SVG 的发展与应用

肖 昕 （重庆邮电学院传媒艺术学院 400065）

摘要：文章总结了 SVG 的特点、功能、现状、应用及 svg 未来发展方向。

关键词：SVG XML 矢量图形

1 引言

可伸缩矢量图形格式 SVG（Scalable Vector Graphic）是 W3C 组织为适应 Internet Web 应用飞速发展的需要而制定的一套基于 XML 语言的用于描述矢量图形、图像的标准规范（该标准于 2000 年 8 月 2 日正式公布）。它是一个工业标准，不属于任何组织和个人。

由于 SVG 是一种完全开放的二维矢量数据格式，并得到众多国际知名软件厂商（尤其是微软和网景公司）的支持，然而由于空间数据结构复杂，数据种类繁多，因此，在 WebGIS 中，若将地理空间数据编码成 SVG 格式来进行空间数据存储、传输和表现，则会有效地消除针对现有专有空间数据格式所产生的数据传播中的问题，是当前研究的一个热点。

2 SVG 的具体应用

2.1 如何得到数据及如何表现动态效果

因 SVG 最重要的特性是，图象可以根据实时数据动态改变，并且实现起来简单，响应迅速，但 SVG 仅仅提供了可以动态改变图象的机制，它本身并不能完成与数据库连接、取得实时数据的功能。如何将 SVG 页面的动态与服务器的数据动态结合起来，在数据实时变化的同时即时展现，是利用 SVG 实现实时信息发布的难点所在。

2.1.1 数据读取问题

使用 JSP 技术，JAVA Server Page 是以用户界面为中心的，标准的动态网页技术. JSP 技术类似于微软的 ASP（Active Server Page），是将 HTML（或 XML）的标签延伸以嵌入程序代码. 由于 JSP 是基于 JAVA 语言的，因而相比于 ASP 而言，具有良好的跨平台性和可移植性. 镶嵌式的 JSP 技术大大简化和方便了网页的设计和修改. 如要显示一个 70×140 的矩形

```
<% int width =70; int height =140; % >
<rect class =" bar2" x =" 100" y =" 150" width ="
<% =width% >" height =" <% =height% >" / >
```

当客户端发出请求后，JSP 在服务器端被解释为 Servlet 并且执行，将结果——即生成的 SVG 页面返回给客户端. 由此可见，JSP 只是通过将网页界面表达与内容分离而加速了 Web 应用的开发过程，实际上是为方便编写而改进了的 Servlet。

2.1.2 实现原理

（1）服务器端：实时数据的读取[4]

在服务器端用 ASP 通过 XMLDOM 生成一个 XML 文件，ASP 每过一个给定时间就从数据库取一次数据，并改 XML 元素的值。这些都可以由 Microsoft XMLDOM 对象来实现

（2）客户端：Web 图象的改变

把 SVG 嵌入 HTML 页中，并在 SVG 源文件中加入如下 JavaScript 语句，用 JavaScript 请求一个带有实时数据的 XML 文件，当 XML 文件到达后，XML 文件被解析，并通过 SVGDOM 和 XMLDOM 的相互作用，把实时数据填入相应的 SVG 元素中，以达到更新图象的目的。其中 XML 文件的解析由 JavaScript 激活 IE 自带的 Microsoft XML 解析器来完成。

2.2 使用 XSL 技术实现 XML 文件到 SVG 文件的转换

采用 XSL 技术将 xml 文件转换为 SVG 文件，优点是修改图形时不需要修改 SVG 源文件，只需要修改 xml 文件，而且可以结合 ASP 或者 JSP 技术动态生成 SVG 文件。具体实现方法是：采用 Xalan – Java2 XSLT 处理器，它完全支持 W3C 的 XSL1.0 版本建议标准和 Xpath1.0 标准，将 XML 文件根据 XSL 样式表文件转换为 svg 格式的文档。

实现方法：

(1) 编辑 XML 和 XSL 源文件 example. xml 和 example. xsl

(2) 利用 Xalan 处理器转换生成 SVG 文件

进入 MS－DOS 界面，转到 example. xml 和 example. xsl 所在目录后，运行以下命令：java org. apache. xalan. xslt. Process －in example. xml －xsl example. xsl －out example. svg，系统自动生成以下 SVG 文件。

2.3 将 SVG 嵌入到网页中

将 SVG 图形对象嵌入到网页中，使用如下 HTML 代码来实现：

<embed width="640" height="560" type="image/svg－xml" id="svgmapctrl" pluginspage="http://www. adobe. com/svg/viewer/install/" src="default. svg" > </embed> 其中 embed 标签指定为一个嵌入的对象，width，height 分别指定该对象的宽度、高度，type 指定类型为 image/svg－xml ，src 指定为 svg 数据文件的 URL 地址，指定这样的标签并在浏览器中打开，浏览器便回调用 SVG Viewer 在指定区域进行显示。此处，src 指定的是一个 svg 文件，在系统中，要求根据不同的请求获取不同的数据，则可以将其改为动态的 url，如服务端的 Java Servlet，由服务端动态生成。

2.4 SVG 在地图方面的运用

2.4.1 地图的显示方法

运用 SVG 显示地图，可以有两种解决方案。一种是将地图的空间数据（如 Shape 格式数据）转换成 SVG，存放在服务器端。客户端请求数据之后，将 SVG 地图下载到客户端，然后在浏览器上进行显示；对于属性数据，可以将与空间数据相关的属性数据转换成文本形式，也下载到客户端。这样，与地图相关的所有数据都下载到客户端，用户可以脱机使用。另一种方案，地图的空间数据的处理和上一种方法一样，而属性数据可以通过客户、服务交互语言从服务器端的属性数据库中实时申请。这种方法的体系结构如图 1。

具体数据显示方法

(1) 点数据。其输出可采用两种方法，一种是通过绘制填充颜色的小矩形来进行，其输出代码如下：<rect id="point1" x="10" y="10" width="2" height="2" style="fill:black"/> 代码的输出结果是在浏览器页面上，画出一个长宽都是 2，填充颜色为黑色的实心小矩形，其左上角点的坐标为(10,10)。另外一种方法是绘制填充颜色的小圆形，其输出代码如下：<circle id="point2" cx="10" cy="10" r="2"/> 代码将在浏览器页面上，显示一个以(10,10)为圆心，半径为 2 的实心小圆，这个半径较小的小圆，在用户看来，就是一个点，其坐标为圆的圆心坐标(10,10)，"point2" 可惟一标识这一个点。

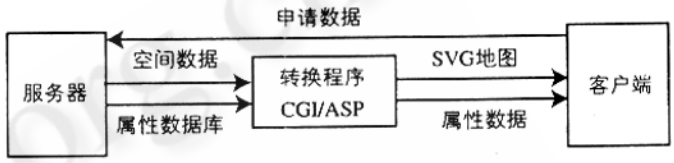

图 1　SVG 体系结构

(2) 弧段数据。弧段是由若干相关或不相关的折线组成，一条折线又是由一组顺序节点组成。根据弧段的特点，可通过输出路径(path)的方式来输出弧段，其输出代码如下：<path id="arc1" d="M1010L1020L2020L3030L3540"/> 代码将点(10,10)、(10,20)、(20,20)、(30,30)、(35,40)相连成一条线，这条线可以被描边，"arc1" 就可以惟一标识这一条线。

(3) 多边形数据。一般多边形是由一条封闭的弧段组成。输出多边形可以像绘制弧段一样来描绘，不同之处在于描绘多边形需多加一个点，而且这个点与起点相同。其输出代码如下：<path id="polygon1" d="M1010L1020L2020L3030L3540" style="fill:black"/> 代码表示节点为(10,10)、(10,20)、(20,20)、(30,30)、(35,40)的多边形，填充颜色为黑色，"polygon1" 可以惟一标识这个多边形。

(4) 字符数据。也就是对地图几何数据一些属性的说明，如名称等。其输出代码可表示为：<text id="t1" x="10" y="10">first text</text> 代码表示在点(10,10)处输出 "first text"。

2.4.2 基于 SVG 的地图浏览器的设计

一、首先根据地物几何形状将地物分为点状实体、线状实体、面状实体、注记体和栅格体。在 SVG 中，对于地物的表示可通过 SVG 中规定的一些基本图形、图像元素来实现，具体来说点状实体通过 use 元素对点

状符号的引用来表示,线状实体和面状实体通过 path 元素来表示,注记体通过 text 元素表示,而栅格体则通过 image 元素表示。

对地图的组织采用分层组织法,其基本思想是将地理空间数据不同的类型划分成若干图层,如道路层、建筑层等,这时层与层之间相互独立,需要时可以将几个层叠加起来进行操作,它的优点是数据结构简单,缺点是不能表达不同层间对象之间的关系。

各层地物的 SVG 实现[5]

(1) 点图层。对于点图层,由于点状地物是通过点状符号类进行表达的,因此,点图层除了需要定义点状符号的颜色外,还要对点状符号本身进行定义。下面就是点图层(路灯符号)的例子。

```
<g id = " Layer Pole ( 0 )" style = " fill : red ; stroke : red ' >
<Symbol >
<g id = " Symobl 0002" transform = " matrix ( 400 400 )" >
<circle style = " fill : none ; stroke width : 0.1 ;" cx = "0" cy = "0" r = ".5" / >
.....
<circle style = " fill : none ; stroke width : 0.1 ;" cx = " 1" cy = "2.5" r = ".5" / >
</g >
</Symobl >
```

(2) 线图层。在 SVG 中,对于传统的线状符号支持不够,目前只能通过线宽、线型(通过 stroke - dasharray 可任意定义)两个参数进行描述。以下是线图层的例子。

```
<g id = " Layer - Road ( 1 )" style = " fill : none ; stroke : red ; stroke - width = 2 ; stroke - dasharray : 15 10" >
```

注意,若在 GML Map 中没有给出图层的 Symbol ID 属性,则不需要指定 Sytle 属性中的 stroke - dasharray 项。

(3) 面图层。在 SVG 中,面状符号可通过 Pattern 元素来实现,由于面状符号是通过点状符号按一定规律进行排列的,因此,面状符号是在点状符号的基础上进行定义的。以下是一个面图层的例子。

```
<g id = " Layer - Build ( 2 )" style = " stroke - width : 0.5 ; stroke : blue ; fill : url ( #pattern1 )" >
<Symbol >
<g id = " Symbol - 0002" transform = " matrix ( 400 400 )" >
<circle style = " fill : none ; stroke - width : 0.1 ;" cx = "0" cy = "0" r = "0.5" / >
</g >
</Symbol >
<defs >
<pattern style = " fill : blue" id = " pattern1" x = " 0" y = "0" width = "40" height = "40" patternContent Uints = " userSpaceOnUse" patternUints = " use- rSpaceOnUse" >
<use xlink : href = " #Symbol - 0002" x = "0" y = "0" / >
<use xlink : href = " #Symbol - 0002" x = "40" y = "40" / >
</pattern >
</defs >
```

注意,若在 GML Map 中没有给出图层的 Symbol ID 属性,则 style 属性的 fill 项直接指定为颜色即可。

(4) 注记层。相对来说,注记层的样式要简单得多,只需指定 style 属性中的 stroke 即可。

(5) 栅格层。栅格层不需要指定任何显示样式。

2.4.3 地图操作功能的实现[6]

(1) 地图的缩放、漫游。地图的缩放、漫游是 SVG 地图浏览器的重要功能,为了达到通过工具条中的按钮来控制地图缩放、漫游的目的,可将所有的地图数据组织在一个地图分组元素下(该元素的 id 属性可设为 Map),即将整个地图作为一个复合图形组来看待。在这种情况下,就可通过设置该分组元素的 transform 属性来实现地图的缩放、漫游。transform 属性包含 6 个参数,分别控制图形的 6 种变形,即横向缩放、纵向缩放、横向倾斜、纵向倾斜、横向移动和纵向移动。其中对于缩放、移位参数的设置可分别通过比例 Scale (Xscale Yscale) 和移位 translate (Xmove, Ymove) 实现。在具体实施时,需注意的是通过比例 Scale 和移位 translate 对 transform 属性的设置效果是一次性的而不是累加的。此外,通过比例 Scale 所进行的地图缩放是以地图的左上角为基点实施的。因此,要想实现对

地图的缩放、漫游就需要对地图同时进行比例 Scale 和移位 translate 操作。

（2）属性数据的查询。属性信息的查询是地图浏览器的一项重要功能，要实现这个功能，首先需要了解属性数据是如何在 SVG 文档中进行组织的。一般来说，属性数据采用两种方法与图形进行连接，即外联法和内嵌法。外联法是指属性数据与图形数据分开存储，SVG 文档中仅包含地物的图形数据，而属性数据存放在服务器端的数据库中，两者通过地物标识号进行连接。当在客户端进行属性数据查询时，客户端可通过地物标识采用 ASP 技术在服务器端从数据库中提取相应的数据在客户端进行显示。而内嵌法则是将属性数据与图形数据包含在同一分组元素中，分组元素中的 id 属性为地物的标识，属性数据通过一自定义元素 GeoAttribute 将各属性项包含在一起，以下就是对一嵌有属性数据的地物描述。

```
<g id = "FA – 01 – 001" onclick = "query('FA – 01 – 001')" >
<GeoAttribute >
<rect x = "5" y = "5" width = "40" height = "40" style = "fill:red" / >
<text x = "10" y = "20" style = "fill: red" > A4 1 </ text >
<Vector Mapm,SVG ,Map BrowserGeoAttribute >
<Price > >34214 </Price >
</GeoAttribute >
</g >
```

在这里需说明的是尽管 SVG 规范中并没有包含对属性数据的标记，但我们可以自行根据需要在 SVG 文档中使用自己的标记，SVG 插件在处理含有这些标记的 SVG 文档时会忽略这些标记，而我们则可在自己的处理程序中对这些标记中的数据进行相应的处理。按照上述的数据组织形式，在宿主页面中即可设计出相应的程序代码来实现图形数据到属性数据的查询。

（3）图层的开关。由于在 SVG 地图浏览器中所操作的 SVG 地图是以分层的方式进行组织的，在该数据组织模式中，属于一个图层的地理特征数据组织在同一个分组元素下，且该分组元素的 ID 属性设置为图层的名称，这样对图层的显示控制就比较容易。首先通过图层名称在 SVGDOM 树中找到对应的分组元素，然后将该元素的 Visibility 属性值设置为 hidden 或 visible，即可达到对图层的关闭与显示。

3 存在的问题

SVG 只能处理二维图形图像，只能进行平面处理，对于三维图形图像的处理就无能为力了，对于三维图形图像的处理就必须借助其他的工具如 CAD、VRML、3DMAX 等软件处理工具进行处理，如何将 SVG 与这些工具有机结合起来对图形图像进行处理是以后研究的一个方向。

参考文献

1 王仲、董欣、陈晓鸥，SVG——一种支持可缩放矢量图形的 Web 浏览语言规范，中国图象图形学报，2000. 12。

2 陈传波、赵婷，SVG 与 XML 的集成技术在动态 Web 图象上的应用，计算机工程与科学，2002. 3。

3 陈传波、王菁、邓凯，基于 SVG 的实时数据动态发布技术的研究，小型微型计算机系统，2002. 5。

4 陈传波、赵 婷，SVG 与 XML 的集成技术在动态 Web 图象上的应用，计算机工程与科学，2002. 3。

5 孙少红、边馥苓，SVG 网络图像标准支持下的地图显示，测绘信息与工程，2002(5)。

6 周文生、胡鹏、贾永红，Web 环境下 SVG 地图浏览器的设计与实现，测绘学院学报，2002 2。