SuperPath: A Necessary Primitive for Vector Graphic Formats

Jean-Claude Moissinac, Cyril Concolato, Jean Le Feuvre

► To cite this version:

Jean-Claude Moissinac, Cyril Concolato, Jean Le Feuvre. SuperPath: A Necessary Primitive for Vector Graphic Formats. SVG Open 2010, 2010, Paris, France. hal-01793955

HAL Id: hal-01793955

<https://hal.science/hal-01793955>

Submitted on 17 May 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

SuperPath : A Necessary Primitive for Vector Graphic Formats

Jean-Claude Moissinac	Cyril Concolato	Jean Le Feuvre
<i>Telecom ParisTech; Institut Telecom; CNRS LTCI</i> 46, rue Barrault 75634 PARIS CEDEX 13 {moissinac, concolato, lefeuvre}@telecom-paristech.fr		

Abstract—Most 2D graphic languages deployed on the Internet today lack the ability to represent the topology of contiguous regions efficiently. This is becoming problematic with the growing number of map-based applications. In this paper, we introduce a new graphical primitive, called SuperPath, to extend existing languages for 2D vector graphics, such as SVG. This primitive provides a way to represent the contours in a 2D graphic with a set of reusable chunks of contour. We present the motivations for this work with examples of 2D maps and some associated problems related to their editing, rendering, or adaptation. We give some results, which were obtained using the new primitive.

Keywords: *Picture description languages; multimedia languages; vector graphic; map; adaptation; anti-aliasing*

I. INTRODUCTION

With the increasing diversity of terminals and networks, modern multimedia technologies need to address multi-publication and adaptation of multimedia services and documents to environmental contexts such as terminals and user preferences. Some previous works, like [1], have shown that the more structured a document is, with rich meta-information, the greater the capability to process it, to modify or adapt it is.

In our work on multimedia documents, we usually work with common 2D graphic formats such as SVG –Scalable vector Graphics- or MPEG-4 BIFS –Binary Format for Scene- and MPEG LAsE-R –Lightweight Application Scene Representation- as well as other, more specialized languages (MIF –MapInfo Interchange Format-, InkML, etc.). SVG is becoming a common representation for 2D graphic documents (web browsers, Linux distributions) and, with the adoption of the Tiny profile by groups such as MPEG, OMA –Open Mobile Alliance- or 3GPP –3rd Generation Partnership Project-, it is expected to become a central format on mobile devices among which phones and, portable media players. MPEG-4 BIFS is used to display text, media and graphics for interactive services in the T-DMB broadcasting system, used in Korea for mobile TV and in France for digital radio.

Both formats rely on a similar representation of 2D vector graphics based on the use of contours with filling and stroking properties. However, these languages have limitations when editing, adapting or interacting with complex graphic content. More precisely, both languages lack the ability to define an area as a list of references to pieces of contour. This feature is present, for example, in the MapInfo Interchange Format (MIF/MID) [8] dedicated to map representation. In this paper, we present an extension to

the SVG language called SuperPath, which makes it possible to define a region as a set of contours.

The remainder of this paper is organized as follows. Section 2 describes related works. Section 3 presents the motivations of this work and explains the benefits of the SuperPath approach. Then, Section 4 proposes a syntax to use the SuperPath extension with SVG. Section 5 evaluates the proposal; finally, Section 6 concludes this paper and indicates future work.

II. RELATED WORK

The following works and technologies described in this section are related to our proposal.

In [6], Gangnet & al. present the benefits of representing a 2D graphic by a structure where all the different regions don't share anything except the borders. They call this structure a planar map. The advantage of such representations is a complete knowledge of the topology of a drawing, enabling powerful processing and rendering. However, the main drawback of this approach is its complex data structure, not suited for usage on mobile devices. Moreover, such a structure is not well suited to our scenario, because the concepts used are too far from those available in SVG or MPEG-4 BIFS.

In a compromise between the planar map representation and the contour representation, the Flash format [5] defines Shape objects. Shape objects are the smallest graphical entity that can be manipulated: either for transformations or for user interaction. The content of a Shape object is a sequence of lines and curves, for which a color to the left, a stroke color and a color to the right are defined. This representation enables a very efficient conversion to bitmap and solves some of the problems mentioned in this paper (e.g. antialiasing and format compactness). On the other hand, there is no notion of region within a Shape, and points within a Shape cannot be reused in another Shape. It is therefore quite difficult to maintain the semantics of original regions in Flash content.

In [4], Dailey proposes useful new elements in the SVG language independently of our work. One of them has some aspects in common with our approach. A major advantage of our proposal compared to this work is to preserve structural information in a simple way, closer to the SVG design. We have furthermore followed a language-agnostic approach, applicable to other languages.

III. RATIONALE OF SUPERPATH

In this section, we will first present the interest of such a primitive through an example. We then highlight a series of benefits of its adoption.

A. The map example

Fig. 1 represents some of the departments and regions of France. The regions are composed of several departments, thus the borders of each department belong to two polygons, those in neighboring departments, and also to the polygons that define the regions. Finally, it is common that a department border is defined by the course of a river. Some contour chunks will be redefined up to five times in the same way: two for contours of the departments, two for the contours of a region, one for the course of a river.

Figure 1. A map with reuse of some geometric elements

Such a representation is common for a map; two neighboring contours commonly share a piece of contour. Existing languages for 2D vector graphics such as SVG cannot currently express this relationship. This leads to problems in several applications when editing, adapting or rendering graphics. The following section details the benefits of using a more adequate representation.

B. Benefits of SuperPath

Benefit 1 – To enable coherent adaptation

If the SVG file contains the information that two chunks of contours are the same, both contours can be adapted in a coherent manner, processing it only once (Fig. 2) and avoiding artifacts produced by divergent adaptations (Fig. 3). Here, a point in the middle of the red diagonal is moved to the left on the grid while the corresponding points on the blue and green path don't move, leaving a hole between them.

Xiaofeng [9] presents a detailed paper about how to adapt some vector graphics to reduce their complexity. Kazi-Aoul [7] presents the most common reasons to adapt a multimedia document to a particular context.

Figure 2 Coherent adaptation: two neighboring polygons share a border part; when adapting that part, the adaptation must be coherent

Figure 3 Artifact due to divergent adaptation of the diagonal on different contours; by rounding a coordinate in the red polygon, we move a point and we create a hole which can be avoided if we know that polygons red, green and blue share some parts

Benefit 2 – To facilitate editing the contours

If the SVG file contains the information that two chunks of contours are the same, an editor can suggest to the user either to modify both in the same manner, or to make different modifications.

Benefit 3 – To manage the anti-aliasing problem

The rendering model of SVG, commonly shared by other technologies, presents a problem with regards to anti-aliasing as illustrated in Fig. 4. The SVG rendering model assumes that drawing on the canvas follows the painter's algorithm, i.e. objects encountered first in the document are drawn first, and objects following in the document are rendered on top. Imagine that a black triangle is rendered first against a white background. Its border will be anti-aliased using a mix of white and black giving some grey pixels. Then, when another black triangle is rendered, sharing a side of the previous triangle, its borders will be anti-aliased as a mix of black and previously rendered grey pixels. Hence, instead of having a mix between black and black, which is black, the result is a mix between black and grey which is dark grey. So, some pixels on the border between both triangles are brighter than they should be. In such a case, knowing that the border is common between both black triangles would make it possible to produce a better anti-aliased result.

Figure 4 Two black triangles (a) with common anti-aliasing problem (b) and without (thanks to SuperPath)

Benefit 4 – To reduce the file size

For a lot of vector graphics, among which maps and cartoon drawings, the main part of the data resides in the representation of the points that define the contours. When a group of points is duplicated, the file size is unnecessarily increased. Using a single representation of duplicated group of points will therefore reduce the file size.

Holes are contours inside a contour which define a region excluded from the main contour (and presumably transparent). Using a better representation of duplicated chunks also allows for better representation of holes.

Figure 5 Representing holes always implies reusing parts of paths

Fig. 5 shows two ways to get a blue shape containing a green shape containing a hole which lets you see what is behind the green and blue shapes. In the first way (first line of the figure), the green shape shares its external contour with the blue shape. In the second way (second line in the figure), the green and blue shapes share the description of the hole.

Benefit 5 – To optimize the display

A player that knows that two chunks of contours are the same can optimize the rendering operations, for example performing the linearization of Bezier curves into straight segments only once.

More generally, it is useful and simple to preserve the semantics of chunk identity for various processing applications covering a wide range of scenarios.

IV. SUPERPATH IN SVG

This section presents the solution offered by SuperPath to the aforementioned problems and then details the proposed syntax for the SVG language.

A. Principles

As mentioned above our approach is language-agnostic and can be applied to any vector graphic format using a contour-based representation of objects. To add SuperPath, there are generic principles to follow for these formats.

Defining the chunks

We need to introduce the concept of chunks using the target language model. As a chunk can be used by different contours with different display styles, a chunk must not be dependent on a style. The chunk therefore only contains geometrical data (points and control points).

Using chunks references

We need to be able to reuse these chunks in different shapes. This is usually achieved through an explicit or

implicit identifier for the chunk and further referencing in each shape which uses this chunk. Obviously, if the targeted language has syntax to express references, we need to be coherent with that syntax for chunk referencing (e.g., SVG xlink or BIFS DEF/USE mechanisms).

Reversing the chunk order

As illustrated in Fig. 6, duplicated chunks could be used as they are first defined or in the reverse order. The syntax must enable such usage.

Figure 6 Three shapes and the sequence of points along each contour

Defining chunk joins

We need to define how two successive chunks of a path are joined. The simplest way is to draw a line between the end of a chunk and the beginning of the next.

B. Proposed syntax for SVG

In this section, we present a syntax of SuperPath for the SVG language which is compact and easy to author. We propose the following new SVG elements:

- `<superpath>`: this element has no specific attribute. It supports traditional SVG attributes such as `id`, `transform`, `class`... It is used as a container element for the elements below and defines its own styling.
- `<subpath>`: this element declares a chunk used in a SuperPath. Its syntax is the same as the standard SVG `<path>` element, but the semantics are different. This element is only displayed when it is a child of a `<superpath>` element.

A subpath can have two forms:

- It can contain a `d` attribute, analog to the one in the path element; if it has an `id` attribute, then the `d` part can be reused by referencing another SuperPath;
- It can contain an `xlink:href` attribute which references the `id` of another subpath. In this case, its geometry is defined by the `d` attribute of the referenced subpath.

In addition, the geometry of a subpath can be used in the reverse order, as explained in 4.A. An “order” attribute is therefore defined with possible values ‘direct’ or ‘reverse’; the ‘direct’ value is the default.

In both forms, the “cmd” attribute expresses how the subpath is joined with the previous one. There is currently only one possible value “line”, indicating that a straight line is used to join the chunks.

Please note that the definition of a chunk can be similar to the definition of the corresponding part of the original content. An overhead will come from the size of the references to the chunk.

As an example, Fig. 4 can be represented as follows:

```
<superpath id="redpath" fill="red">
<subpath id="sp1" d="M...." />
<subpath id="sp2" d="...." />
<subpath id="sp3" d="...." />
</superpath>
<superpath id="greenpath" fill="green">
<subpath id="sp4" d="M...." />
<subpath id="sp5" d="...." />
<subpath xlink:href="#sp1" />
</superpath>
<superpath id="bluepath" fill="blue">
<subpath xlink:href="#sp2" />
<subpath id="sp6" d="...." />
<subpath xlink:href="#sp4"
order="reverse" />
</superpath>
```

The subpath with `id sp1` is the one shared by the green and red shapes; the subpath with `id sp2` is shared by the red and blue shapes; the subpath with `id sp4` is shared by the green and blue shapes.

C. Evaluation

As explained previously, the SuperPath approach provides great advantages in many cases (adaptation, interactivity). We must however check the compactness of the proposed representation.

We have implemented a program which takes an SVG file, finds all the `<path>` elements, computes all reused

chunks and then produces an SVG file extended with the `superpath` /`subpath` elements (the process could be done in a SVG renderer, but the CPU cost will be repeated for each rendering and the semantic not apparent in the file) .

We can note that when the original content has no path or when no path chunk is reused, our algorithm does not modify the content. For other cases, our algorithm produces files in which semantic information about regions is restored, thanks to `<superpath>`. We present below some results for relevant sample files.

Table I presents significant characteristics of our samples. As we can see in Table 1, these samples (representative of our tests) are mainly composed of path data. Therefore, a better representation of path data is likely to result in a reduction of size. File ‘mapfrance’ is a map. File ‘toon1’ is a step in a cartoon animation.

Table I. Test set

Filename	mapfrance	Toon1
File size (KB)	119	103
Number of SVG paths	336	50
Number of points	9452	21179
Number of duplicated points	5434	12961
Cumulated path size (KB)	81	101
Path data/total size	68%	98%

We can see some results in Table II. The first column is the name of the original file. The second column is the size in kilobytes of that file. The third column is the size of the generated file using the `<superpath>` element.

The files used are SVG files taken from the web (maps from Wikipedia) or from previous projects (extracts from cartoons) [2][3]).

Table II Size with and without SuperPath

SVG File	Original (KB)	With SuperPath (KB)
AfricaMap	642	571
FranceMap	873	851
Toon1	103	105
GAEcho	102	55
G0005	36	34

As we can see, a reduction in size is observed in much of the test content, and in other cases the increase in size is quite small. GAEcho is a special case to illustrate the case where the whole points are duplicated; in that case, as predictable, the size is halved. The benefit (about size) mainly depend of the number of reused subpath/points. The main result here is: we benefit of a better structuration without cost in size.

The reduction could be really greater if the file were properly built, particularly in maps. As an example, it is common with hand-made maps that the frontier between two regions is present in two `<path>` elements with slight differences due to the authoring tool. In such cases, our algorithm cannot currently determine that the points are semantically identical, and the resulting file size is therefore not as compact as it could be if the content were properly authored.

D. Conclusion

We have shown that a SuperPath feature is quite useful for modern vector graphic formats. We have also described how such a feature could be introduced into these languages, and provided the proposed syntax for the SVG language. This syntax has been tested on several sequences and generally provides a reduction in size.

In future work, we will further improve the structure for better compactness of the representation and investigate other chunk join types. We plan to propose this feature to the W3C SVG Working Group, and to propose a similar feature in MPEG-4 BIFS.

ACKNOWLEDGMENT

This work was supported by the JEMTU project in the *Institut Telecom*.

REFERENCES

- [1] Bulterman A., “Specification and Support of Adaptable Networked Multimedia,” 1993, Multimedia Systems 1(2), 68 - 76.
- [2] Concolato C., Moissinac J.C., Dufourd J.C., “Representing 2D cartoons using SVG,” SMIL Europe 2003
- [3] Concolato C., Dufourd J.C., Moissinac J.C., Dufourd J.C., “Creating and encoding of cartoons using mpeg-4 bifs: methods and results,” IEEE Transactions on Circuits and systems for Video Technology, 2003.
- [4] Dailey D., “Suggestions for additions to the W3C Specification for SVG,” <http://srufaculty.sru.edu/david.dailey/svg/spec.html> 31.3.2010
- [5] Flash File Format, http://www.adobe.com/devnet/swf/pdf/swf_file_format_spec_v10.pdf 31.3.2010
- [6] Gangnet M., Hervé J.C., Pudet T., Van Thong J.M., “Incremental Computation of Planar Maps,” ACM SIGGRAPH Computer Graphics Volume 23 , Issue 3, 1989
- [7] Kazi-Aoul Z, Demeure I., Moissinac J.C., “PAAM : a Web Services Oriented Architecture for the Adaptation of Composed Multimedia Documents,” PCDN 08
- [8] MIF File Format, http://www.directionsmag.com/mapinfo-l/mif/Mif_j.htm 31.3.2010
- [9] Xiaofeng M., DeCarlo D., Stone M., “Abstraction of 2D Shapes in Terms of Parts,” ACM NPAR 2009