SCALABLE VECTOR GRAPHICS (SVG)

Chengyuan Peng
Department of Computer Science and Engineering
Helsinki University of Technology
pcy@tml.hut.fi

ABSTRACT

A new XML technology, called Scalable Vector Graphics (SVG) will bring the fast and high-resolution vector graphics that we currently enjoy in the print world to the Web. SVG overcomes the shortcomings of existing pixel-based web graphics and contains more sophisticated features, such as scripting, gradients, animation, filter effects, and interactivity. It'll integrate seamlessly with traditional Web standards, such as HTML, GIF, and CGI, complement with dynamic and interactive capabilities to forge the future of Web graphics. This presentation aims at giving an overview of technical principles defined in W3c without stepping into the details of SVG tags since the special SVG features are achieved by implementing the principles.

1 INTRODUCTION

Whether you are on the Web to do personal shopping, information searching, or business-to-business activities, high-quality graphics are critical to clear and effective communications. SVG will move design beyond the bare boxes and frames of today's web [10]. Complex layouts, sophisticated typography, and high-resolution artwork are both possible on the web.

The SVG format is a new XML grammar for defining scalable vector-based 2D graphics for the Web and other applications and usable as an XML Namespace [15]. SVG is a text-only collection of XML commands, similar to the PostScript language. This means that SVG files can be edited in simple text editors and can be generated easily from server-side tools such as CGI and Perl [7].

The SVG as an open standard is currently a working draft at the W3C. Its group members come from key industry leaders, Adobe Systems, AOL/Netscape, Apple, Autodesk, Canon, Corel, CSIRO, Eastman Kodak, Excosoft, Hewlett-Packard, IBM, ILOG, IntraNet Systems, Macromedia, Microsoft, OASIS, Opera, Oxford Brookes University, Quark, Sun Microsystems, and Xerox [1]. The W3C has issued SVG as a candidate recommendation [1]. And now SVG is well on the way to becoming the vector graphics language of choice for the Web [13].

Two most commonly used graphics formats on the Web (GIF and JPEG) today are pixel-based. Pixel graphics lose quality when zooming [13]. In addition, rendering pixel graphics for the Web needs extra work. Each pixel must contain all the information needed to display an image. They're big, slow, and dumb.

SVG, on the other hand, overcomes the pixel graphics' limitations. It is entirely based on XML language for describing 2D graphics via vector graphics, text and raster graphics [9]. And because it's vector-based, SVG graphics have the same high quality whether they're displayed on laptops, handhelds, high-end monitors, TVs, or even printed on paper without the "staircase" effects you see when printing bitmapped images [11]. Graphical formats can be defined by CSS. It is stylable with CSS and can transform with XSLT [6].

SVG files are much smaller and more compressible than comparable JPEG or GIF images [13]. Since SVG graphics take up less space than existing GIFs and JPEGs, they help optimize browser's performance.

In addition to faster download speeds and high-resolution printing, SVG also has some other nice features, such as high-performance zooming, panning inside of graphics without reloading, animation, filter, kerning, masking, scripting, and

linking [9]. Users can dig deep into an SVG object to obtain more dynamic information [8].

Since SVG is based on XML and entirely text-base, it allows search engines to index SVG graphics and users to search for text within them [9].

The SVG works with Java technology. The SVG complements Java's high-end graphics engine (i.e. the Java 2D API) [2]. Any parser that can read XML can also read SVG. Integration of SVG into existing DOM means that SVG elements can be controlled and modified by the usual JavaScript/Java interfaces [2].

Since SVG is based on an emerging open format developed by W3C and its member companies, such as Adobe, Microsoft, and Netscape, there will be no need for the designers and developers to learn new software or abandon existing tools and technologies [10]. They can seamlessly integrate XML-based SVG with HTML, CSS, JavaScript, CGI, and other Web technologies. They can continue to use software they already employ [9]. Thus, interactive and dynamic effects are possible on both HTML and SVG using the same set of scripts [10]. Web designers and developers will not need to learn new software or abandon existing tools and technologies in order to incorporate SVG into their designs [10].

2 SPECIAL SVG FEATURES

SVG has all the features of normal graphics, but its basic primitive is the Graphic Object, not a point or line [5]. SVG allows for three types of graphic objects: vector graphic shapes (e.g., paths consisting of straight lines and curves), images and text [5]. Graphical objects can be grouped, styled, transformed and composed into previously rendered objects. The feature set includes nested transformations, clipping paths, alpha masks, filter effects and template objects.

2.1 Rendering Model

SVG uses a "painters model" for rendering [6]. Paint is applied in successive operations to the output device such that each operation paints over some area of the output device. When the area overlaps a previously painted area, the new paint partially or completely obscures the old. When the paint is not completely opaque, the result on the output device is defined by the mathematical rules for composing (i.e. simple alpha blending).

SVG supports the following built-in types of paint which can be used in fill and stroke operations: Solid color, Gradients (linear and radial), and Patterns. SVG allows any painting operation to be filtered.

Elements in an SVG document fragment have an implicit drawing order. The first elements in the SVG document fragment getting "painted" first. Subsequent elements are painted on top of previously painted elements. SVG document fragments can be semi-opaque. In many environments (e.g., Web browsers), the SVG document fragment has a final compositing step where the document as a whole is blended translucently into the background canvas.

Grouping elements have the effect of producing a temporary separate canvas onto which child elements are painted. Upon the completion of the group, the effect is as if the group's canvas is painted onto the ancestors canvas using the standard rendering rules for individual graphic objects.

2.2 Basic data types

The common data types for SVG's properties and attributes are Angle, color, coordinate, frequency, integer, length, list of xxx, number, paint, percentage, time, transform-list, and URI.

2.3 Styling

The advantages of style sheets are presentational control, flexibility, faster download and improved maintenance [16]. SVG combines scripting, DOM and CSS (Dynamic HTML) and is widely used for animation, interactivity and presentational effects.

SVG uses styling properties to describe many of its document parameters. Styling properties define how the graphics elements in the SVG content are to be rendered. SVG shares many of its styling properties with CSS and XSL. Except for any additional SVG-specific rules, The following properties are shared between CSS2 and SVG, i.e., font, text, and other properties for visual media (most of these properties are also defined in XSL) [6].

XSLT offers the ability to take a stream of arbitrary XML content as input, apply potentially complex transformations, and then generate SVG content as output. XSLT can be used to transform

XML data extracted from databases into an SVG graphical representation of that data.

CSS is a widely implemented declarative language for assigning styling properties to XML content, including SVG. It represents a combination of features, simplicity and compactness that makes it very suitable for many applications of SVG [6][16].

XSL style sheets define how to transform XML content into something else, usually other XML. When XSLT is used in conjunction with SVG, sometimes SVG content will serve as both input and output for XSL style sheets. Other times, XSL style sheets will take non-SVG content as input and generate SVG content as output [6].

SVG content can also be used as an exchange format (style sheet language-independent).

2.4 Coordinate systems, Transformations and units

For all media (text, graphics, etc.), the SVG canvas describes the space where the SVG content is rendered. The canvas is infinite for each dimension of the space, but rendering occurs relative to a finite rectangular region of the canvas. This finite rectangular region is called the SVG *viewport*. For visual media, the SVG *viewport* is the viewing area where the user sees the SVG content [6].

The size of the SVG *viewport* is determined by a negotiation process between the SVG document fragment and its parent. An initial *viewport* coordinate system and an initial user coordinate system are identical. Both coordinates systems are established such that the origin matches the origin of the *viewport*, and one unit in the initial coordinate system equals one pixel in the *viewport*. The *viewport* coordinate system is also called *viewport* space and the user coordinate system is also called user space.

At any point in an SVG drawing, a new *viewport* can be established in which all contained graphics are drawn. After establishing a new *viewport*, a new *viewport* coordinate system, a new user coordinate system, the unit identifiers of the coordinate, and a new clipping path are also implicitly established [6].

A new user space (i.e., a new current coordinate system) can be established at any place within an SVG document fragment by specifying transformation matrices, such as rotation, skewing, scaling, and translation. Establishing new user spaces via coordinate system transformations are fundamental operations to 2D graphics and represent the usual method of controlling the size, position, rotation and skew of graphic objects. The transformmations have effects on all user space coordinates and lengths on the given element and all of its ancestors. Transformations can be nested to any level. The effect of nested transformations is to post-multiply (i.e., concatenate) the subsequent transformation matrices onto previously defined transformations.

The supported length unit identifiers are: em, ex, px, pt, pc, cm, mm, in, and percentages. All coordinates and lengths in SVG can be specified with or without a unit identifier [6].

2.5 Path

Paths represent the geometry of the outline of an object. Compound paths (i.e., a path with subpaths are possible to allow effects such as "donut holes" in objects.

In addition to line path, there are also cubic Bézier, quadratic Bézier, and elliptical arc path.

2.6 Basic shapes

SVG contains the six basic objects, i.e., rectangle (rectangle, including optional rounded corners), circle, ellipse, polyline, polygon, and line [6]. These basic shapes, along with paths, constitute the graphic shapes of SVG. The most important element is the path object and it is the most flexible one. The open or closed line objects and polygons can be drawn using path element.

The basic shapes can be stroked, filled, and used as clip paths. Lines may carry the following attributes: thickness, line end type, vertex markers, broken line mode, broken line offset, etc.

You can draw anything you want with the <path> element. When you have your drawing ready, you can add text to it by using the text element. This could be useful for buttons, organizational charts, or maps [5].

2.7 Text

SVG has powerful text capabilities. SVG has the following text features: font specification, text

orientation and direction, text alignment, and rich text formatting.

Text can be rendered as a single line or laid out on a complex path). A block of text can be separated into several text spans and different properties can be applied to each of them. We can easily manipulate the text layout in the direction, alignment and relative position.

There is no automatic line breaking or word wrapping in SVG. In order to realizing line breaking and layout arrangement within a block of text, you have to use the <tspan> element.

SVG allows for any parameters in generating text elements, such as font family, font type, size, position, width, direction, inclination, etc. In order to properly represent non-Latin or self-made fonts, SVG allows for external font description files, or for replacement of missing Unicode characters by supplementary graphics or symbols. Entire fonts may also be embedded into an SVG file [4].

SVG supports for Unicode, Latin, East Asian, semantic character sets such as English, Spanish, Japanese, Hebrew and Arabic [6].

2.8 Painting: Filling, Stroking, and Marker Symbols

Path elements, text elements and basic shapes can be filled which means painting the interior of the object and stroked which means painting along the outline of the object [6].

Some elements, such as path, polyline, polygon, and line elements can also have marker symbols drawn at their vertices (cf. 2.11). With SVG, you can paint (i.e., fill or stroke) with a single color, a gradient (linear or radial), or a pattern (vector or image, possibly tiled), and custom paints available via extensibility (cf. 2.19).

SVG uses the general notion of a **paint server** [6]. Gradients and patterns are just specific types of paint servers. Paint servers are specified using a URI reference.

The SVG user agent performs color interpolations and compositing in the sRGB color space or in a (light energy linear) linearized RGB color space when rendering gradients, performing color animations, performing alpha compositing of graphics elements into the current background, or performing various filter effects.

2.9 Color

SVG has 16 million colors. Because I uses reliable color model, SVG prints with the same colors you see on your display and at full printer resolution. Filling, lines or texts can take on a transparency value.

All SVG colors are specified in the sRGB color space. Additionally, SVG content can specify an alternate color specification using an ICC (International Color Consortium) profile. If ICC-based colors are provided and the SVG user agent supports ICC color, then the ICC-based color takes precedence over the sRGB color specification [6].

The ICC has established a standard, the ICC Profile, for documenting the color characteristics of input and output devices. Using these profiles, it is possible to build a transform and correct visual data for viewing on different devices.

2.10 Gradients and Patterns

Gradients and Patterns are used for filling and stroking.

Gradients consist of continuously smooth color transitions along a vector from one color to another, possibly followed by additional transitions along the same vector to other colors. SVG provides two types of gradients, i.e., linear gradients and radial gradients [6].

A pattern is used to fill or stroke an object using a pre-defined graphic object which can be tiled at fixed intervals in *x* and *y* to cover the areas to be painted [6].

2.11 Clipping, Masking and Compositing

SVG allows any painting operation to be limited to a sub-region of the output device by clipping and masking. SVG supports only simple alpha blending (composing) [6].

Clipping paths uses any combination of path, text and basic shapes to serve as the outline of a 1-bit mask, where everything on the "inside" of the outline is allowed to show through but everything on the outside is masked out.

Masks are container elements which can contain graphics elements or other container elements

which define a set of graphics that is to be used as a semi-transparent mask for compositing foreground objects into the current background [6].

One key distinction between a *clipping path* and a *mask* is that *clipping paths* are hard masks (i.e., the silhouette consists of either fully opaque pixels or fully transparent pixels, with the possible exception of antialiasing along the edge of the silhouette) whereas *masks* consist of an image where each pixel value indicates the degree of transparency vs. opacity. In a mask, each pixel value can range from fully transparent to fully opaque.

2.12 Filter effects

A filter effect consists of a series of graphics operations that are applied to a given source graphic to produce a modified graphical result. The result of the filter effect is rendered to the target device instead of the original source graphic [6].

So far SVG supports sixteen filter primitives: Blend, ColorMatrix, ComponentTransfer, Composite, ConvolveMatrix, DiffuseLighting, DiffuseLighting, DisplacementMap, Flood, GaussianBlur, Image, Merge, Morphology, Offset, SpecularLighting, Tile, Turbulence.

A shadow is an example of a filter effect. By defining various filters, different types of shadow effects can be created for three types of graphic objects: text, shapes, and raster images.

Blend composites two objects together using commonly used imaging software blending modes. It performs a pixel-wise combination of two input images [6].

ColorMatrix applies a matrix transformation on the RGBA color and alpha values of every pixel on the input graphics [6].

ComponentTransfer performs component-wise remapping of data for every pixel. It allows operations like brightness adjustment, contrast adjustment, color balance or threshold [6].

Composite performs the combination of the two input images pixel-wise in image space using one of the Porter-Duff composing operations: *over, in, atop, out, xor*. Additionally, a component-wise *arithmetic* operation (with the result clamped between [0..1]) can be applied [6].

ConvolveMatrix applies a matrix convolution filter effect. A convolution combines pixels in the input image with neighboring pixels to produce a resulting image. A wide variety of imaging operations can be achieved through convolutions, including blurring, edge detection, sharpening, embossing and beveling [6].

DiffuseLighting lights an image using the alpha channel as a bump map. The resulting image is an RGBA opaque image based on the light color with alpha = 1.0 everywhere. The lighting calculation follows the standard diffuse component of the Phong lighting model. The resulting image depends on the light color, light position and surface geometry of the input bump map. The light map produced by this filter can be combined with a texture image using the multiply term of the *arithmetic* composing method. Multiple light sources can be simulated by adding several of these light maps together before applying it to the texture image [6].

Flood creates an image with infinite extent filled with the color and opacity values [6].

GaussianBlur performs a Gaussian blur on the input image [6].

Image refers to a graphic external to this filter element, which is loaded or rendered into an RGBA raster and becomes the result of the filter primitive. This filter can refer to an external image or can be a reference to another piece of SVG. It produces an image similar to the built-in image source except that the graphic comes from an external source [6].

Merge composites input image layers on top of each other using the *over* operator with *Input1* on the bottom and the last specified input, *InputN*, on top. Many effects produce a number of intermediate layers in order to create the final output image [6].

The canonical implementation of **Merge** is to render the entire effect into one RGBA layer, and then render the resulting layer on the output device. In certain cases (in particular if the output device itself is a continuous tone device), and since merging is associative, it might be a sufficient approximation to evaluate the effect one layer at a time and render each layer individually onto the output device bottom to top.

Morphology performs fattening or thinning of artwork. It is particularly useful for fattening or thinning an alpha channel [6].

Offset offsets the input image relative to its current position in the image space by the specified vector. This is important for effects like drop shadows.

SpecularLighting lights a source graphic using the alpha channel as a bump map. The resulting image is an RGBA image based on the light color. The lighting calculation follows the standard specular component of the Phong lighting model. The resulting image depends on the light color, light position and surface geometry of the input bump map. The result of the lighting calculation is added. The filter assumes that the viewer is at infinity in the z direction [6].

Tile creates an image with infinite extent by replicating the input image in image space [6].

Turbulence creates an image using the Perlin turbulence function. It allows the synthesis of artificial textures like clouds or marble. The resulting image will have maximal size in image space. It is possible to create bandwidth-limited noise by synthesizing only one octave [6].

2.13 Interactivity

SVG drawings can be interactive and dynamic. Moving or clicking the mouse over any scalable vector graphic generates immediate feedback, such as highlighting, text tips, and real-time changes to the surrounding HTML text. Animations can be triggered either declaratively (i.e., by embedding SVG animation elements in SVG content) or via scripting [9].

SVG supports for zoom, pan, scale on any portion of an SVG image and not see any degradation without waiting for extra graphic data to download.

SVG proposes a variety of user events. Three event categories are specified: mouse events, keyboard events, and state change events (concerning display and SVG file loading state) [6].

Some interactive display environments provide the ability to modify the appearance of the pointer (*cursor*). Three types of cursors are available in SVG: Standard built-in cursors, Platform-specific custom cursors, and Platform-independent custom cursors [6].

2.14 Linking

Because SVG content often represents a picture or drawing of something, a common need is to link into a particular view of the document.

XLink and XPointer allow for linking from within SVG files to other files on the Web. These files could be for other SVG files, SMIL presentations, or simple HTML pages.

Hyper links may be used to refer to other files, or to other elements within an SVG document.

2.15 Scripting

It is possible to specify the scripting language. Events can cause scripts to execute (cf. 2.13).

2.16 Animation

The Web is a dynamic medium. Thus there is no doubt that SVG supports for the animation. SVG content can be animated in the following ways [6]:

a) Using SVG's animation elements. The various elements can define motion paths, fade-in or fade-out effects, and objects that grow, shrink, spin or change color.

b) Using the SVG DOM. Every attribute and style sheet setting is accessible to scripting. SVG offers a set of additional DOM interfaces to support efficient animation via scripting. Therefore, any kind of animation can be achieved. The timer facilities in scripting languages such as ECMAScript can be used to start up and control the animations.

c) SVG has been designed to allow future versions of SMIL to use animated or static SVG content as media components.

d) It is expected that future versions of SMIL will be modularized and that components of it could be used in conjunction with SVG and other XML grammars to achieve animation effects.

SVG's animation elements were developed in collaboration with the W3C Synchronized Multimedia (SYMM) Working Group, developers of the Synchronized Multimedia Integration Language (SMIL) 1.0 Specification.

SVG offers interpolation of in-betweens. Interpolation options are: step-by-step, linear, or spline. Various parameters may be animated, such as color value, position, position along a path, rotation, scale, etc.

2.17 Fonts

One disadvantage of the today's Web font facility is that no particular font formats need to be supported. And so that different implementations support different Web font formats. Thus it is difficult for Web site creators to post a single Web site that is supported by a large percentage of installed browsers [10].

Reliable delivery of fonts is considered a critical requirement for SVG. SVG designers will be able to create SVG graphics with whatever fonts they use like in the print world even if the viewer hasn't purchased the fonts being displayed.

SVG utilizes CSS2's WebFont facility as a key mechanism for reliable delivery of font data to end users.

The purpose of SVG fonts is to allow for delivery of glyph outlines in display-only environments. SVG fonts that accompany Web pages must be supported only in browsing and viewing situations. Content creator will need to license the given font before editing the SVG file.

Because SVG fonts are expressed using SVG elements and attributes, in some cases the SVG font will take up more space than the font were expressed in a different Web font format.

The characteristics and attributes of SVG fonts correspond closely to the font characteristics and parameters described in the CSS2 specification. Unlike standard graphics in SVG, where the initial coordinate system has the y-axis pointing downward, the design grid for SVG fonts, along with the initial coordinate system for the glyphs, has the y-axis pointing upward for consistency with accepted industry practice for many popular font formats.

SVG fonts and their associated glyphs do not specify bounding box information.

An SVG font can be either embedded within the same document that uses the font or saved as part of an external resource.

2.18 Metadata

Metadata is information about a document. Metadata serves to deposit information on the document in a structured way. Data concerning authorship, date of publication, version, title, brief description, etc. are inserted. Those may be embedded into an SVG file, just as external code (e.g. JavaScript).

In the computing industry, there are ongoing standardization efforts towards metadata with the goal of promoting industry interoperability and efficiency. Content creators should track these developments and include appropriate metadata in their SVG content which conforms to these various metadata standards as they emerge.

The W3C Note "Metadata and SVG" [not yet published] discusses in detail various issues concerning metadata and SVG. The document provides a current set of recommendations about appropriate uses of metadata in conjunction with SVG. The W3C has ongoing metadata activities which provide general metadata guidelines. One of the W3C's metadata activities is the definition of Resource Description Framework (RDF), a W3C Recommendation for specifying metadata [6].

Individual industries or individual content creators are free to define their own metadata schema but are encouraged to follow existing metadata standards and use standard metadata schema wherever possible to promote interchange and interoperability. If a particular standard metadata schema does not meet your needs, then it is usually better to define an additional metadata schema in an existing framework such as RDF and to use custom metadata schema in combination with standard metadata schema, rather than totally ignore the standard schema [6].

2.19 Extensibility

Extensibility is one of the most important features of SVG. 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 in SVG.

SVG allows inclusion of elements from foreign namespaces anywhere with the SVG content. In addition, SVG allows inclusion of attributes from foreign namespaces on any SVG element. SVG's including foreign namespaces can be used for the following purposes: Application-specific

information so that authoring applications can include model-level data in the SVG content to serve their "roundtripping" purposes (i.e., the ability to write, then read a file without loss of higher-level information) and Supplemental data for extensibility [6].

One goal of SVG is to embed foreign objects. It provides a mechanism by which other XML language processors can render into an area within an SVG drawing, with those renderings subject to the various transformations and composing parameters that are currently active at a given point within the SVG content tree. One particular example of this is to provide a frame for XML content styled with CSS or XSL so that dynamically reflowing text (subject to SVG transformations and composing) could be inserted into the middle of some SVG content. Another example is inserting a MathML expression into an SVG drawing.

3 CONCLUSIONS

The SVG standard was/is developed and supported (mainly by their products) by all major graphics and software companies and organizations, that are web relevant: Adobe, Apple, AutoDesk, Bit-Flash, Corel, HP, IBM, ILOG, Inso, Kodak, Macromedia, Microsoft, Netscape, Oasis, Open Text, Oxford University, Quark, RAL, Sun-Microsystems, W3C und Xerox [6]. As for the future, this circumstance will guarantee a broad support regarding import and export filters as well as converter and viewer development. Since SVG is a very well documented and open XML standard, one may easily generate and convert out of one’s own scripts or programs. It may be expected that SVG will also be supported by GIS software companies and that SVG due to its application in web projects will attain significance as a general graphics exchange format. Next to Adobe Illustrator and PDF, SVG is one of the few well-documented ASCII graphics formats. As compared to the first two, it decidedly offers even more possibilities. For the representation of SVG documents, as for now the client still needs a plugin (by Adobe), which the user must install. In future versions of WWW browsers, however, SVG interpreters will be standard. Integration is also possible via CSIRO's Java applet [11].

Up to now, any serious attempt concerning Internet cartography was limited by narrow technical specifications, which made high quality cartography rather impossible. Most cartography 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 cartography to concentrate on contents, on interactions, still typical for monitor cartography. The open character of 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 data processing and sales/promotion.

Adobe, a big driver in the SVG working group, has already said that it plans to integrate SVG into its suite of graphics applications (Illustrator, Photoshop, and GoLive). So I doubt we'll be hand coding too many SVG files—one can only enter so many coordinates. Instead we'll probably be using updated versions of our current tools to create SVG images. Adobe also says it is working on browser plug-ins for Navigator and Explorer to display SVG files, as well as a micro-viewer for a range of clients—from handheld devices to desktop computers [11].

Both Netscape and Microsoft are also participating in the working group, so there's a good chance we'll see their spin on displaying SVG graphics as well [16].

4 REFERENCES

[1] http://www.w3.org/2000/08/svg-pressrelease.html

[2] http://www.sun.com/software/xml/developers/svg/java2d-api/

[3]. Introduction to JSP and SVG http://www.sun.com/software/xml/developers/svg/jsp/

[4] SVG Text http://www.sun.com/software/xml/developers/svg/text/;$sessionid$BNHA5GIAAAS5HAMTA1FU45Q

[5] SVG Shapes. http://www.sun.com/software/xml/developers/svg/shapes/;$sessionid$BNHA5GIAAAS5HAMTA1FU45Q

[6] http://www.w3.org/TR/SVG/

[7] http://www.oingo.com/topic/83/83779.html

[8] http://wdvl.com/Authoring/Languages/XML/SVG/

[9] Vincent Hardy. Scalable Vector Graphics (SVG): An Executive Summary.

http://www.sun.com/software/xml/developers/svg/
[10] Anthony Celeste, The Future of Web Design http://www.designer.com/focus/articles/web_future/web_future_print.htm
[11] http://www.adobe.com/svg/overview/overview.html
[12] http://wdvl.com/Authoring/Languages/XML/SVG/DoingIt/
[13] http://www.webdeveloper.com/design/design_svg_intro.html
[14] http://www.w3.org/Graphics/SVG/Overview.htm8
[15] http://www.carto.net/papers/svg/index_e.html
[16] http://tech.irt.org/articles/js176/