INTRODUCTION OF SVG AS A DATA INTERCHANGE FORMAT FOR ARCHITECTURAL DOCUMENTATIONS

Günter Pomaska
Faculty of Architecture and Civil Engineering, Bielefeld University of Applied Sciences, Germany
gp@imagefact.de

CIPA Working Group VII - Photography

Keywords: Architecture, Internet / Web, modelling, architectural heritage conservation, vector graphics

Abstract:

Web tools provide software for generating data formats and interactive viewing. Since the bandwidth of the Web is still limited, not by technology but accessibility of hardware, several companies introduced their own formats. Targeting to small file formats and speed.

One can observe that since 1997 3D environments are published on the Web. Numerous software packages were released to the market. Lots of them escaped soon without reaching importance. Due to the proprietary formats and the limited bandwidth of the Internet connections most of them became failures. VRML, virtual modelling language, was introduced to create and model virtual environments. This standard, defined by the Web 3D consortium, is more used for off-line presentations instead of distributing 3D worlds via the Internet.

SVG (scalable vector graphics) introduced in 2001 is limited to 2D and should be the answer of the Web community to Macromedia's Flash. SVG is defined by the World Wide Web Consortium (W3C) for 2D vector graphics. The success of SVG lies in its standardisation and unification of this XML-based language. With SVG interactive viewing and animation of graphic content is possible. A visitor of a Web site can zoom through maps and drawings, containing a huge amount of data. The quality of the presentation is high and even loss-free despite very large-scale factors.

Users of SVG need conversion tools for transforming existing data to that format. Viewers are available as stand-alone software or plug-ins for Internet browsers. Designing Web pages with SVG requires some knowledge of the XML concept behind. This contribution will discuss the usefulness of SVG for architectural drawings and some kinds of thematic mapping. First experiences with façade drawings of Prussian classicism are presented.

1 FLASH OR SVG?

SVG (scalable vector graphics) is the World Wide Web Consortium (W3C) standard definition for interactive 2D Web graphics. This format includes text, bitmaps, vector graphics, animation and interactivity - everything from one source. SVG formats can be generated with simple text editors, XML tools or converters. SVG files scan be viewed with stand-alone viewers or browser plug-ins.

Flash is a proprietary format for animations from Macromedia. The format far spreads and is supported by Macromedias powerful animation tool. Despite this software there are several good reasons to apply SVG.

SVG can be used without sophisticated authoring software; a simple text editor is sufficient. Easy access via script languages is provided. The format is very well structured; the source is readable as a text format, while Flash is embedded in binary formats.

Referencing of external images enables easy updating of SVG presentations. Flash takes advantage from embedded images. But updating means recompiling. SVG references images and supports a much wider colour space and colour management.

Flash animations are replayed as a series of single frames. Every image is precalculated and stored in the animation. SVG has extended features by applying interpolations between two positions. This results in a much better structured and shorter code.

The Web community provides resources like tutorials, scripts, software and application samples. Not least SVG benefits from the advantages of a non-propriety standard.

2 RASTER AGAINST VECTOR DATA

A raster file represents a matrix with colour intensity values. Zooming in or out is provided by extrapolation or interpolation of neighboured pixels. Loss of quality and information is accepted.

Vector data are defined by co-ordinates (points) and path segments like lines, curves (bezier, splines) and others. The connection between two points is a path segment. A closed path segment defines a surface. Two points can be connected via infinite paths to each other. The shortest way is the straight line. Other ways need appropriate parameter definitions. Paths build the mathematical shape to which the visual attributes like colour, fill mode or line thickness are applied.

The quality of appearance of a vector-based graphic is independent from its zoom factor. The amount of memory is reduced compared to raster graphics. Therefore less transfer time

is needed. Figure 1 demonstrates this difference between raster and vector graphics. Left is a raster format small and enlarged. Right is a vector format; the enlargement does not loose image quality while the file size is the same. The sample is taken from the document www.w3c.org/tr/SVG-access.

Figure 1: Comparison of raster and vector graphic

3 CONVERTING AND VIEWING

Adobe and Apache provide common viewers for SVG. Adobe's SVG viewer (release 3.0 at the time of writing) is provided as a plug-in for Web-browsers or as a stand-alone version.

Downloading from www.adobe.com/svg leads to a quickly and automatically installation.

Clicking the right mouse button in a SVG window opens the context menu with access to the functions zoom-in, zoom-out, panning, help and some others. There is no learning curve existing.

The Batik SVG browser from Apache can be downloaded as a standalone program from xml.apache.org/batik/svgviewer.html

For CAD files (DWG, DXF) and bitmaps (BMP, WMF) are converting tools existing. I downloaded SVG Factory, ACME CAD Converter and CAD2SVG for test purposes from the Internet.

SVG Factory is freeware. It converts BMP / WMF files to SVG. Optimisation can be selected as a transfer option. With this limited functionality and the required conversion from DWG/DXF to WMF or BMP it seems to not be sufficient for CAD applications. But for bitmap images it is useful and results in sufficient conversions. The Bitmap shown in figure 3 was reduced from 2200 kByte from its original to 332 kByte as the resulted SVG file. The interface of SVG Factory is very simple but easy to access. The software can be downloaded from www.svgfactory.com.

ACME CAD Converter is equipped with a professional CAD interface. An imported CAD file (DWG/DXF) can be viewed in 3D mode. 3D data is supported to a certain extend. The user interface supplies a submenu for the layer table. Layers can be switched on/off as well as line type converting. All the results shown here are converted without any problems. Some better structuring of the SVG file could be provided. Using the layer tables and some final editing solve those slightly inconvenience. The license fee is in the range of shareware. For downloading visit www.freefirestudio.com.

Another converter is cad2svg from Savage Software. This converter is unfortunately not able to handle all 3D elements. Only wire frame information will be converted. Nevertheless CAD2SVG is a professional tool including features one cannot find in other software. The home page can be visited under www.savagesoftware.com.

All tested converters are writing to SVG and SVGZ the compressed file format.

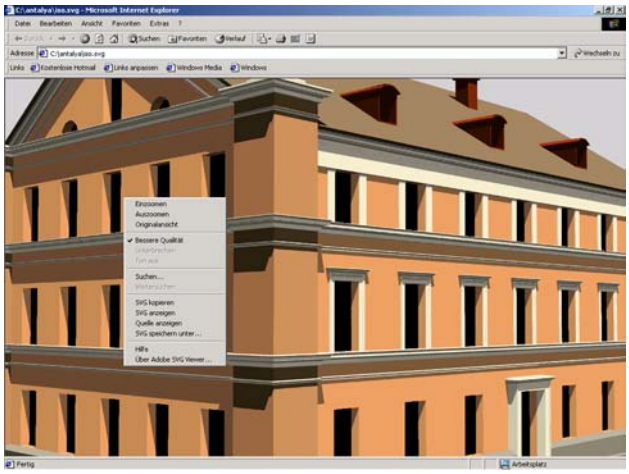

Figure 2: Converting bitmap to SVG with SVG Factory. The result is displayed in a browser window with Adobe SVG viewer plug-in including the context menu

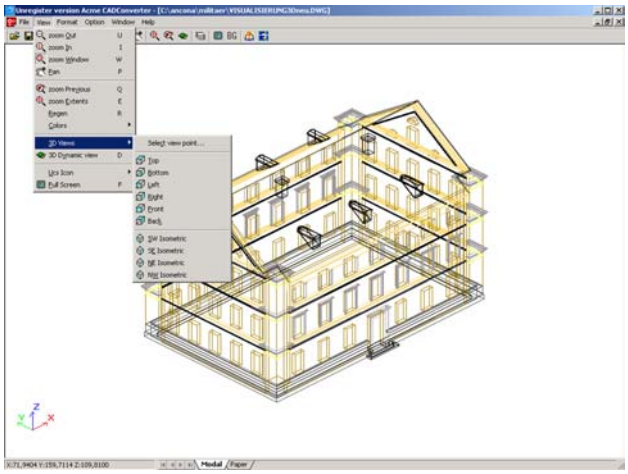

Figure 3: Converting with ACME CAD converter. The 3D object is transformed into an isometric projection by applying the 3D view menu.

4 SVG DATA STRUCTURE

The following content partly explains the SVG data structure and tags with respect to representation of architectural drawings or maps that should be distributed via the Internet. The data source is a DWG or DXF file converted with a conversion tool into SVG. The focus here is on structuring the graphic, setting attributes and linking to other references.

The SVG specification is separated into graphic elements, container elements, graphic referencing elements and text content elements.

The file starts with the XML prologue as parsing information:

```
<?xml version="1.0" standalone="0">
```

The root tag for a SVG document is **<svg>.** The attributes **width** and **height** define the visible area for the document starting in the lower left corner of the frame. The attributes **viewbox** and **preserveAspectRatio** define the area of the graphic scaled to the width and height values. The concept is well known from basic graphic programming libraries. Identification of name space for xml is referenced by the **xmlns** attribute.

```
<svg xmls=http://www.w3.org/2000/svg width
= "400" height="300" viewBox="0.0 0.0
1024.0 764.0"
preserveAspectRatio="xMinYMax meet">
```

The following tags are structure elements:

<**g**>	grouping of graphic elements
<**defs**>	for definitions referenced later in the code
<**desc**>	element description
<**title**>	labelling elements
<**symbol**>	grouping references

Continuing the above example needs the following statements:

```
<title>South Façade of Building</title>
<desc>Converted from DWG to</desc>
<g>
<!-- the graphic content is embedded here -
->
</g>
</svg>
```

The tags <**title**> and <**desc**> are used for descriptions and are ignored by the render engine. Tag <**g**> is for grouping graphic elements. Transformations or attributes defined for a group will affect all the child elements of this group. The SVG graphic format can be embedded in HTML or XML document for example or can be addressed as an independent file.

4.1 Graphic Tags

The graphic content consists of predefined primitives like rectangle <**rect**>, circles <**circle**>, ellipses <**ellipse**>, lines <**line**>, polylines <**polyline**> or polygons <**polygon**>. Appearance of the objects is defined by style-attributes like fill or stroke, which can be set by style sheets or as particular shape properties.

Arbitrary lines are constructed from the definition given in the <**path**> tag. The notation is similar to Postscript. A list of points starts with the letter d followed by a string including the co-ordinates with a letter in front. The letter indicates the characteristic. M stands for move to, L for line to, C stands for curve and so on. A z at the end of the list indicates a closed path.

The co-ordinate pairs are separated by blanks. The following code shows a path with two straight lines rendered in black colour.

```
<path d="M499.93 340.11L499.91 340.06
499.88 340.00 "
fill="none" stroke="black" stroke-
width="0.2"/>
```

4.2 Interactivity

SVG provides sophisticated facilities for animation and interactivity. Here we focus on interactivity by hyperlinks and demonstrate a mouse over event for opacity of an image.

A link sets the connection between two documents while an anchor points to a reference in the same document. SVG treats Xlink as an own namespace. This namespace is referenced with xmlns:link.

The elements providing a link are covered with the tag

```
<a xlink.href="document.svg"
   target="_blank">
   <g> graphic elements </g>
</a>
```

Where document.svg is a sourcefile providing code like HTML, VRML or SVG. Target points to the target window where the new document has to be rendered.

Code that is not included in the XML or SVG specification like JavaScript or CSS, must be terminated between

```
<![CDATA[
   <!- - script code or CSS code -- >
]]>
```

The code `image { opacity: 0.5; }` defines a value for opacity of an image.

Following is the complete definition for the initial opacity style of an SVG document:

```
<defs>
  <style type="text/css">
    <![CDATA[
      image {
       opacity: 0.5;
      }
    ]]>
  </style>
</defs>
```

A URL can reference images in the SVG image tag like in HTML. The image tag imports the file at the position x, y. The size is defined by the parameters width and height.

```
<image xlink:href="wagenhaus.gif" x="580"
y="380" width="64" height="32">

<set attributeName="opacity"
attriuteType="CSS" to="1.0
begin="mouseover"/>

<set attributeName="opacity"
attriuteType="CSS" to="0.2"
begin="mouseout"/>

</image>
```

Furthermore there are two events announced. In one case the opacity is set to 1.0 in the other case the value is set to 0.1. The events will be initialised while touching the image with the mouse (mouseover) or while leaving the image (mouseout).

This example is used in the following mapping application for displaying views of buildings in transparency or full colour while touched by the pointing device.

5 MAPPING APPLICATION

Building an interactive VR world in an immersive environment is projected in co-operation between Preussen-Museum, Minden and University of Applied Sciences, Bielefeld. Some studies of access to VRML models and interactivity are already discussed.

Here a suggestion is submitted to implement an interactive map for investigating the past and the future of the area named Simeonsplatz. The sample can be visited under www.divide-by-zero.com/svg/simeonsplatz.svg. This file is in a very early state and for the time being not optimised. But hints to possibilities for final implementation can be read from this example. A screenshot is displayed in figure 4.

The simple file structure as an extract from the complete code is demonstrated here. The optimised file with referenced tree symbols by the <use> tag and <text> tags will be around 90 kByte.

```
<?xml version="1.0" encoding="UTF-8"?>
<svg width="1200.0" height="830.232"
     viewBox="75.0 50.0 1032.000 714.000">
<title>Simeonsplatz in Minden</title>
<desc>CADConverter + Editor</desc>

<defs>
<!-- definitions -->
</defs>

<g id="legend">
   <text x="480" y="93"
    text-anchor="middle"
    font-size="16" font-family="Verdana"
      fill="black"
      stroke="black" stroke-width="0.1">
      Simeonsplatz in Minden</text>
   <rect x="87" y="62" width="24"
    height="8"
    fill="rgb(128,128,128)" stroke="black"
    stroke-width="0.2"/>
<!-- other graphics for the legend -->
</g>

<g id="surfaces">
  <path d="M227.63 364.08L231.96 369.64
  <!-- graphics for surfaces -->
  fill="rgb(216,192,177)"
  stroke="rgb(216,192,177)"
  stroke-width="0.2"/>
</g>

<desc>image reference to Schwichow-
Denkmal</desc>
<image xlink:href="schwichow_denkmal.gif"
   x="400"
   y="120" width="32" height="32">
  <set attributeName="opacity"
   attributeType="CSS"
   to="1.0" begin="mouseover"/>
  <set attributeName="opacity"
   attributeType="CSS"
   to="0.2" begin="mouseout"/>
</image>

<g id="wagenhaus">
   <a xlink:href="wagenhaus.wrl"
      target="_blank">
      <! graphic embedded in a link -->
   </a>
</g>

<g id="buildings">
   <!-- other buildings, not of particula
interest -->
</g>
<g id="trees">
   <!-- the trees, should be referenced -->
</g>
<g id="text">
   <!-- text for the drawing -->
</g>

<!-- Please note: SVG renders   -->
<!-- like the painter algorithm -->
<!-- The element coming in last -->
<!-- is visible in front of     -->
<!-- others behind.             -->

</svg>
```

Figure 4: An interactive mapping application

The map carries a legend, displays topography and buildings. Links to some VRML models and references other documents. While moving the mouse over images with very high transparency, the image occurs in full colour. Those images are renderings of the CAD models provided for interactive walk-through and telling about the history. The immersive environment installed for test purposes is for the time being a multi projection on three screens driven by one computer equipped with a PNY Quattro graphic board.

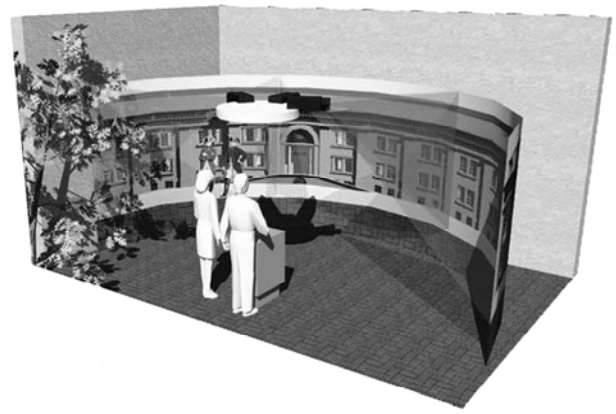

Figure 5: VR model of the multi projection environment

6 Conclusion

SVG with its interactivity and possibility for importing data from different sources is introduced as an interface for accessing a database of a historic site. The basic tags for designing and giving structure to a SVG document have been mentioned. Some conversion tools available from the Internet have been successfully tested. The future work for this project will be: bringing together the historic data and projected buildings, adding more interactivity and animations to generate an interactive map with a high degree of information.

It is obvious that SVG has the potential to provide a solution for the above-mentioned requests.

References

References from books:

Fibinger, Iris
SVG Scalable Vector Graphics, Markt + Technik Verlag, Muenchen 2002

Günter Pomaska
Internetpräsentation von Bauprojekten, Bauwerk-Verlag Berlin 2002

References from other literature:

Günter Pomaska
Implementation of Web 3D Tools for Creating Interactive Walkthrough Environments from Building Documentations,
ISPRS WG V/4 and IC WG III International Workshop on Vision Techniques for Digital Architectural and Archaeological Archives, Ancona, Italy, July 2003

Mine Hamamcıoglu Turan
Discussion of two Photogrammetric Techniques Combined for Documentation of Defensionskaserne in Minden with Reference to Architectural Heritage conservation,
CIPA 2003 XIXth International Symposium, New Perspectives to Save Cultural Heritage, Antalya, Turkey, Sept. 2003

References from the Internet http://

SVG viewer :
www.adobe.com
Specifications:
www.web3d.org/tr/svg
SVG – Learning by Coding
www.datenverdrahten.de
Application of SVG providing the mapping sample from figure 4:
www.divide-by-zero.com/svg/simeonsplatz.svg
About the project Simeonsplatz:
www.imagework.de/kaserne.html
Other references from the Web are given in the text