USE OF SVG AND ECMASCRIPT TECHNOLOGY FOR E-LEARNING PURPOSES

A. Neumann [a]

[a] Institute of Cartography, ETH Zurich, CH-8093 Zurich, Switzerland
neumann@karto.baug.ethz.ch

ISPRS Workshop Commissions VI/1 – VI/2
Tools and Techniques for E-Learning, Potsdam, Germany, June 1-3, 2005

KEY WORDS: Web Technologies, Content Development, Standardization, Future Trends

ABSTRACT:

SVG (Scalable Vector Graphics) is a XML based markup language used to describe and integrate vector graphics, raster graphics and text. SVG is developed by the W3C web consortium as an official web standard, with the support of major computer graphics and mobile phone companies, such as Adobe, Canon, Corel, IBM, Kodak, Nokia, Opera, Sun, etc. SVG Mobile was furthermore adopted by the 3GPP consortium as a part of the 3GPP mobile phone standard. SVG graphics can be animated and enriched with interactivity. Scripting languages and network interfaces help build interactive applications. SVG's rich visualization options and the support of interactivity make it a natural candidate for providing graphics and interactive examples in e-learning environments. The ability to access the SVG source code and have a glance "under the hood" to see how things are made, is especially useful for learning and sharing purposes. SVG also provides a fun way to introduce programming and illustrate the functionality of algorithms. Students are usually motivated if they can graphically visualize what they program. The paper first summarizes SVG's capabilities. The second part will discuss strengths and weaknesses of the SVG approach and explain why SVG is a useful technology for e-learning purposes. The following section describes usage scenarios and gives a number of e-learning examples. Domains will include mathematics, geometry, electronics, programming and GIS. Development tools and authoring systems will be mentioned. The last part will discuss and outline current developments regarding the upcoming SVG 1.2 version.

1. INTRODUCTION

1.1 What is SVG?

SVG (abbreviation for Scalable Vector Graphics) is a webstandard for describing two dimensional vector graphics. It integrates vector graphics, raster graphics and text. SVG graphics can be interactive and animated. Bindings for scripting languages and network interfaces enable developers to build rich interactive graphics applications. Like many other W3C webstandards, SVG is based on XML and can therefore benefit from other neighbor technologies in the XML world and the ever increasing number of XML development tools. The open nature of SVG and the rich visualization options make it an ideal technology for visualizing scientific results or explaining and communicating complex learning topics. Interactivity in SVG applications, if implemented correctly, can stimulate intellectual curiosity and be an invitation to explore complex phenomena.

1.2 Current Status

SVG 1.0 became a W3C recommendation in September 2001. The W3C (World Wide Web Consortium) combines and coordinates the interests of research institutes, companies and universities and develops interoperable technologies, such as specifications, guidelines or software, with the overall goal to make the information on the web more accessible to both humans and machines. SVG 1.1 is the current recommended SVG version and became a recommendation in January 2003. The main goal of SVG 1.1 was to introduce profiles for mobile devices: SVG Tiny for very constrained devices, such as mobile phones, and SVG Basic for stronger mobile devices, such as PDAs and Smartphones. Version 1.2 is currently under development, a major upgrade that will help SVG to be a better foundation for interactive applications. Improvements include multimedia (audio, video), streaming, text-wrapping support, better network interfaces, vector effects, better compositing and blending and a new binding language (sXBL) that will allow content developers to share SVG building blocks more easily. The new SVG 1.2 features will be discussed in more detail in the last part of this paper. The development of SVG specifications is an open process. Anyone can give feedback on the public mailing lists and can read the current status of the working drafts at the SVG W3C homepage (Lilley et al, 2005).

1.3 SVG viewer implementations

Although SVG has been existing as a W3C recommendation for a while, native SVG support in webbrowsers is still in its infancy. At the time of writing, SVG content is best deployed using the Adobe SVG plugin, available for all major platforms and browsers (Linux, MacOSX, Solaris, Windows) which can be downloaded from the Adobe SVG homepage (Adobe, 2005a) or for developers at the beta viewer download area (Adobe, 2005b). Adobe will release a new version of its plugin after the SVG 1.2 specification is finalized.

Fortunately, three major browser projects are currently introducing native SVG support. Mozilla SVG (Mozilla, 2005) already has a rather stable SVG implementation that supports scripting but no SMIL animation. Mozilla SVG will be enabled by default in the next major Mozilla/Firefox upgrade. Opera introduced SVG Tiny support in their latest brower (version 8) which supports animation but still lacks CSS and scripting support (Opera, 2005). The Unix KDE browser Konqueror already introduced SVG support last year and includes scripting and basic animation support (KDE, 2005). At the time of writing, it is not yet clear if the other browser projects (Apples Safari and Microsoft Internet Explorer) will follow the trend to native SVG support.

A very good standalone SVG viewer based on Java is the Apache Batik viewer, which consists of a viewer, a rasterizer and converter, a pretty printer, a serverside framework and a font-converter. The Batik SVG viewer can be embedded as a rendering component into other Java Applications. Batik can also convert SVG files to PDF format for printing. (Apache Batik, 2005) SVG mobile viewers are available from Bitflash, Zoomon and Nokia. The SVG mobile specifications are now also part of the 3GPP mobile phone standard.

2. SVG CAPABILITIES

2.1 Document Structure and Rendering Model

SVG documents are built upon a regular XML document tree, consisting primarily of a header, processing instructions, comments, XML elements and attributes. Elements that appear first in the document tree are rendered first, subsequent elements are drawn on top of the previous elements, taking into account opacity, blending, filters, clipping and masking. As in any other XML file, elements may have unique ids that can be used to reference other elements. Element instances (<use /> elements) can reference other elements and override their attributes. The <defs /> section serves as a repository. Elements in this section are not rendered, but may be referenced elsewhere in the file. Typically, the <defs /> section contains gradient and pattern definitions, symbols and markers. Elements may also be grouped, which is particularly useful when several elements form a logical group or share common attributes. Elements or groups may be temporarily hidden from the document tree, a mechanism that helps simulate map layers. Description and title elements help describe the content of elements, groups or files in a verbal form, a technique that helps make SVG files more accessible to disabled persons or search engines. Switch elements help conditionally process the document tree, e.g. based on the system language or available SVG features.

2.2 Coordinate Systems and Transformations

The origin of the coordinate system in the SVG canvas is in the upper left corner with the positive x-axis pointing right and the positive y-axis pointing downwards. This implicates that in order to represent map coordinate systems one either has to multiply the y-axis by a factor of -1 or has to transform all elements within the map group. The latter approach has disadvantages when using text labels within the map, as they would appear upside down. Supported units are em, ex, px, pt, pc, cm, mm, in and percentages. Of particular interest to cartography and GIS is the viewBox attribute that allows to establish a new coordinate system inside the existing coordinate system. This way one can have nested coordinate systems, e.g. a device oriented coordinate system in screen pixels in the SVG root element and one or more nested map oriented coordinate systems. Itt is therefore possible to introduce real world coordinates, such as meters or kilometers, which helps if one has to merge different data sources. Geographic coordinate systems are not directly supported. If one has to reproject data one has to do it in a GIS or spatial database system prior to conversion to SVG. However, it is possible to include metadata about the projection system.

2.3 Basic Geometry Elements

SVG knows the following basic shapes:

- Rectangle (<rect />)
- Circle (<circle />)
- Ellipse (<ellipse />)
- Line (<line />)
- Polyline (<polyline />)
- Polygon (<polygon />)
- Path (<path />)

The above listed geometry types are more or less self explaining. The most powerful and interesting geometry type is the <path /> element. Path elements can describe all other basic shapes. Path elements can contain quadratic and cubic spline curves and arc segments. Geometry can be described in either absolute or relative coordinates (relative in the sense that a subsequent coordinate is related to its previous coordinate pair). Path elements can contain holes, and several disjunct paths can be combined to one single path. Paths can be open or closed. The symbol element can contain any SVG code. A symbol can be made of basic shapes and may also contain animations.

2.4 Text and Fonts

Figure 1: Some of the Text options in SVG, Source: http://www.carto.net/papers/svg/samples/text.shtml, © André Winter

Text support in SVG is very sophisticated. Almost any text feature available in DTP or graphics software is also available in SVG. Like any other basic shape, text can also have fillings, strokes and can be clipped or masked or can serve as a clipping path. Individiual glyphs or groups of glyphs can be shifted or rotated and text can also be aligned on path elements. It is even possible to animate a text along a path. The <tspan /> element allows the attachment of common attributes to a group of glyphs. SVG fully supports internationalization, including Unicode support, left to right text, bidirectional text or text that

runs from top to bottom. Together with the <switch /> element one can also deploy multilingual content in one single SVG file. To ensure that the SVG file displays the correct font, one can include the fonts in the SVG file, using the SVG font format. Glyphs in SVG fonts can basically contain any geometry and they can also contain animations. The individiual glyphs usually contain SVG path geometry. SVG fonts also support kerning tables, although not all SVG renderers do. SVG fonts can be converted from other font formats (e.g. truetype, opentype and type1) using the Apache Batik project (Batik, 2005) or the Fontforge application.

One major problem with text in SVG up to version 1.1 was the missing text wrapping feature. Until now, one had to manually introduce linebreaks, using <tspan /> elements nested in the text-element. SVG 1.2 introduces this missing feature by supporting text wrapping in arbitrary shapes.

2.5 Filling, Stroking, Opacity

SVG elements can be filled with uniform color, linear and radial gradients and patterns. Pattern tile definitions can contain raster data, vector elements and animations and are repeatedly drawn to fill the polygon. Gradient parameters can be animated as well. As to stroking, one can define the color, stroke width, linecaps and linejoins, miter angles, dash arrays and dash offsets. Opacity can be separately defined for strokes, filling or both. Group opacity treats elements as a group as opposed to treating each group element individiually. Of particular interest to cartography and GIS are the markers. Markers are symbols that are placed at each vertex of a shape or path. One distinguishes start-, end- and mid-markers. Markers can be automatically oriented to adapt to the bisector angle or tangent vector of two adjacent line or curve segments. Markers can be used to attach arrows to shapes or to represent objects at vertices, such as poles along a power supply line.

2.6 Styling

There are alternative ways to style elements in SVG. One can use CSS styles (internal and external) as in XHTML, XML presentation attributes or XSLT. However, not all of them are implemented in every user agent. XML presentation attributes are supported by any SVG user agent, while CSS are only implemented in some (Adobe, Batik and Mozilla SVG) and XSLT support is still in its infancy. It is also possible to define media dependent styles. This is potentially useful to hide user interface elements when printing SVG graphics or to provide different styling options for handhelds or projection systems.

2.7 Filters

Figure 2: Filter example: combination of a gaussian blur, offset, specular lighting and composite filter, Source: http://www.w3.org/TR/SVG11/images/filters/filters01.svg © W3C consortium

Filter features are unique to SVG. They aren't currently present in competing formats, such as Flash or CGM. Filters can be attached to both raster and vector elements. Vector elements are rasterized during the rendering pipeline, hence there is an opportunity to include filters. Typical applications for filters are color corrections, brightness and contrast adaptions, blurring and sharpening, illumination filters, generation of drop shadows and halo effects, convolution filters, displacement and morphology filters, generating turbulence, etc. Filters may be combined in any order and the output of one filter may be piped to the input of the next filter. Every filter parameter can be animated which can lead to very interesting effects. Filters are very powerful visualization options, but may require a fair amount of computing power.

For examples on SVG filters have a look at the excellent examples and tutorials by Michel Hirtzler (Hirtzler, 2002; Hirtzler 2005a) and Kevin Lindsey (Lindsey, 2003).

2.8 Interactivity and Scripting

Interactivity and scripting are key parts when it comes to making SVG appealing for e-learning applications. SVG graphics are by default zoomable and pannable. Many SVG viewers support additional interactions, such as search for text, or start/pause animations. SVG supports hyperlinks and custom cursors.

Various event types enable script or SMIL operations to react to user or system events. Supported events are the following:

Status Events
- SVGLoad
- SVGUnload
- SVGAbort
- SVGError

Zoom and Scroll Events
- SVGResize
- SVGScroll
- SVGZoom

UI Events
- focusin
- focusout
- activate

Mouse Events
- click
- mousedown
- mouseup
- mouseover
- mousemove
- mouseout

Keyboard Events
- keydown
- keyup
- keypress

Animation Events
- beginEvent
- endEvent
- repeatEvent

Mutation Events
- DOMSubtreeModified
- DOMNodeInserted
- DOMNodeRemoved
- DOMNodeRemovedFromDocument
- DOMNodeInsertedIntoDocument
- DOMAttrModified
- DOMCharacterDataModified

Any of the events listed above can trigger either a script function or a SMIL interaction. Mutation events listen to changes within a particular node in the XML document tree. Currently, they aren't implemented in the Adobe SVG viewer version 3. SMIL is a declarative way of specifiying interactions or animations. SMIL constructs generally contain a trigger (either time based or event based), the target, the attribute to change, duration and interpolation parameters. SMIL can be regarded as a simple, declarative scripting language.

The other, more flexible, way of modifying SVG documents is to use a clientside scripting language. Scripts can either be embedded in the SVG files or referenced (external files). SVG defines a language independent API to access and manipulate the SVG DOM. The most widely used and implemented scripting language in conjunction with SVG is ECMAScript (the standardized version of Javascript). One reads or changes attributes, creates, moves or deletes elements and loops over the document tree as it is the case with any XML or XHTML document. (Neumann et al, 2005a) is a tutorial about manipulating SVG documents using ECMAScript and the DOM. SVG also provides network interfaces to directly talk to serverside applications. .getURL() and .postURL() are methods which allow transfering and retrieving of data from and to the server without having to reload the SVG file. SVG 1.2 will add more network options for client-server communication.

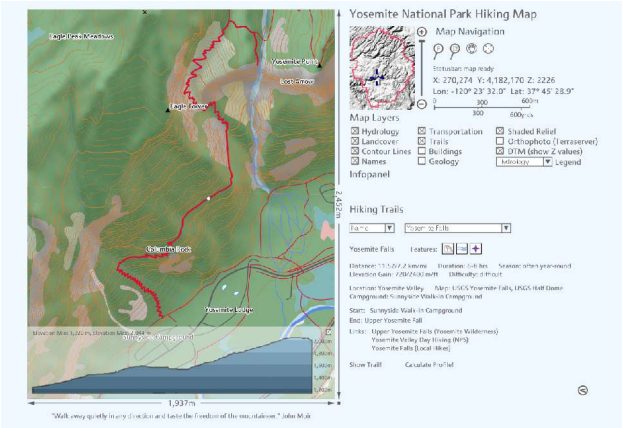

Figure 3: Yosemite National Park Hiking Map - Example of a highly interactive mapping application that makes extensive use of SMIL and scripting, Source: http://www.carto.net/williams/yosemite/, © Juliana Williams

A very useful attribute regarding interactivity is the "pointer-events" attribute. This attribute controls the sensitivity of graphic elements regarding the reaction to mouse events. One can either set the attribute to "none" (no reaction to mouse events), "fill" or "stroke" (and a few more options). Using this attribute one can forward the events to the underlying elements or even receive events from invisible elements. This is useful for receiving events from invisible layers or to avoid "flicker effects", which is the case when smaller elements "steal" the event-sensitivity from the underlying larger elements. This is often the case with text elements above a polygon layer.

2.9 Animation

Almost any element and attribute can be animated in SVG. There are currently two ways to implement animations in SVG: the first way is to use Javascript and a timer that repeatedly changes attributes in elements. This approach requires programming know-how but guarantees maximum flexibility when it comes to interpolation methods and logic. The second way is again SMIL, a descriptive way to define animation parameters. SMIL animations can trigger script execution and vice versa. Both, script based and SMIL animations can be triggered by the events listed above.

SMIL offers five elements for descriptive animations: <animate />, the most general element for animating numeric, interpolateable attributes, <set /> for setting non interpolateable attributes, such as string-based values, <animateMotion /> for moving elements along a motion path, <animateColor /> for animating color values and finally <animateTransform /> for animating transform attributes. Common attributes of the five animation elements are "begin", "end", "dur" (duration), "from", "to", "by", "repeatCount", "repeatDur", "fill", "calcMode", "keyTimes", "values", "keySplines".

Of particular interest are the latter attributes: "calcMode" allows to specify the interpolation method (discrete, linear, paced and spline), "keyTimes" and "values" allow the setting of timestamps (in percentage of the full duration) and corresponding values, fixpoints that the interpolation has to respect, and "keySplines" define acceleration or deceleration effects. Two very useful tools for defining keySplines and keyTimes are available at (Hirtzler, 2005b) and (Hirtzler, 2005c). (Neumann, 2003) shows an example where keySplines and keyTimes are combined with progressive line drawing.

Following is an example, with source code and the corresponding graphics, where a text-string is animated along a bezier curve. First, a path is defined with a unique id. Next, a text-element is created with a <textPath /> element as a child. The <textPath /> references the path with the id "curve". Nested within the <textPath /> element are a <tspan /> element, containing the actual text and a negative delta-y offset to place the text above the line, and an <animate /> element that animates the "startOffset" attribute from 0 to 50%. The animation is started if the user clicks on the text with the id "go". Finally, a text element needs to be placed with the id "go" which starts the animation.

```
<path id="curve" d="M100 200Q200,100 300,200
 T500,200 M100 200Q200,100 300,200 T500,200"
 style="stroke:blue;fill:none"/>
<text style="font-size:25;fill:red;">
 <textPath startOffset="0%" xlink:href="#curve">
  <tspan dy="-10">Textpath on Bezier's curve
  </tspan>
  <animate begin="go.click" dur="5s"
   repeatCount="1" attributeName="startOffset"
   values="0%;50%"/>
 </textPath>
</text>
<text id="go" x="550" y="380" style="font-
 size:25;">GO</text>
```

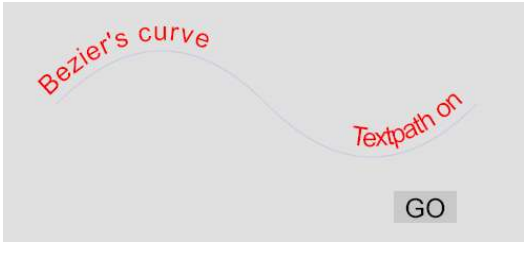

Figure 4: Animated Text along path – © M. Hirtzler, Source: http://pilat.free.fr/english/animer/text_bezier.htm

2.10 Extensibility and Metadata

As a XML based language, SVG supports foreign namespaces. It is possible to define new elements or add new attributes to

existing SVG elements. Elements and attributes in a foreign namespace have a prefix and a colon before the element or attribute name. Elements and attributes in foreign namespaces that the SVG viewer does not know, are ignored. However, they can be read and written by script. Foreign namespaces are used to introduce new elements (e.g. GUI elements, scalebars) and for the attachment of non-graphical attributes to SVG graphic elements (e.g. GIS non-graphical attributes). Those attributes can be analyzed and used to create thematic maps or charts. Additionally, one can display those attributes upon mouse-over. This is not only useful for maps and drawings, but also for user-interfaces, technical drawings, charts, etc.

Following is an example that includes GIS attributes in map data. Within the doctype the attribute lists for the <svg /> and <path /> elements are extended:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
 "http://www.w3.org/Graphics/SVG/1.1/DTD/
 svg11.dtd" [
<!ATTLIST svg xmlns:attrib CDATA #IMPLIED>
<!ATTLIST path
            attrib:ctry_code CDATA #IMPLIED
            attrib:ctry_name CDATA #IMPLIED
>
]>
<svg width="100%" height="100%"
viewBox="-2726130 -5330377 6272480 5460306"
xmlns="http://www.w3.org/2000/svg"
xmlns:attrib="http://www.carto.net/attrib/">
 <g id="countries" style="fill:none;">
  <g id="countries_AD">
   <path id="ctry_336" attrib:ctry_code="AD"
    attrib:ctry_name="Principality of Andorra"
    d="M-679572 -13980921-464  .... " />
   ...
  </g>
 </g>
</svg>
```

For the inclusion of metadata, the W3C consortium recommends the use of the RDF (resource description framework) or the Dublin core standard. The RDF fragments should be included in a <metadata /> tag and should be defined in a foreign namespace (e.g. Rdf). Hier Referenz auf RDF geben.

3. WHY USE SVG FOR E-LEARNING?

3.1 Strengths of the SVG approach

SVG is well suited to play a major role in E-Learning environments. The visualization options available in SVG graphics go beyond competing file formats. Any attribute can be animated and the available interactivity options and script bindings allow the building of fully interactive applications that do not need to hide from stand-alone offline multimedia applications. It is important to note that SVG should be used as a complementary technology and in conjunction with other established web-technologies, such as XML, XHTML, static raster graphics and movies. SVG is primarily a presentation and exchange format that can and should be generated out of other storage formats, databases and XML sources. Specifically, it is recommended that one uses SVG in connection with other XML based e-learning markup languages, such as ELML (Bleisch, Fisler, 2005). SVG should be used for static illustrations, animations and interactive applications.

It is a fact that interactive applications can motivate students. Interactivity can involve and immerse students compared to a dry verbal and static presentation. Benjamin Franklin said "*Tell me and I forget. Teach me and I remember. Involve me and I learn"*. One can let the student solve the problem himself, directly with an interactive SVG application. Specific learning goals can be isolated and students can concentrate on the essential tasks rather than first having to learn complex GIS applications.

When using SVG to teach or illustrate phenomena, one can involve students in different levels of integration. In a first level one can introduce and explain the subject in movie-like animations with a low level of interactivity. The student could still determine the speed of the progress by letting him choose the speed of the animations or by letting him step forward and backwards whenever he wants. In a next level one could guide the student through a workflow and support him with wizard-like interfaces where the student is confronted only with simple decisions within the current context of the workflow. In the highest level of interactivity and complexity, one could leave all decisions open to the student without forcing him into certain chronologic sequences. Finally, after a student worked through an e-learning lesson, he can take self assesments or exams, to test whether the student understood the topic and can correctly use and apply the newly acquired knowledge, both in theory and practice.

The extensibility features of SVG allow to introduce domain specific extensions and share components with other SVG content developers. Metadata can be included in a text-based format that can be equally well read and/or understood by humans and machines. Its text-based nature and the option to embed semantics and context directly in the file format make SVG substantially more accessible than any other graphics format. Search engines can easily analyze the context and semantics of SVG files and applications. They can index title and description tags.

A feature that made HTML successful in the early days of the web is the ability to view the source code of webpages. This way, web developers can learn from other websites and "look under the hood" of Javascript and DHTML applications. Learning (D)HTML and Javascript does not require expensive courses or certifications but can be learnt by anybody who is willing to invest some time for "learning by coding" and is open to learning from other developer's source codes. Unlike many proprietary technologies, such as Flash or already compiled Java applets, which are usually like black boxes, the source code of SVG applications can always be analyzed. Additionally, discussion boards and mailinglists exist where developers can share ideas, code and support new developers. These mailinglists are available for almost any open webstandard and are often faster and better than commercial support. Many examples, tutorial and demo sites exist for SVG where people can learn from each other. Good starting points are (Meinike, 2005), (Hirtzler, 2005d), (Lindsey, 2005) and (Held/Neumann/Williams/Winter, 2005).

The "learning by coding" approach is especially interesting with SVG, since students can directly manipulate graphics and visualizations by changing elements and attributes in the source code. That way, students can learn complex subjects, such as geometry, mathematics and programming in a fun and intuitive way. Teaching experience has shown that students are usually highly motivated if they can immediately see feedback of their programming efforts, even if the courses are more difficult and time-consuming than the average course in their curriculum. Good SVG editing environments allow simultaneous work in the source-code or document tree and graphically in the SVG canvas, with the support of GUI based drawing tools. Updating the canvas graphically, automatically adjusts the source code and vice versa.

One significant advantage of SVG is its XML base. Web developers that are already familiar with (X)HTML, Javascript, XML, CSS and XSLT can immediately use their existing knowledge when learning SVG. The XML base means that SVG content can be created by any text or XML-editor. While it is generally useful to use graphics software or specialized SVG editors for creating or editing SVG content, it is possible to edit content with free text-editors. That means that one can also quickly adopt existing SVG files over low-bandwidth terminal sessions, directly on a webserver.

SVG is also particularly useful for data driven visualization of business data, charts, maps and technical drawings, as it can be generated using XSLT conversion or any scripting or programming language the developer is familiar with. Developers are therefore not limited to a specific serverside framework and there aren't any vendor lock-ins. Furthermore, many GIS or spatial databases already support SVG generation. Having defined conversion rules, updating the SVG presentation is easy and can be automated.

For deploying SVG files and applications, there is usually no license fee necessary. Last but not least, it is important to mention that the development of the SVG specification is open to everyone. Everyone is invited to give feedback on the public W3C SVG mailinglist (W3C, 2005). Members of the SVG working groups, however, need to be either W3C members or "invited experts" due to their valuable contributions. The current members of the W3C SVG working group represent a good average of the graphics and mobile hard and software industry as well as private persons and research institutions. In contrast, proprietary graphics formats, although sometimes documented, are usually under complete control of one company and are often patented. Quite often, if a company goes bankrupt, continuous support is not guaranteed.

3.2 Weaknesses of the SVG approach

Apart from all the positive aspects of SVG there are unfortunately also weak aspects. One issue is that it takes longer for SVG to penetrate the web developer market than initially expected, mainly because important software companies, such as Microsoft and former Macromedia did not actively support SVG. So far, the Adobe SVG viewer plugin is needed to view SVG content. Currently, that viewer has around 30% market penetration. It was only recently, that Opera and Mozilla announced native SVG support for their browsers. Even Adobe had temporarily slowed down SVG support for a while, because some Adobe managers saw it as a threat to PDF, the current number one money maker of Adobe. Luckily, that attitude changed and Adobe is again more actively supporting SVG. SVG meanwhile also enjoys widespread support in the Open Source scene. The SVG file format is supported by the two major Unix desktops (KDE and Gnome). Many OS vector graphics or layout software projects support SVG for import and export. Some of them even use it as their native file format.

The situation is much better with mobile phones. SVG was from the beginning part of the 3GPP standard, and mobile SVG viewers are already widespread available in many 3rd generation mobile phones. SVG is now also natively supported by the Opera and Mozilla mobile versions. Finally, most of the mobile phone vendors, such as Nokia, Motorola, etc. actively support SVG.

One major drawback of the SVG approach is that good tools for content creation are still in their infancy. While it is trivial to create static and animated SVG graphics, tools for scripting development are not yet mature. Hence, the content creation of highly interactive content is still reserved to the more computer literate developers who are used to directly working in the source code. However, that situation is going to change in the long run, esp. with the native integration of SVG in the Mozillla browsers. Several existing ECMAScript debugging tools support professional ECMAScript and SVG development. Debuggers are currently integrated in the Apache Batik project, the eSVG product and the Mozilla and Firefox webbrowsers.

Finally, there is the problem of not being able to hide the source code effectively. This can be a positive feature, but quite a few content creators are hesitant to use open standards where they cannot use code protection. While methods exist to obfuscate Javascript or disable the "View Source" function in the Adobe SVG viewer, it is usually trivial for computer literate people to still have access to the source code. This is also possible for documented binary formats, such as Flash. Quite a few programs exist to decompose swf files and extract the individual media elements, such as graphics, movies and text. To be able to hide the source code in the future, Adobe is looking into the so-called digital rights management.

4. USAGE SCENARIOS AND EXAMPLES FOR SVG IN E-LEARNING ENVIRONMENTS

As already mentioned in section 3.1 it is recommended to not use SVG exclusively, but in collaboration with other webstandards such as XHTML, CSS, XML and XSLT. XML should be used to store the content, while XHTML and SVG can be used for presentation. This strict separation of content and presentation ensures that one can easily generate various versions or deliver content for several output devices by simply applying different stylesheets or conversion rules. The following list of usage scenarios and examples is only exemplary and by no means complete – good starting points for finding additional SVG examples are http://www.svgx.org/, http://www.svg.org/ and http://www.carto.net/papers/svg/links/

4.1 Geometry, Mathematics and Computer Graphics

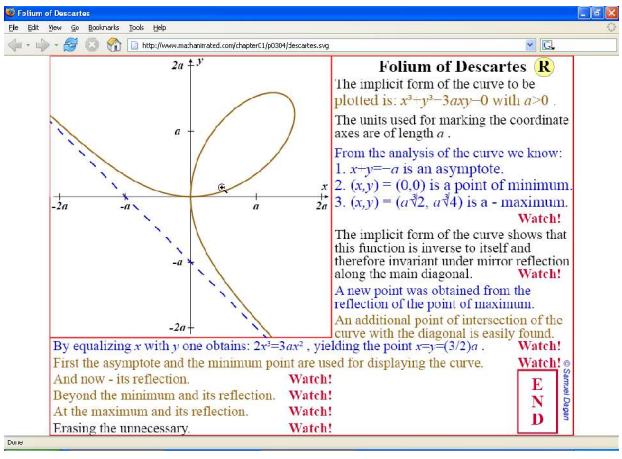

Figure 5: Animated visualization of the "Folium of Descartes" curve, © Samuel Dagan

SVG offers all the basic shapes needed to visualize geometric and mathematical problems. Animation and interactivity can be used to show solutions. Interactivity can also be used to let students demonstrate solutions in self assesments and exams. Examples for SVG in mathematics and geometry can be seen in (Hirtzler, 2005d), (Dagan, 2005) and (Crocodile Software, 2005). The latter citation points to an offline commercial e-learning software that uses the Mozilla webbrowser with SVG

support for the rendering part. Kevin Lindsey (Lindsey, 2003a) provides a good tutorial on how to recursively draw bezier curves with SVG illustrations.

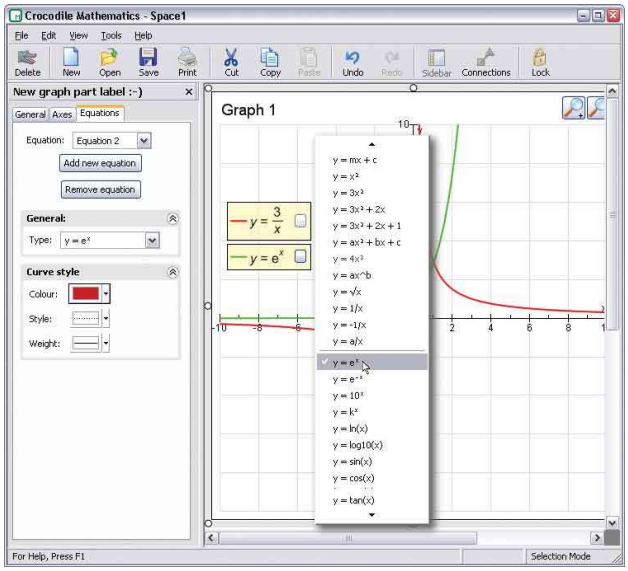

Figure 6: Crocodile Mathematics - a Mozilla and SVG based offline e-learning application, © Crocodile Software

4.2 Engineering, Simulations and Technical Documentation

SVG is well suited for teaching engineering subjects, presenting technical drawings and explaining, visualizing or simulating instruments. Animations can visualize the operation of machines, technical devices or circuit diagrams. In technical drawings one can display non-graphical attributes (such as article numbers or part names) on mouse-over. Tooltips or infopanels can be used for that purpose. In simulations, the user can interactively manage control panels, control flows or change environmental parameters.

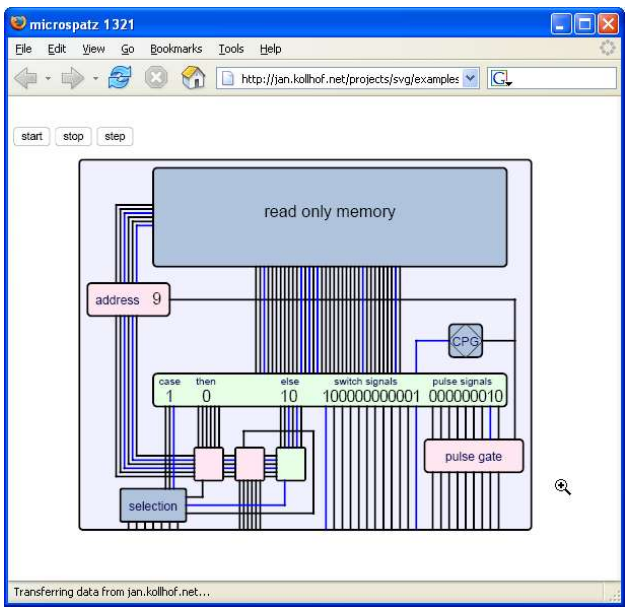

Figure 7: Animated visualization of a microcontroller layout, © Jan Kollhof, Source: (Kollhof, 2004a)

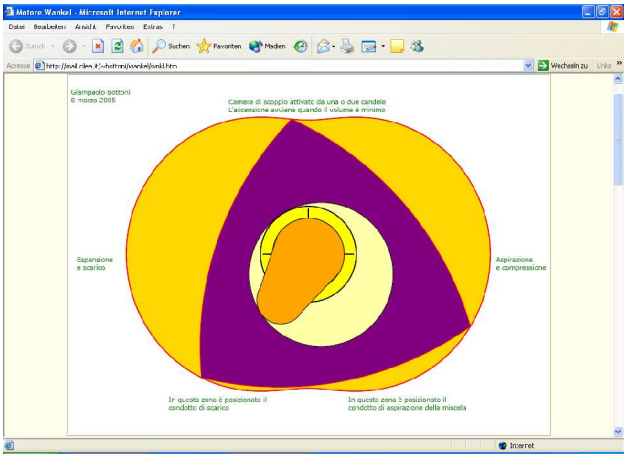

Figure 8: Animated visualization of the Wankel engine, © G. Bottoni, Source: (Bottoni, 2005)

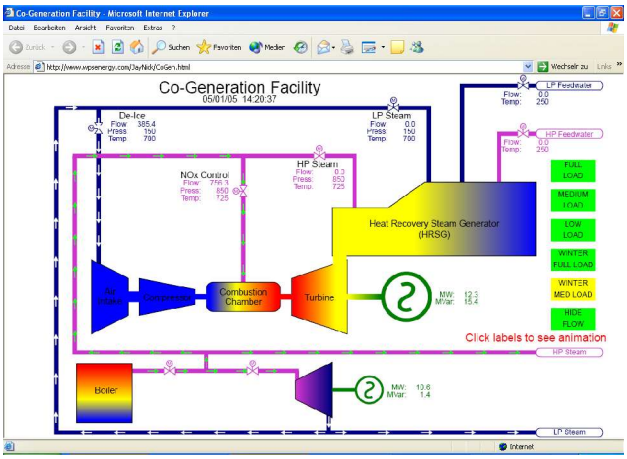

Figure 9: Interactive and animated visualization of a co-generation facility, © Jay Nick, Source: (Nick, 2005)

Examples include electrical troubleshooting diagrams from Hyundai (Hyundai, 2003), an animation of the principle of the Wankel engine (Bottoni, 2005), an animation of the functionality of a micro controller (Kollhof, 2005a), animations and graphics on control systems and real-time metering (Nick, 2005) and SVG-based instruments visualization (Wade Johnson, 2005).

4.3 GIS and Cartography

The rich visualization and interactivity options of SVG make it particularly useful for mapping and GIS. The available fill and stroke options, symbols and markers enable higher quality map graphics and complex symbolizations. Interactivity helps display additional non-graphical data and enables analysis functions. Basic GIS functionality can be directly implemented in SVG, while more complex GIS analysis functions can be delegated to serverside GIS or spatial databases. In the latter case, SVG is only used as a presentation tool. Data acquisition and analysis functions can be directly practiced in interactive SVG applications. Complex workflows can be explained and split up into smaller exercises.

An example for using SVG in mapping applications is the Tirolatlas (Förster/Winter, 2005 – currently requires Internet-Explorer on Windows), an innovative online atlas containing maps, charts, text and tables. An online digitizing tool (Neumann, 2004) demonstrates GIS data acquisition through digitizing where the output can be directly saved into a spatial

database. Additional examples include an illustration of a line simplification algorithm (Lindsey, 2003) and a demonstration of the Dijkstra shortest path algorithm (McCormack, 2004).

Figure 10: SVG based interactive digitizing tool, © A. Neumann, Source: (Neumann, 2004)

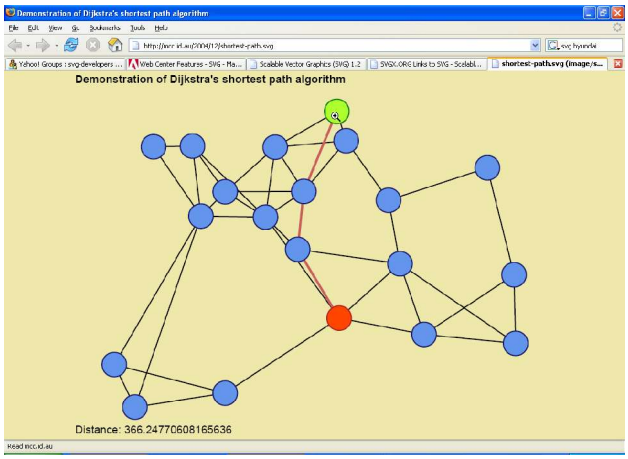

Figure 11: Interactive visualization of the Dijkstra shortest path algorithm, © C. McCormack, Source: (McCormack, 2004)

4.4 Programming and Visualization of Algorithms

Students are usually highly motivated if they can program graphical applications, because they can immediately see feedback and the results are less abstract than with other programming exercises. SVG can also be used for illustrating the functionality of algorithms. The algorithm source code or pseudo code can be displayed next to the graphical result and the program code can be stepped through. This way, the relation between a line in the source code and the influence on the graphical representation is made obvious.

Examples include Jan Kollhofs demonstration of a sorting algorithm (Kollhof, 2004b) and Thomas Meinikes SVG demonstrations "SVG - Learning by Coding" (Meinike, 2005), an extensive collection of SVG code examples.

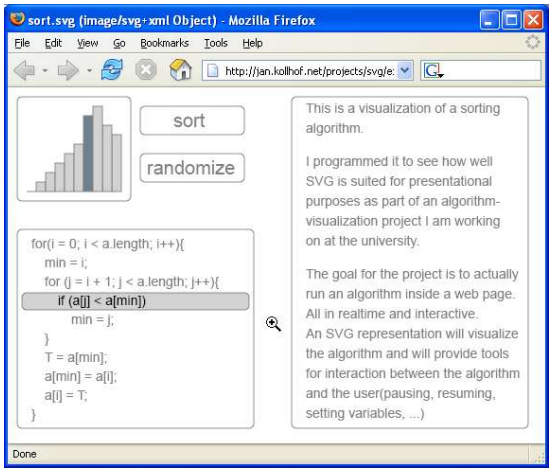

Figure 12: Animated visualization of a sorting algorithm, © Jan Kollhof, Source: (Kollhof, 2004b)

4.5 Games and Kids

Many examples show the usefulness of SVG for smaller games and animations. Games can have a high pedagogical value if they are carefully authored and transport learning subjects adequate to the age of the intended audience. Useful games are crossword puzzles, quizzes in any subjects, finding locations in maps, ordering and structuring geographic phenomena or solving geometrical or mathematical problems.

Examples include the "Tirol for Kids" section in the Tirolatlas, containing various geography or cartography related animations and games for learning and testing the geography knowledge of kids and students (Förster/Winter, 2004). A website dedicated to SVG and gaming (Ellis, 2005) lists a number of SVG based games.

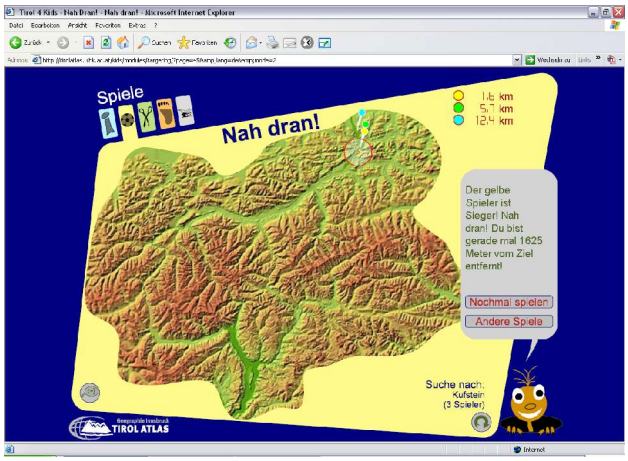

Figure 13: "Nah dran", an SVG game for testing geographic knowledge, © Tirolatlas, Source: (Winter/Förster, 2004)

5. SVG CONTENT GENERATION

Although easy and user friendly SVG development tools are still in their infancy, there are already more options to generate SVG content than for any other graphics format. Following is a non-exhaustive list of SVG generation options. In many cases a combination of different generation or conversion options lead to a satisfying result.

5.1 Text Editors and XML Editors

The text or XML base of SVG is probably one of the largest advantages. It allows the creation of content in any simple text editor, even across small bandwidth network connections. XML editors are more comfortable and allow validation of SVG syntax. Many XML editors also support code folding and typing support. Some editors feature an internal preview of the graphical rendering of the SVG content.

5.2 Graphics Format Converters

Many general purpose graphic converters nowadays support SVG for reading and writing. Specific converters, such as the Docsoft QuickSVG CGM to SVG converter (Docsoft, 2005), even support the conversion of interactivity functions or the inclusion of non-graphical data.

5.3 Graphics Software

Most companies developing graphics or CAD software also support SVG, among them Adobe (with Illustrator) and Corel (with CorelDRAW). Almost all open source graphics software supports SVG for import and/or export.

5.4 Specialized SVG Editors

Specialized SVG editors usually use SVG as their native file format. They are specifically tailored towards the features of SVG. While the creation of SVG static geometry and simple animations is currently covered well enough, most editors still lack professional scripting and interactivity support. Creating complex interactive SVG applications still requires good programming skills. In this category there are SVG editors such as the Open Source Inkscape (Inkscape, 2005) project, an animation editor from Ikivo and Adobe (Ikivo, 2005), and RapidSVG (Xstream, 2005), a tool specifically tailored towards e-learning, with support for simple GUI widgets.

5.5 SVG printer drivers

SVGMaker provides a SVG printer driver for the Windows platform, which enables SVG printing support for any Windows application. The program works similar to the Adobe Distiller software and also provides a control for page navigation in multipage documents.

5.6 XSLT Conversion

SVG developers can benefit from the fact that SVG is XML based and can easily convert SVG content from any XML data utilizing XSLT conversion. This method is particularly useful for data-driven applications and documents that need automated conversion workflows.

5.7 Scripting and Programming Languages

Again, developers can benefit from the text-base of SVG. Any programming language provides text file output. Many programming or scripting languages even provide libraries for reading and/or writing SVG content.

5.8 Export from Databases

Many (spatial) databases already provide SVG export support for maps and charts. Examples include Postgis/PostgreSQL and Oracle. Given the fact that most DBMS provide procedural language extensions and APIs to most popular programming languages it is relatively easy to extend other DBMS for that purpose.

5.9 Export from GIS or Mapping Servers

Most of the current GIS software vendors already support SVG either as an export format or as part of their webmapping server products. Companies that support SVG include ESRI, Intergraph, Microimages, Smallworld, Safe Software (FME) and many third party vendors that enhance existing GIS with more sophisticated SVG export functions. Examples of third party products are MapViewSVG (UISMedia, 2005) for ESRI and SVGMapMaker from DBXGeomatics (DBXGeomatics, 2005) for Mapinfo.

6. NEW DEVELOPMENTS IN THE UPCOMING SVG 1.2 SPECIFICATION

Compared to the SVG 1.1 specification, in which the W3C introduced sub profiles, the SVG 1.2 specification is a major upgrade. Numerous new features potentially enhance the multimedia experience, add new graphics, visualization and networking options, and improve and facilitate interactivity and application development. The new features listed below are only a very brief summary of the upcoming SVG 1.2 additions and improvements. For a full list of new features, please consult (W3C, 2005). The introduction of the SVG 1.2 specification is not without controversy, as it partially overlaps with the scope of other webstandards and is becoming increasingly complex.

6.1 XBL Support (XML Binding Language)

sXBL will allow custom markup extensibility via pre-packaged components. Developers can create re-usable higher level components in a custom XML vocabulary. It builds on the concept of a shadow tree containing lower level SVG elements that will be rendered instead of the original custom markup. The shadow tree will be generated using templates and scripting. Applications for XBL are widgets, visual effects and domain specific extensions with common markup. Cartographers could e.g. define a XBL version of north arrows or scalebars. It is beyond the scope of this article to explain the functionality of the XBL technology.

6.2 Text Wrapping

SVG 1.2 finally adds text wrapping support. Text can flow in arbitrary shapes and is constantly updated if the connected shape changes size and shape. Exclude shapes can define obstacles a text has to wrap around. FlowText can have a different coordinate system than the linked flowRegion. This means if a flowRegion is rotated, the text lines can stay horizontally aligned. Hourglass example, Pilat Example.

6.3 Editable Text and Text Selection Events

Implementing editable text (e.g. text input form elements) in SVG up to version 1.1 was complicated. One had to write relatively complex script constructs to simulate text input boxes or implement simple text editors. SVG 1.2 will add an "editable" attribute that allows the editing of single text elements or flowtext. A text selection interface will allow script access to the text selection mechanism and add events that fire if a text was selected.

6.4 Official Audio and Video Support

While the Adobe SVG viewer supports audio with a proprietary extension since version 3, SVG 1.2 now officially adds support for audio and video media. At the time of writing it has not officially been decided what video formats and codecs are mandatory and which are optional. The SVG viewer should, however, at least support the royalty free ogg vorbis audio format. Next to the usual SMIL attributes and synchronization options, the <audio /> and <video /> elements support an "audio-level" (volume) attribute. Video content can be transformed, clipped and masked and may be overlayed by transparent elements or can be transparent by itself. SMIL elements or script interfaces can control the playback of the media.

6.5 Transition Effects

SVG 1.2 supports the transition effects (e.g. checkerboard, dissolve, blending, etc.) defined in SMIL 2.0. These effects are useful for multipage documents, presentations or slideshows. The effects are already implemented in the Adobe SVG viewer version 6.

6.6 Multipage Support

New <pageset /> and <page /> elements will enable multipage support. This feature should be useful for presentations, multipage publications, books, cartoons, etc. Until now, one had to simulate this feature by changing the visibility or display attribute on groups simultaneously. Page navigation should be possible by SMIL, script or page keys.

6.7 Streaming Support and Progressive Rendering

Streaming support allows viewers to start displaying content or start animations while the file is still downloading. For that purpose, authors can specify that the timeline starts "onStart", after an elements opening tag is fully parsed, or "onLoad" (the default) which is fired after a tag is fully processed. Authors can also define whether content can be discarded after using it, or if the viewer needs to keep it because other elements still need to reference it. The new specification also defines how progressive rendering should be implemented. A "prefetch" attribute controls when external resources need to be loaded.

6.8 Better Time Control

In SVG 1.2 it will be possible to control the playback speed of animations. Values are relative to the parent time container. Negative values cause elements to play backwards. In SVG 1.2 the author can introduce multiple independent time containers. Additional time containers may be nested. In that case, the "speed" attribute accumulates. Until now, animation playback could only be controlled globally. It will also be possible to "jump" directly to a snapshot in time. In the future, animations and media elements can also be started using access keys.

6.9 Multiresolution Images

SVG 1.2 will support multiresolution images. Threshold values ("min-pixel" size and "max-pixel" size) control what content ("subImage") will be displayed. SubImages can contain both raster and vector graphics and have the usual "x", "y", "width" and "height" attributes that can be used to determine which parts of the subImages need to be displayed.

6.10 Vector Effects

Vector Effects are particularly interesting for GIS and cartography, but also for interactive drawing applications. They serve several purposes: multistroking and multifilling allow the combination of various fill and stroke options, e.g. a gradient can be applied in combination with a pattern or uniform color to the same geometry, or a multistroke freeway line signature can be made up of several stroking options. A setBack effect can interrupt stroking before and after a vertex. Reverse allows to reverse a path, thus affecting animations and markers. Join and "vePathRef" allow the buildin of a new path out of other paths. This method can be used to build simple topological structures by building polygons out of edges. Union, intersect and exclude allow the combining, intersection and substraction of paths.

6.11 Additions to the Rendering Model and Enhanced Compositing

SVG 1.2 introduces a background element and new options for alpha compositing, such as clip-to-self, knock-out and "comp-op". Comp-op specifies composition operators, such as "xor", "multiply", "difference", "exclusion", "lighten", "darken".

6.12 Extended Links

Extended links allow links to multiple targets where the user is presented with a choice (e.g. menu or popup list). This is useful for many interactive applications, e.g technical drawings. A switch may even provide text in different languages.

6.13 Application Development, Scripting and DOM enhancements

One goal of this specification section is to finally standardize existing proprietary extensions, such as the network interfaces ".getURL()" and ".postURL()" or the "window" object. Furthermore, there will be support for focus and navigation and tooltips. DOM enhancements will introduce better support for coordinate system translations and a new "wheel" event for mouse wheels or "jog dials". There will also be better network interfaces (e.g. URL Request or sockets, including the option to abort a request), a progress event that allows the monitoring of download progress, and a better timer interface. A file upload dialog and the possibility for persistent client side storage support (similar to cookies) further facilitate SVG application development.

7. SVG BOOKS AND WEBSITE REFERENCES

7.1 Selected SVG Books:

Bader, H., 2004, SVG Reporting, Software und Support Verlag.

Cagle, K., 2002, SVG Programming, The Graphical Web, APress.

Campesato, O., 2003, Fundamentals of SVG Programming: Concepts to Source Code (Graphic Series), Charles River Media.

Eisenberg D., 2002, SVG Essentials, O'Reilly.

Fibinger, I., 2002, SVG - Scalable Vector Graphics, Praxiswegweiser und Referenz für den neuen Vektorgraphikstandard, Markt und Technik.

Frost, J., Goessner S. and M. Hirtzler, 2003, Learn SVG, Self Publishing (http://www.learnsvg.com/). Also available as an e-book, in french language and with online tutorials.

Laaker, M., 2002, Sams Teach Yourself SVG in 24 Hours, Sams Publishing.

Watt, A. and Lilley C., et.al, 2002, SVG Unleashed, Sams Publishing.

7.2 References from websites:

Adobe, 2005a, Adobe SVG Viewer Download Area, http://www.adobe.com/svg/viewer/install/ (accessed 26 Apr. 2005)

Adobe, 2005b, Adobe SVG Viewer Pre-Release Download Area, http://www.adobe.com/svg/viewer/install/beta.html (accessed 26 Apr. 2005)

Apache Batik, 2005, Batik SVG toolkit, http://xml.apache.org/batik/ (accessed 26 Apr. 2005)

Bleisch S. and J. Fisler, 2005, https://sourceforge.net/projects/elml/ (accessed 29 Apr. 2005)

Bottoni G., 2005, Motore Wankel, http://mail.cilea.it/~bottoni/wankel/wnkl.htm (accessed 29 Apr. 2005)

Crocodile Software, 2005, Crocodile Mathematics, http://www.crocodile-clips.com/crocodile/mathematics/index.htm (accessed 29 Apr. 2005)

Dagan S., 2005, Math Animated, http://www.mathanimated.com/ (accessed 29 Apr. 2005)

DbxGeomatics, 2005, SVGMapMaker, http://www.dbxgeomatics.com/products/svgmapmaker/SVGMapMaker.aspx (accessed 29 Apr. 2005)

Docsoft, 2005, Quick.SVG 2005, http://www.quicksvg.com/ (accessed 29 Apr. 2005)

Ellis D., 2005, SVG Games, http://a.1asphost.com/svggames/ (accessed 29 Apr. 2005)

FontForge, 2005, FontForge, http://fontforge.sourceforge.net/ (accessed 27 Apr. 2005)

Förster K. and A. Winter, 2005, Tirolatlas, http://tirolatlas.uibk.ac.at/ (accessed 29 Apr. 2005)

Held G., Neumann A., Williams J. and A. Winter, 1999-2005, Scalable Vector Graphics Examples and Articles, http://www.carto.net/papers/svg/samples/ (accessed 26 Apr. 2005)

Hirtzler M., 2002, SVG Filters, http://pilat.free.fr/svgopen/paper.htm (accessed 27 Apr. 2005)

Hirtzler M., 2005a, Using SVG filters, http://pilat.free.fr/english/filters/index.htm (accessed 27 Apr. 2005)

Hirtzler M., 2005b, keyspline, http://pilat.free.fr/english/animer/keysplines.htm (accessed 28 Apr. 2005)

Hirtzler M., 2005c, keytimes, http://pilat.free.fr/english/animer/keytimes.htm (accessed 28 Apr. 2005)

Hirtzler M., 2005d, Pilat Informatique Educative, http://pilat.free.fr/ (accessed 29 Apr. 2005)

Hyundai, 2003, Hyundai 2003 Tiburon Sample ETM, http://www.hmaservice.com/svgdemo/ (accessed 29 Apr. 2005)

Ikivo, 2005, Ikivo Animator, http://www.ikivo.com/animator/index.html (accessed 29 Apr. 2005)

Inkscape, 2005, Inkscape - Open Source Scalable Vector Graphics Editor, http://www.inkscape.org/ (accessed 29 Apr. 2005)

KDE developers, 2005, KSVG, http://svg.kde.org/ (accessed 26 Apr. 2005)

Kollhof J., 2004a, Microspatz – Micro Controller Simulation, http://jan.kollhof.net/projects/svg/examples/microspatz/microspatz.svg (accessed 29 Apr. 2005)

Kollhof, J., 2004b, A sorting algorithm visualization, http://jan.kollhof.net/projects/svg/examples/sort.svg (accessed 29 Apr. 2005)

Lilley C., Jackson D., J. Ferraiolo et al, 2005, Scalable Vector Graphics – XML Graphics for the Web, http://www.w3.org/Graphics/SVG/ (accessed 26 Apr. 2005)

Lindsey K., 2001-2005, SVG Examples, http://www.kevlindev.com/ (accessed 26 Apr. 2005)

Lindsey K., 2003a, Drawing Bezier Curves, http://www.kevlindev.com/tutorials/geometry/bezier/index.htm (accessed 29 Apr. 2005)

Lindsey K., 2003b, Simplifying a Polyline, http://www.kevlindev.com/tutorials/geometry/simplify_polyline/index.htm (accessed 29 Apr. 2005)

Lindsey K., 2005, Filters, http://www.kevlindev.com/tutorials/basics/filters/feComponentTransfer/index.htm (accessed 27 Apr. 2005)

McCormack C., 2004, Demonstration of Dijkstra's shortest path algorithm, http://mcc.id.au/2004/12/shortest-path.svg (accessed 29 Apr. 2005)

Meinike Th., 2005, Daten verdrahten, SVG - Learning by Coding, http://www.datenverdrahten.de/svglbc/, (accessed 29 Apr. 2005)

Mozilla, 2005, Mozilla SVG Project, http://www.mozilla.org/projects/svg/ (accessed 26 Apr. 2005)

Neumann A., 2003, Example for an animated bus track, http://www.carto.net/papers/svg/samples/animated_bustrack.shtml (accessed 29 Apr. 2005)

Neumann A., 2004, Digitizing Tool, http://www.carto.net/papers/svg/digi/ (accessed 29 Apr. 2005)

Neumann A. and J. Williams, 2005a, Manipulating SVG Documents Using ECMAScript and the DOM, http://www.carto.net/papers/svg/manipulating_svg_with_dom_ecmascript/ (accessed 27 Apr. 2005)

Nick J., 2005, SVG Demos of Control Systems and Real-Time Metering Systems, http://www.wpsenergy.com/JayNick/default.asp (accessed 29 Apr. 2005)

Opera, 2005, Ahead of the game: Opera introduces Native SVG support in Desktop Release, http://www.opera.com/pressreleases/en/2005/03/16/ (accessed 26 Apr. 2005)

UISMedia, 2005, MapViewSVG, http://www.mapview.de/eng (accessed 29 Apr. 2005)

Wade Johnson, G., 2005, SVG based Instruments Demo, http://www.anomaly.org/wade/projects/instruments/Instruments.svg (accessed 29 Apr. 2005)

Winter A. and K. Förster, 2004, Tirol for Kids, http://tirolatlas.uibk.ac.at/ (accessed 29 Apr. 2005)

W3C, 2005, Scalable Vector Graphics (SVG) Full 1.2 Specification, http://www.w3.org/TR/SVG12/ (accessed 29 Apr. 2005)

Xstream, 2005, Xstream RapidSVG, http://xstreamsvg.com/ (accessed 29 Apr. 2005)

7.3 Acknowledgements

I'd like to thank the SVG community and W3C SVG working group for providing numerous examples and their continuous support with SVG and programming problems. The open source community provides various SVG authoring and viewing software for free. Specifically, I'd like to thank the Batik, Inkscape, KSVG, librSVG and Mozilla team. Furthermore, I'd like to thank the Adobe SVG team for their feedback and fruitful discussions which helped solving SVG development problems or introducing new features in the SVG specification. My girlfriend Juliana Williams helped correcting my typos and disentangling complicated sentences.