Cartographer's Wishlist Regarding Future SVG Specifications/Implementations

Compilation of the SVG.Open 2003 panel discussion, AN/BP, 2003-09-18

Markers

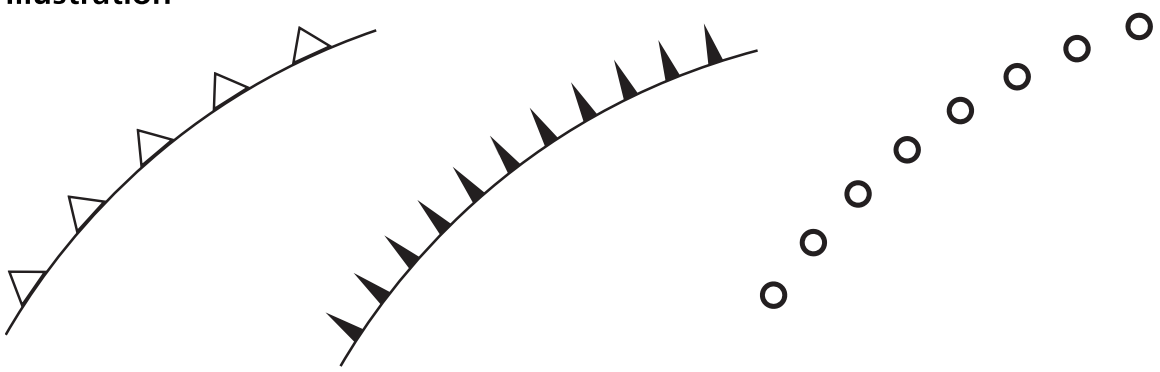
Problem Description

Markers are so far only possible at line-ends or vertices. For many complex cartographic line styles it would be useful to place markers along a line at equal distance, f.e. every 2 units. Probably the problem could be solved with different approaches than using markers. A current workaround is that you use "textPath" element and place SVG glyphs along line (maybe a misuse of the textPath element).

Use Cases

Cartographic symbolization (e.g. topographic maps, geologic maps), technical drawings, CAD

Illustration



The above figure shows cartographic lines commonly used in topographic or geologic maps (e.g. edges, borders, fences, walls, hedges, etc.).

Events on Markers

Problem Description

For many cases it would be useful to have events on markers/vertices. The event handler should also return the number of vertices (position index) from the beginning of the path.

Use Cases

In GIS and CAD drawings there might be attribute data associated with vertices. In order to display that data we would need events. Another use case would be line-editing: a user could select and drag a vertex and edit an existing path (using some ECMAscript).

Shared Geometry and Directed Pathes

Problem Description

In GIS, a common data model is to have an indexed list of nodes, lines and polygons. The nodes help to build line segments, the line segments help to build polygons. Because every node, line and polygon has a unique id, they can be referenced and re-used. This means that f.e. a line-segment can help to build two adjacent polygons. In many cases GIS systems also use a constraint that polygons should not overlap (not suitable for all geographic models, of course). This vector data model, together with the concept of topology, helps to avoid redundancy (data for adjacent polygons is only stored once), to ensure data consistency and is also the base for many spatial algorithms, regarding geometry operations and spatial analysis. Common topology assumptions in GIS are: Connectivity (lines connect to each other at nodes), Area Definition (lines that connect to surround an area define a polygon), Contiguity (Lines have directions and left and right sides). See more on this common GIS data model and topology, incl. illustrations, in the attached PDF "gis_vector_data_model.pdf" (source ESRI, www.esri.com). Please note that in ESRI terminology an arc equals a line-segment.

Use Cases

To avoid redundancy in data storage, save bandwidth and enable spatial analysis. For many cases in mapping, it would also help to swap the direction of a path. This would help for animation along a line (animateMotion): f.e. a train would move along a path towards it's end, than the path-direction would be swapped and the train can move back towards the start. It would also help for directed markers and arrows, or progressive path-animations, esp. if you do subsequent progressive path animations (f.e. for showing routing results of navigation systems). Directed pathes would also be useful in flowchart, orgcharts and circuit diagrams.

Zoomable/Non-zoomable nested SVG's

Problem Description

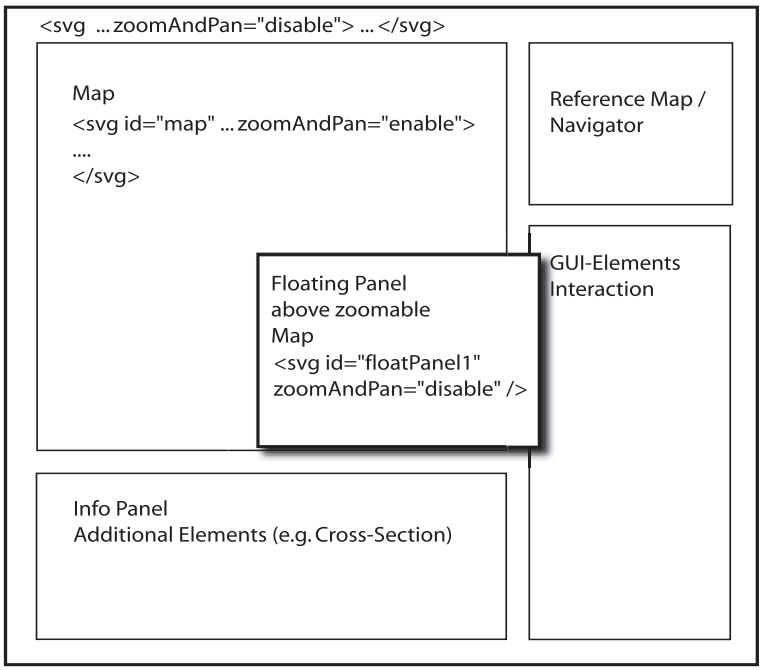
Sometimes it would be useful to have a non-zoomable User-Interface written in SVG with a zoomable graphic within. It would be useful to have an attribute within a nested SVG-element that allows to set "zoomAndPan" to "enable" or "disable", not just for the outermost SVG-element. To achieve that goal currently, one has to use scripting.

Use Cases

User interfaces for webmapping applications, interface to CAD applications/drawings, scientific visualization, games, etc.

It should still be possible to have non-zoomable areas (such as floatable panels of a GUI) above/overlapping zoomable areas, such as a map or complex zoomable Graphics – see illustration below.

Illustration



The above illustration shows a SVG mapping application, where the outermost svg-element should be non-zoomable, because it represents the GUI. Nested within is a mappping area where the user should be able to zoom. Finally we should be able to use non-zoomable floatable panels above zoomable SVG-elements.

Non-Scalable Symbology (stroke-width, patterns, dashes, etc.)

Problem Description

For many cases (CAD, GIS, mapping) a line-width, dashing or pattern should stay constant when zooming in to the graphics. It could be defined in percentages of the initial viewBox. Chris said that this could be handled relatively easy by rationalizing the way that coordinate spaces are handled in internal references.

Use cases

CAD, mapping applications, Flowcharts, Business Graphics

Access to Additional Geometry Properties

Problem Description

For some applications it would be useful to have access to different geometry properties, such as the following. Since many SVG viewers might already store some of these properties internally, it would be useful to expose those attributes to the SVG developer.

- element.getArea()

- element.getPerimeter() (this should already be handled by .getTotalLength() ?)
- element.getCentroid() (which computation method?)
- path.getNrVertices() or .getNrSegments()
- path.getNrOfSubpaths() (path containing multiple polygons, delimited by “z” command)
- subPath.isClosed() ?
- any more to add?

Use Cases

GIS, Mapping, CAD, geometry applications, Learning/edu-Applications. GIS software usually calculates these values and add them as standard columns to a geometry's attribute table.

PathData API

Problem Description

In addition to the above geometric properties, SVG developers sometimes want to change parts of the path Data (d-attribute), f.e. to apply custom filters and algorithms or change individual nodes when editing existing path Data. A pathData API would need to expose individual subPathes, line segments/curves/arcs and coordinates. Furthermore a developer should be able to delete individual segments at a certain position or add a new segment. I suggest relative coordinates should be translated to absolute coordinates by a possible pathData API. No idea how complicated this can get from an implementers point of view? Currently a SVG developer can read/split and rewrite the entire d-attribute, but this can get complicated and error-prone with more complex pathes.

Use Cases

Interactive SVG based drawing Applications, Interactive Webmaps, GIS, CAD, Business Graphics (f.e. filtered line graphs)

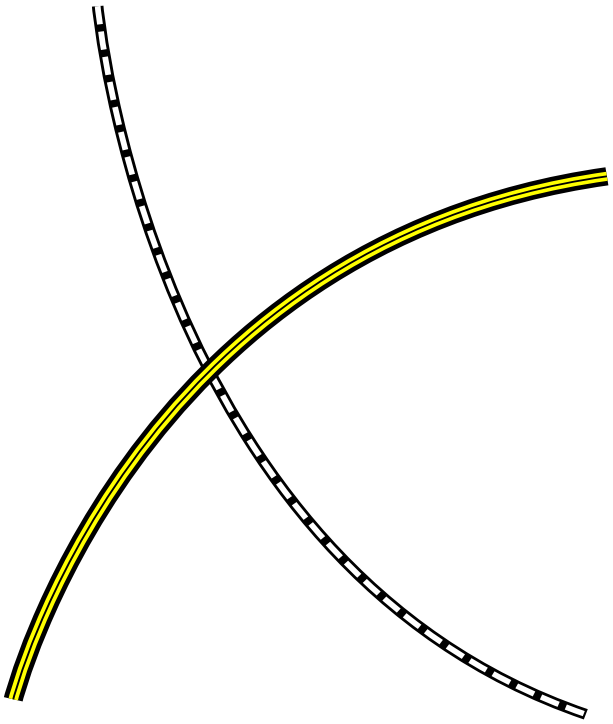
Vector Effects

Problem Description

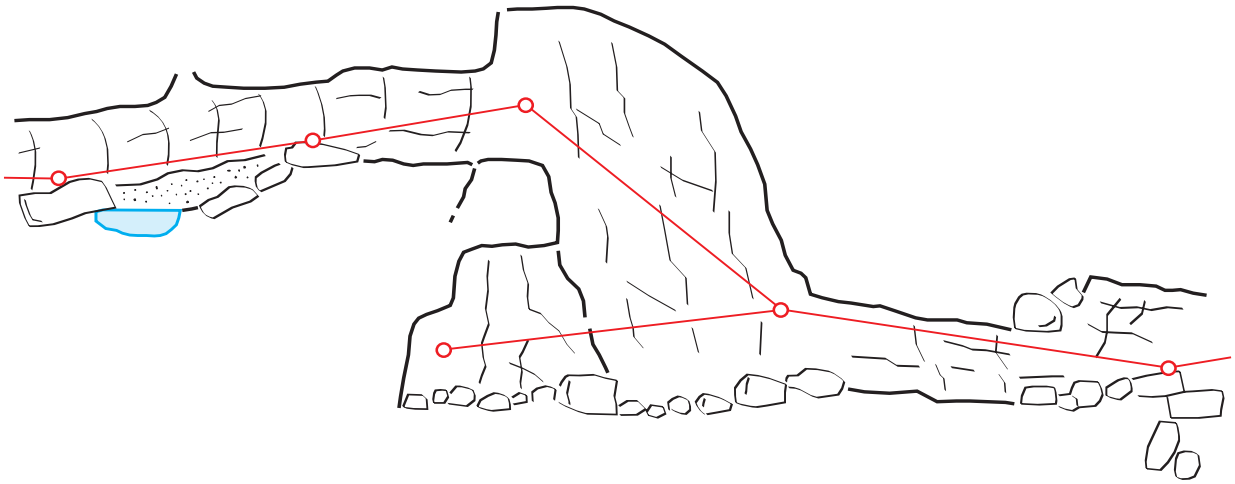
Vector Effects open a whole new range of creativity applications if they are implemented in an extensible way. I am not sure how complex it's implementation for the SVG viewer makers would be. And so far I don't know exactly which effects would actually be covered. I mainly list use cases where we could use vector effects:

Use Cases

According to Peter Sorotokins original posting on vector effects from 2003-09-11 vector effects would allow for multiple strokes, a requirement that cartographers have frequently, see f.e. the figure below where multiple strokes make up one line symbolization for a freeway or railway signature.



To my understanding vector effects would also allow to simulate illumination effects or calligraphic lines. Those are frequently used in artistic drawings or cliff drawings in maps, that help to give a line a more “human/artistic” touch compared to static, technical drawings with fixed linewidthes along the whole line. See f.e. the example below where rock structures in a cave-map are using calligraphic line effects.



Another use case would be the generation of sinuous lines (or wavelines – I don't know the correct english term) as shown in the illustration below, where the original straight line would turn into a sinuous line after applying the vector effect.

Additional use cases would be random distortions within the range of the specified parameters to create a jittery line effect, or some of the more commonly used artistic media brushes that Adobe Illustrator and Corel Draw supply.

I don't know if the regular (or in some cases random) placement of symbols/markers along lines (see marker topic above) could also be achieved using vector effects?

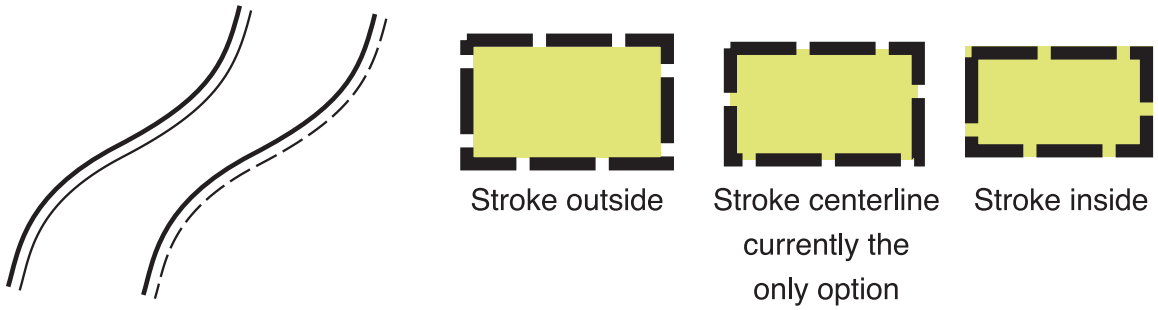
More Stroking Issues

Problem Description

In some cases we would like to stroke only to the left or right of a path, or like to specify an offset value to the path geometry for the stroke to be rendered. See also quotation in the SVG 1.2/2.0 requirements doc: "SVG may allow the user to control the user the location of the stroke, for example the stroke could be centered on the outline, adjacent to the outline and outside the shape, or adjacent to the outline and inside the shape." These effects should be combinable with multiple strokes (see also vector effects). Or could this even be achieved using vector effects?

Use Cases

Assymetric lines are commonly used in topographic and geologic maps. May also be used in other technical drawings. See examples for assymteric linestyles below. The rectangle illustrates inside, centerline and outside stroking.



ImageData API

Problem Description

For quite a few applications it would be useful being able to read and write individual pixel values. To get the values, developers would probably call image.getValue(x,y) and get the gray/RGB/Alpha Channel values as a result. To write data, SVG developers should be able to set colors and alpha values at a certain position or replace whole chunks of image data (f.e. an image row?). For some cases it would also be useful being able to change the color table of indexed images. Imagine having a binary (0,1) image representing forest areas, where you want to allow the user setting the green value of the forest areas ...

Use Cases

Interactive SVG based image image editing and processing, GIS, webmapping. In GIS and remote sensing it is common that a VAT (Value Attribute Table) is associated with pixel values (gray value,

RGB value), f.e. imagine a landuse image where a index value of 1 would represent rye, whereas 2 would represent Barley and 3 would represent wheat, etc. Being able to read the cell value, a GIS/mapping application could display the corresponding value, matched with the VAT. In the field of interactive SVG based terrain modeling you would be able to do clientside calculations and display the result as an image (f.e. aspect, slope, visibility, etc.) Obviously all image manipulations and interactive SVG based image processing would also benefit from such an API.

Addition/Comment Brandon Plewe:

It seems that a more elegant long-term solution would be for PNG and/or JPEG 2000 and/or GeoTIFF to have a standard DOM that could be supported in SVG, allowing us to use the same approach that SVG 1.0 (and now ASV6) allows for SVG "images." I realize that those standards are beyond our control, but it really suprises me that the image people haven't expressed a desire to use ECMAScript to make interactive images. (remark Andreas: isn't the PNG spec under W3C control?)

Selections/Multiselections

Problem Description

Many interactive SVG applications require the ability to select/multiselect elements. Of course, this is currently possible using scripting, however if the WG could find an idea that makes this task easier for SVG developers, probably many developers and applications would benefit. As a result of this mechanism, the developer should get back a reference list of unique ids, thus allowing him to later delete and manipulate selected elements.

As it is the case with selective zooming, the author should be able to determine regions in the DOM tree that are selectable (f.e. map, interactive drawing) and regions that are not (f.e. GUI section). This could f.e. be set at a svg-element level or at a group-level. The SVG User Agent should provide selection mechanisms: selection (single, multiple) and deselection mechanisms (probably like the current convention that shift key allows for multiple selections and a repeated click on an element deselects it again). The selection state can be activated by customized contextMenu, shortcuts or script calls. An area selection (select all elements that are intersected with a rectangular area, or are fully within a rectangular area) would be additionally useful.

Use Cases

Interactive Drawing Applications, mapping, GIS, CAD, interactive business graphics, educational applications, etc.

Tooltips

Problem Description

Tooltips would be useful for giving hints on GUI elements or tools, as well as to display additional attribute data (f.e. a GIS attribute, or geometric property). Tooltips should either display text in it's simplest form, but maybe also arbitrary SVG content. The content of the tooltip (text or arbitrary SVG) should be dynamic at runtime, for being able to react to user interaction (f.e. display current mouse position).

Use Cases

Too many to list here. Should be interesting for all domains.

Executing High Performance Code

Problem Description

To find a way to bring in high-performance code (e.g. Java) to perform complex tasks (i.e. viewer applets). There are many things I'd like to do to make powerful interactive graphics (e.g. spatial analysis, automatic labelling, spatiotemporal query), but it is not feasible in ECMAScript. Perhaps SOAP/XMLP would solve this problem; does it support service migration to the client, or just remote web services? I know that this is probably a viewer issue, not an SVG encoding issue, but both need to be standardized.

Use Cases

All sorts of applications where high performance tasks (too complex for ECMAScript) should be executed on the client, but the output should be displayed finally in the SVG viewer plugin.

The Cartographers are very thankful for many of the improvements that SVG brought for us and are also addressed in the upcoming SVG 1.2 specification. We really appreciate the huge work that working group members and implementers invest to make SVG a reality!