Improving Accessibility of Web-Based GIS Applications

Nieves R. Brisaboa, Miguel R. Luaces, José R. Paramá, David Trillo
Databases Laboratory. Department of Computer Science. University of A Coruña
Facultade de Informática, Campus de Elviña, s/n. 15071. A Coruña. Spain
(brisaboa, luaces, parama, trillo)@udc.es

José R. Viqueira
Systems Laboratory. Department of Electronics and Computer Science.
University of Santiago de Compostela
Instituto de Investigaciones Tecnológicas, Campus sur, 15782, Santiago de Compostela, Spain
joserios@usc.es

Abstract

A major problem of Vector Active Map formats such as WebCGM and Scalable Vector Graphics (SVG) is that, in order to display them in most web browsers, either a plug-in has to be installed or an applet has to be downloaded. In this paper, a web service is presented whose functionality enables the transformation of vector active maps from SVG to a new DHTML (HTML + JavaScript) Active Map Representation, improving this way the accessibility of web-based GIS applications. This new representation, which is also part of the present work, includes a raster representation of the map and a vector representation of its geographic objects. The former is used as a background image of the map whereas the latter enables the response to user-triggered events. An R-tree spatial index structure is used to access the geographic objects affected by each event in order to execute the relevant action.

1. Introduction

The research efforts undertaken during the last decade in the areas of Geographic Information Systems (GIS) and the World Wide Web (WWW), have lead to the development of new technologies that enable the publishing of maps with geographic information in the Web. Consequently, many current GIS applications are directly accessible through the Internet. A good example of the aforementioned developments is the widely used Web Map Service (WMS) specification [5], proposed by the Open Geospatial Consortium Inc. (OGC). A compliant OGC WMS accepts an HTTP map request and responses with the relevant map in either vector or raster format.

Raster map formats, such as PNG, JPG and GIF, can be directly inserted in HTML and displayed in any web browser. However, the activity of these map formats is quite restrictive. In particular, a geographic object contained in a raster map cannot individually respond to user events (mouse movements and clicks) with its own functionality. This limitation is partially solved by the incorporation of an HTML tag <MAP>, however, the visualization properties of an object cannot yet be changed as a response to a user event. For example, it is not possible to change the fill colour of a polygon as a response to a mouse click on it.

The limitations of functionality of raster formats are not present in active vector formats, such as WebCGM [6] and Scalable Vector Graphics (SVG)[7]. A major problem of these formats is that, in general, either plug-ins or applets have to be used to display them. It is well-known that the use of these elements decreases the accessibility of web pages.

In this paper, a new active map representation is proposed that is fully implemented using DHTML and whose functionality supports the common characteristics of SVG active maps. The proposed DHTML code includes a raster representation of the map and a vector representation of each of its geographic objects coded in JavaScript. The former is used as a background image of the map whereas the latter enables the map to respond to user triggered events. An Event Controller, implemented in JavaScript, makes use of an R-tree spatial index structure to access the geographic objects affected by each mouse event, in order to execute the relevant action (which is also coded in JavaScript). Finally, a web service was developed that transforms active maps from SVG to the new DHTML representation.

The remainder of this paper is organized as follows. In Section 2 the new DHTML Active Map Representation is presented. The JavaScript Event Controller is described in Section 3. Section 4 is devoted to the architecture of the web service developed. Design decisions as well as the advantages and drawback of the present approach are discussed in Section 5. Finally, conclusions and pieces of further work are identified the last section.

2. DHTML Active Map Representation

A DHTML active map representation is described in this section that supports the subset of SVG that is typically used to model active maps. Such subset is outlined below.

```
<svg viewbox="0 0 20 20" width="10px"
                         height ="10px">
  <script type="text/ECMAScript">
   <![CDATA[
    function change_colour(evt){
      var target= evt.target;
      target.setAttribute("fill","white");
    }
   ]]>
  </script>

 <g stroke="black" fill="silver"
    onclick="change_colour(evt)">
  <polygon points = "1 11 6 1 11 6.5 19 8
                    10 18">
 </g>
</svg>
```

Figure 1. Active Map in SVG Format.

Active Maps in SVG: For the purposes of the present paper, an Active Map is a collection of geographic objects (points, lines, and surfaces), each of them with: i) a visualization style (outline and fill colours and patterns, icons, etc.) and ii) a behaviour, expressed in some client scripting language (ECMAScript, JavaScript, etc.), which enables the map to respond to events such as mouse movements or mouse clicks. An active map in SVG format [7] is an XML document whose tags are used to describe the properties of the included geographic objects. The SVG document in Figure 1 represents a map that contains a single surface. The <svg> tag includes the geographic coordinates of the minimum bounding rectangle (MBR) that surrounds the map (viewbox in SVG), as well as the dimension in pixels of the drawing area. Tags <g> are containers used to group geographic objects that share some attributes, either visualization styles or behaviour[1] Visualization styles are defined as tag attributes (see *stroke = "black"* and *fill = "silver"* in Figure 1). Tag attributes are also used to link a function written in some scripting language to a given event, therefore defining the behaviour of the objects included in the tag. Thus, in the map of Figure 1 it is defined that whenever a mouse click is done over some object, the function *change_colour* is used to set its fill colour to "white". Finally, various tags are supported to define the coordinates of relevant graphic elements, such as *rectangles, circles, polylines, polygons*, etc. (see tag <polygon> in Figure 1).

[1]Notice that in GIS a map is typically composed of layers, each of them consisting of objects of the same class (rivers, roads, municipalities, etc.) and with the same visualization style and behaviour.

In order to support the above maps in the new DHTML representation, a basic required preliminary functionality is the rendering of vector elements in HTML. Such basic functionality is obtained from the Vector Graphics Library implemented by Walter Zorn [2].

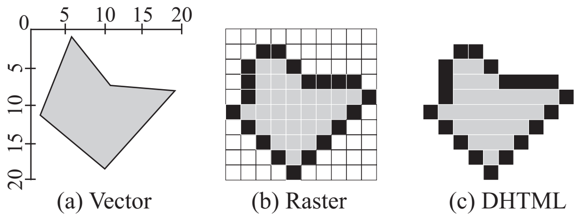

Figure 2. Map Representations.

Walter Zorn's JavaScript Vector Graphics Library: This library contains a collection of JavaScript functions that enable the rendering of graphic elements (ellipses, lines, polygons, etc.) in HTML, making use of well known algorithms [3]. Roughly speaking, each pixel or set of pixels of the raster representation of the input object is rendered as an HTML <div> element. To illustrate this, consider again the SVG map defined in Figure 1. This map contains one polygon, whose vector geometric representation is shown in Figure 2(a). The raster representation of this polygon is shown in Figure 2(b). Figure 2(c) shows the HTML <div> elements generated when the polygon is rendered with the *JavaScript Vector Graphics Library* of Walter Zorn.

Rendering all geographic objects in map with HTML <div> elements is too inefficient. To solve this, a background renderization of the map in raster format is included in the representation. The drawing capabilities of the above library are only used to change the visualization properties of the objects as a result of some user event. A more precise description of each part of the developed representation follows.

Coordinate Manager: It is a JavaScript object that records the geographic coordinates of the MBR of the map and the dimensions of the client drawing area for the map.

Raster Representation: A raster of the map, included in HTML as an image (jpg, png, gif, etc.).

Vector Objects: A JavaScript array consisting of one JavaScript object for each geographic object of the map. Each such object contains as different attributes: the vector representation of the relevant geographic object, its visualization properties, and the name of the JavaScript function to be executed for each possible user event.

Event Functions: The source code of the collection of JavaScript functions referenced by the JavaScript objects above. Notice that the code of these functions may change the attributes of the above objects. The render functionality of Water Zorn's Libray is used to show these changes in the web browser.

Spatial Access Structure: The JavaScript code of an R-tree index structure constructed from the vector representation of the geographic objects of the map. This R-tree is used by the JavaScript Event Controller, described in the following section, to efficiently locate the JavaScript objects affected by each mouse event, in order to execute the appropriate JavaScript functions.

3. JavaScript Event Controller

The JavaScript Event Controller described in this section enables the execution of the activity defined in the JavaScript functions of the DHTML representation of the previous section. Its functionality obeys the following pseudocode.

```
while (true)                                   (1)
begin                                          (2)
    event = captureMouseEvent()                (3)
    screenXY = obtainPointerLocation()         (4)
    mapXY = transform(screenXY)                (5)
    rough = rtreeSearch(rtree, mapXY)          (6)
    for each geo in rough do                   (7)
    begin                                      (8)
        if contains(geo, mapXY) then           (9)
        begin                                  (10)
        responseToEvent(geo, event)            (11)
        if geo.changedStyle then               (12)
            drawGeometry(geo)                  (13)
        end if                                 (14)
    end for                                    (15)
end while                                      (16)
```

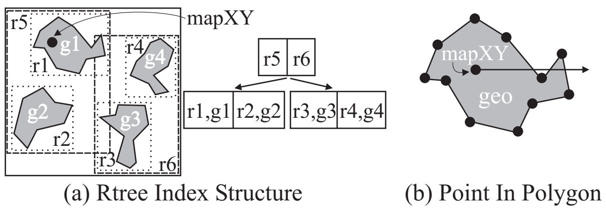

Figure 3. Obtaining Affected Geographic Objects.

In line (3) the algorithm captures a triggered mouse event (mouseMove, mouseUp, MouseDown, MouseClick, etc.). Then in lines (4-5), the position of the mouse pointer is obtained and transformed to the geographic coordinates of the map. The R-tree available in the map representation is searched in line (6) in order to obtain a collection of JavaScript objects potentially containing the location of the mouse pointer. As an example, let us consider the R-tree in Figure 3(a). At the root node, the search algorithm detects that the mouse pointer (*mapXY*) is contained in rectangle r5 and continues the search in the left children. Given that *mapXY* is also contained in r1, then object g1 is retrieved as a potential result. For each object obtained from the R-tree, the algorithm checks whether the mouse pointer is really inside its geometry (line 9). To achieve this, well-known algorithms from computational geometry are applied [4]. A good example of one such algorithm checks whether a point is contained in a polygon (see Figure 3(b)). Generally, point *mapXY* is contained in a polygon *geo* if a half-infinite line segment, drawn from *mapXY* either to the right or to the left, intersects an odd number of segments of the boundary of *geo*. If the above check returns true, then in line (11) the JavaScript function associated to the processed event in object *geo* is called. Finally, if the style attributes of the object were modified during the execution of the aforementioned JavaScript function, then the functionality of the *Walter Zorn's Vector Graphics Library* is used to render again the object in the web browser.

4. SVGTODHTML Web Service Architecture

In order to display SVG maps obtained from standard WMSs, a web service was developed as part of the present work that enables the transformation of such maps from SVG to the *DHTML Active Map Representation* presented in Section 2. Besides, the *JavaScript Event Controller* described in Section 3 may also be obtained from the web service. The architecture of the SVG to DHTML Web Service is shown in Figure 4.

The *GIS Web Application* accesses the functionality of the *SVGTODHTML* web service through a POST request that includes an XML document. The XML Schema of this *XML Request* is shown is Figure 5(a). As it is shown in the figure, the request may include either a SVG document (element *svg*) or reference a URL (element *url*), where a SVG document is ready to be downloaded[2]. Optionally, the request may also include a response mode (element *responseMode*) whose possible values are the following: i) *dhtml*: The service returns only the *DHTML Active Map Representation* (see Section 2) generated for the SVG map either included (element *svg*) or referenced (element *url*) in

[2]This element may be used to include the URL of a *GetMap* request to a WMS.

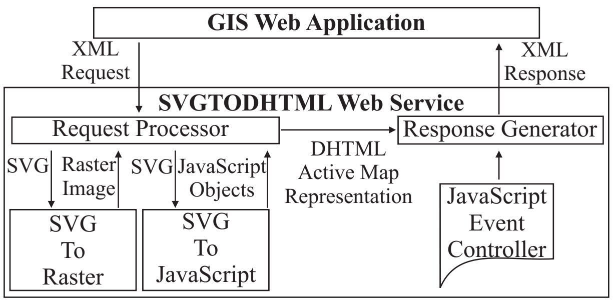

Figure 4. SVGTODHTML Web Service Architecture.

the request. ii) *jsEventController*: The service returns only the source code of the *JavaScript Event Controller* (see section 3), required to display the generated map in the web browser. Neither *svg* nor *url* elements have to be included in this case. iii) *all*: The service returns both the *DHTML Active Map Representation* and the *JavaScript Event Controller*. This is also the default option when the response mode is not included in the request.

The XML Schema of the *XML response* generated by the web service is given in Figure 5(b). This response consists of two optional elements: i) *jsEventController*: It includes the source code of the *JavaScript Event Controller*. ii) *dhtml*: It includes the source code of the generated *DHTML Active Map Representation*. This element is composed of other two elements. Element *raster* includes the raster image of the DHTML representation and element *javaScript* includes the JavaScript code of the DHTML representation.

A brief description of each module follows.

Request Processor: The request sent by the client application is parsed and processed by this module. In the simplest case, if the response mode *jsEventController* is specified, then the *Response Generator* is called to include only the *JavaScript Event Controller* in the response. Otherwise, the *DHML Active Map Representation* must be generated from the SVG document of the request. To achieve this, the *SVG To Raster* module is used to generate the raster image and the *SVG To JavaScript* module is used to generate the required JavaScript code.

SVG To Raster: It renders the SVG obtained from the request in a raster image. The required functionality is supplied by the Open Source tool Batik [1].

SVG To JavaScript: It processes the SVG document obtained from the request and generates the JavaScript code of the *DHTML Active Map Representation*. More precisely, first the JavaScript code of the object that records the geographic coordinates of the MBR of the map and the dimensions of the client drawing area is generated from the <svg> tag of the SVG document. Next, each scripting

```
<?xml version="1.0"?>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="svgToDhtmlRequest">
  <xs:complexType>
   <xs:sequence>
     <xs:choice minOccurs="0">
      <xs:element name="svg" type="xs:string"/>
      <xs:element name="url" type="xs:anyURI"/>
     </xs:choice>
     <xs:element name="requestMode" minOccurs="0">
      <xs:simpleType>
       <xs:restriction base="xs:string">
        <xs:enumeration value="dhtml"/>
        <xs:enumeration value="jsEventController"/>
        <xs:enumeration value="all"/>
       </xs:restriction>
      </xs:simpleType>
     </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

(a) Request XML Scheme

```
<?xml version="1.0"?>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name = "svgToDhtmlResponse">
  <xs:complexType>
   <xs:sequence>
    <xs:element name = "jsEventController"
                type = "xs:string"
                minOccurs = "0"/>
    <xs:element name = "dhtml" minOccurs = "0">
      <xs:complexType>
       <xs:sequence>
         <xs:element name = "raster"
                     type = "xs:base64Binary"/>
         <xs:element name = "javascript"
                     type = "xs:string"/>
       </xs:sequence>
      </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

(b) ResponseXML Scheme

Figure 5. Request and Response Schema.

function contained in the SVG document is transformed to a JavaScript function, with identical name, in the result JavaScript code. Then, the containers (<g> tags) of the input SVG document are recursively processed to obtain a collection of objects, each of them recording: i) geographic coordinates of the relevant geographic object, ii) visualization properties and iii) the name of the JavaScript function to be called for each possible event. Next, the above collection of objects is processed to generate the JavaScript code of the array that will contain the relevant JavaScript objects. Notice that during this process, the geographic coordinates of each object have to be transformed from the geographic coordinate system of the map to the coordinate system of the drawing area. For this purpose the information recorded in the JavaScript object generated in step 1 is required. Finally, the obtained objects are processed again to generate the JavaScript code of the relevant R-tree index structure.

Response Generator: It constructs the XML Response from the *DHTML Active Map Representation* generated by the *Request Processor* and the source code of the *JavaScript Event Controller*.

5. Discussion

The advanced functionality of maps in *Active Vector Represenations* were already discussed in Section 1. It was also stated that a major problem of these maps is that they require the use of either plug-ins or applets to be displayed in a web browser[3]. Plug-ins are the most efficient way to extend the functionality of a web browser. However they have two important drawbacks: i) Due to potential problems of security, they have to be installed by a user with administration rights and ii) they are dependent of a particular web browser. On the other hand, applets solve the security and platform dependence problems of plug-ins. However, they have to be downloaded every time the host web page is visited, which leads to efficiency problems. Besides, in some platforms, such as Windows 2003 Server, the Java Virtual Machine required to execute applets is not natively incorporated.

An opposite approach in terms of functionality is the use of a *raster representation*, which can be easily be displayed in any web browser. Geographic objects of these maps are not enabled to respond to mouse event. The use of the HTML <MAP> tag may incorporate a certain degree of activity in raster maps, however, such activity is still far from the functionality of SVG (for example, it is not possible to change the visualization properties of geographic objects).

The *DHTML Active Map Representation* proposed in the present paper approximates the functionality of an active vector representation with the accessibility properties of a raster representation. In particular, contrary to active vector representations, it can be directly displayed in any web browser with neither the need to install a plug-in nor the need to download an applet. Furthermore, contrary to raster representations, it enables the incorporation of active geographic objects that respond to mouse events by the execution of client script functions, enabling the change of the visualization properties of these objects.

Disadvantages of the *DHTML Active Map Representation*, in comparison with an active vector representation are the following. First, the response to mouse events is much faster if vector representations are displayed using plug-ins or applets. Second, the whole functionality of formats such as SVG cannot be supported in the present approach. For instance, many of the style properties that may be defined for an object are not supported by the present implementation. Finally, the present approach would turn to be completely useless if SVG becomes a standard supported natively by any web browser. A prototype implementation of the web service described in section 4 was already completed. To the best of these authors knowledge, no other DHTML Active Map Representation can be found in the literature with characteristics similar to those of the present one.

[3]An exception to the above rule are the web browser Mozilla and the SVG format.

6. Conclusions and Further Work

A new *DHTML Active Map Representation* is proposed that enables the implementation of the functionality commonly included in SVG maps. A *JavaScript Event Controller*, also part of the present work, enables the display of the present representation in any web browser. Therefore, neither plug-ins have to be installed nor applets have to be downloaded. A web service was also developed that enables the transformation of maps from SVG format to the new DHTML representation. Pieces of further research include the following: i) To choose the spatial indexing structure that better fits the proposed approach. ii) To optimize the Event Controller implementation, to avoid rendering the changes of an object when those changes return the object to its initial state. iii) To improve the rendering algorithms of the Vector Graphics Library provided by Walter Zorn, by minimizing the number of <div> elements. iv) To incorporate all the event types of SVG to the developed prototype.

Acknowledgement

This work was partially granted by MCYT (PGE and FEDER) ref. TIC2003-06593.

References

[1] Batik svg toolkit home page. Retrieved March 2005 from http://xml.apache.org/batik/.

[2] Walter zorn's vector graphics library home page. Retrieved March 2005 from: http://www.walterzorn.com/.

[3] J. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965.

[4] A. Fabri, G.-J. Giezeman, L. Kettner, S. Schirra, and S. Schonherr. On the design of cgal a computational geometry algorithms library. *Software Practice and Experience*, 30:1167–1202, 2000.

[5] Open Geospatial Consortium (OGC). *Web Map Service (WMS). 04-024. Version 1.3*, 2004. Retrieved April 2005 from: http://www.opengeospatial.org.

[6] World Wide Web Consortium (W3C). *WebCGM 1.0 Second Release*, 2001. Retrieved April 2005 from: http://www.w3.org/TR/REC-WebCGM/.

[7] World Wide Web Consortium (W3C). *Scalable Vector Graphics (SVG) 1.1 Specification*, 2003. Retrieved April 2005 from: http://www.w3.org/TR/SVG11/.