their name as the search criterion (e.g. "stafford") and then panning and zooming as required. Searching for local business information is achieved by specifying information such as "hospitals in stafford". Results are presented on the map using labels (A, B, C, etc.) relating to a key on the left-hand side containing the details for each "hit". As above, these labels are overlaid on the map by appropriately placing PNG images with transparent backgrounds.

Route information can be found via queries such as "stafford to stoke-on-trent". Again, details of the route are shown to the left of the map, and again the information overlaid on the map. This time however, there is an interesting dichotomy in the implementation of the overlay. With browsers other than Internet Explorer, the server generates a transparent PNG with a vector of the route which is then overlaid on the map. With Internet Explorer the vector is generated using VML. VML (Vector Markup Language, Microsoft 1998) is an early attempt by Microsoft and others to introduce a standard for 2D vector graphics. It was submitted to the W3C in 1998 (Matthews *et al.* 1998) and Microsoft incorporated it into version 5.0 of their browser. VML was ultimately rejected by the W3C but parts were used for SVG and the same fate befell PGML (Al-Shamma *et al.* 1998) submitted the same year. Internet Explorer support for VML remains in modern versions, and despite its failure in the standards process, Internet Explorer's market share clearly made it a conveniently widespread mechanism for overlaying vectors generated on the fly.

4.2.2 Loredo City Council Map

The Loredo City, Texas council web site includes a page containing an interactive SVG map of the city (http://www.laredotexas.gov/Maps/svg_map/City_Council_Map.htm). This includes roads, railways, schools and electoral boundaries all of which can be hidden or shown as desired. As one would expect from an interactive map one can zoom and pan as required. Points of interest (e.g. schools) can be clicked

on for further information.

This shows the kind of interactive map SVG can be used to produce. Information is clearly presented and easily accessible. Unfortunately, the only way to access all the features is with Internet Explorer and the Adobe SVG Plug-in restricting the potential audience.

5. Analysis

The combination of an open standard language for 2D vector graphics with a standard client API mechanism to control generated graphics appears to be an ideal platform for developers of interactive maps. This should simplify the development process by leaving client rendering to the user agent. But user agent implementation is a key problem.

For the past five years, the only significant browser platform for SVG has been Internet Explorer with the Adobe SVG Viewer plug-in (“IE+ASV”). Unfortunately, this has led to some compatibility issues now that other browsers with native SVG capabilities are arriving; indeed all interactive maps examined in preparing this report only worked correctly when used in the IE+ASV configuration. The reasons for this centre around differences in the way ECMAScript manipulation is achieved. Even the simple map given as an example in this paper requires additional code to overcome these differences. The majority of applications encountered used the archaic, non-standard but widely supported `<embed>` element to include SVG content instead of the standard HTML `<object>` element. Five years ago this may have been the simplest way to achieve this, but today it only causes problems.

Another barrier to adoption is the lack of clients that are SVG ready. Microsoft's Internet Explorer enjoys the largest user base at the moment but, despite its popularity as an SVG platform via the ASV plug-in, does not support SVG directly. There is no sign that this will change in the future, the upcoming version 7.0 (now available in beta test form) still lacks this. Given their historically “assertive” attitude to when and how they will use standards, it is tempting to think Microsoft may have an agenda. Have they decided that SVG support can be delegated to third party plug-ins? Or, having proposed and implemented a 2D vector graphics markup language in their browser already, is there some resentment towards SVG. It may well be the case that due to Google's ubiquity in web services, their Maps system's adoption of VML means that usage of this actually prevails over SVG in the interactive map arena.

The commitment by competing browsers to SVG combined with their steadily increasing user base may make this an increasingly attractive target platform as the level of implementation continues to increase. Will this competition compel Microsoft to introduce native SVG support to Internet Explorer? The presence of (inactive) SVG commands in a Google Maps ECMAScript file hints at a significant boost for SVG coming soon (Schiller 2006).

Mobile devices however are a different story. The SVG Mobile profiles have allowed manufacturers and developers to include lightweight SVG support for various hardware types. This is coupled with increasing interest in the possibilities of applications for nomadic users, such as location-based services, as a source of revenue for mobile communications companies. This may well be SVG's “killer app”, Iviko (2005) asserts that “Mobile SVG is perfectly positioned as an industry standard, ensuring compatibility across handset manufacturers, mobile operators, and service providers and allowing the momentum and size of the 2D wave to build.”

6. Conclusion

In this paper, SVG and related technology has been discussed with respect to interactive maps. Some work in the area as been reviewed and the current state of client technology assessed. Two smart map systems have been looked at, one using SVG and one not. This has led to a consideration as to the factors that have so far impeded the use of SVG and influences and avenues that have emerged as promising ways to overcome this.

There is little doubt that SVG adoption for interactive map systems has so far been disappointing. There are relatively few examples of its use and these are generally restricted to

niche areas. Where it has been used, usage has been restricted to isolated platforms. On the desktop, only now is there the potential for change with the broadening of support in browser systems. However, mobile systems are only just becoming capable of SVG usage and in this sector there is a great deal of interest in this as a platform to build applications and as a source of revenue. SVG may yet emerge as the dominant technology for interactive maps.

7. Bibliography

- Adobe, 2001, Current support for SVG, retrieved from the Internet: <http://www.adobe.com/svg/indepth/pdfs/CurrentSupport.pdf>, April 23, 2006
- Al-Shamma, N., R. Ayers, R. Cohn, J. Ferraiolo, M. Newell, R.K. de Bry, K. McCluskey & J. Evans, 1998, Precision Graphics Markup Language (PGML), retrieved from the Internet: <http://www.w3.org/TR/1998/NOTE-PGML-19980410>, April 25, 2006
- Bugg, K., 2003, SVG & Smart Maps, *Dr. Dobbs Journal* 28(3), ISSN 1-44-789X, 38-41
- Ferraiolo, J., F. Jun & D. Jackson (Eds), 2003, Scalable Vector Graphics (SVG) 1.1 specification, retrieved from the Internet: <http://www.w3.org/TR/SVG11/>, April 25, 2006
- Iviko, 2005, Mobile SVG: The graphics wave of the future, retrieved from the Internet: http://www.ikivo.com/pdf/mobile_svg_wp.pdf, April 25, 2006
- Matthews, B., D. Lee, B. Dister, J. Bowler, H. Cooperstein, A. Jindal, T. Nguyen, P. Wu & T. Sandal, 1998, Vector Markup Language (VML), retrieved from the Internet: <http://www.w3.org/TR/1998/NOTE-VML-19980513>, April 25, 2006
- Microsoft, 1998, Introduction to Vector Markup Language (VML), retrieved from the Internet: <http://msdn.microsoft.com/workshop/author/vml/>, April 25, 2006
- Mozilla Foundation, 2006, SVG in Firefox 1.5, retrieved from the Internet: http://developer.mozilla.org/en/docs/SVG_in_Firefox_1.5, April 15, 2006
- Neuman, A. & A.M. Winter, 2001, Time for SVG – Towards high quality interactive web-maps, in *Proceedings of the 20th International Cartographic Congress*, Beijing, 2001
- Opera, n.d., SVG support in Opera 9, retrieved from the Internet: <http://www.opera.com/docs/specs/opera9/svg/>, April 23, 2006
- Opera, 2003, Opera introduces brand new features in double release, retrieved from the Internet: <http://www.opera.com/pressreleases/en/2003/04/24/>, April 23, 2006
- Reichenbacher, T., 2003, Adaptive methods for mobile cartography, In *Proceedings of the 21st International Cartographic Conference (ICC)*, Durban, South Africa, ISBN 0-958-46093-0
- Schiller, J., 2006, Google Maps v2 to use SVG, retrieved from the Internet: <http://blog.codedread.com/archives/2006/02/10/google-maps-v2-to-use-svg/>, April 25, 2006