USING SVG TO DELIVER WEB-BASED DATA EDITING

J Ngo and I Miller

Spatial Vision

jess.ngo@spatialvision.com.au
ian.miller@spatialvision.com.au

KEYWORDS: SVG, SPATIAL, EDITING, WEB

ABSTRACT

An increasing requirement in web mapping applications is the ability to directly capture and edit spatial features via the browser. This functionality cannot be done with standard HTML as it does not provide a rich enough user interface, enabling only simple graphic features to be created. There are a number of techniques currently used to provide the richer environment required to enable this functionality, including Java applets, ActiveX plugins and Microsoft VML (Vector Markup Language).

Spatial Vision has previously developed and implemented a product called Tractus which uses the Java applet approach to provide web-based editing. Recently we have been considering the possibilities of using SVG (Scalable Vector Graphics) to deliver this capability. SVG is an open standard for enabling the display of and interaction with vector graphics. Although the common browsers do not (yet) natively support SVG, there is a number of free SVG plug-ins available, including one from Adobe.

SVG is often used to handle the complete rendering of web-based maps, streaming vector data to the browser and symbolising it at the browser end. Our approach was a bit different – the trial involved developing a prototype based on a typical ESRI ArcIMS web mapping application to create the underlying map image but to display that map image as an SVG object. This approach would enable us to apply SVG functionality to provide a rich vector editing environment, with the user creating and editing vector objects (through the use of a zoom, area calculation and pan tool) whilst using the raster map image as a background to provide context.

Our approach involved taking advantage of the scripting facilitated in SVG. JavaScript functions were used to perform tasks associated with each of the functionalities featured in the prototype.

To date, the prototype has proven that this approach is very promising for the enhancement of client-side functionalities namely, vector-based editing tools. It was also demonstrated that a completely integrated solution was possible. Currently, we are continuing the development of SVG-based web mapping applications.
INTRODUCTION

In the past ten years, web-based mapping applications have become ubiquitous, from specialized corporate intranet applications to the many internet web sites which include dynamic mapping. Although the means of displaying and interacting these maps are many and varied, most sites, especially public sites, use a simple web browser displaying prerasterised map images in standard HTML pages.

Such an approach ensures that only a standard web browser is required by the user. Standard web browsers and HTML (even DHTML) enable only limited interaction with such a map image – none the less this is adequate to produce quite a functional mapping interface, supporting common functions such as pan, zoom and query.

In the main, these web mapping applications have been view only. For many public sites this is all that is required (or desirable). In corporate environments, the spatial information used in the web mapping was typically either sourced externally or captured and maintained via skilled users using specialist desktop GIS applications. Indeed for many years, the capture and maintenance of spatial information was considered to be the sole province of skilled CAD or GIS staff.

Whilst few would argue with this for cadastral or other “framework” datasets, there are many cases where spatial data can and should be maintained directly by business users as an integrated part of a business process, most likely including textual information as well. In such a case, an organisation must either train business users to be expert CAD or GIS users or provide an integrated and simplified interface for such spatial data to be entered and maintained by business users without specialist training. To support this approach, web mapping systems must be extended to enable the direct editing of spatial features via the web browser.

EXISTING APPROACHES TO WEB-BASED SPATIAL DATA EDITING

To date, most web-based spatial data editing applications have used one of three technologies; an embedded Java applet, an ActiveX-style plug-in and Microsoft Vector Markup Language (VML).

Browser Plug-ins

The first two of these approaches use the plug-in capability of most browsers to load (and in some cases download) a reasonably large application to perform the editing functions. Whilst capable of delivering powerful editing capabilities, this approach has a number of issues including the size (and therefore time) of the download required and the reluctance of many organisations to support such plug-ins in their standard operating environments (SOE).

Another issue with this approach is that the level of integration between the plug-in and its host web page is typically
limited, leading to lower integration between the textual and spatial components of a business application than is desirable. This is also caused by the fact that the business logic for the plug-ins generally resides in the plug-in code itself, whilst for the host web application, it typically resides on the web application server.

Spatial Vision’s Tractus product for web-based spatial data editing uses a Java applet which runs in a browser Java plug-in. Whilst it provides a high level of usability, it does sit in a separate browser window to the host business application and therefore requires the user to choose to move in and out of the editing environment.

Vector Markup Language

The other approach in common use is Microsoft VML. VML is a markup language similar to HTML but based on XML rather than SGML. VML supports the drawing of vector features such as lines and curves, including a range of display attributes such as different strokes and fills. VML is supported in Microsoft Internet Explorer only although the domination of IE means that this is not an issue in most organisations. VML is embedded in a normal HTML page and VML elements can be dynamically manipulated with scripting languages such as VBScript and Javascript.

Compared to plug-ins, VML enables the editing of spatial data to be more directly integrated into a business application. VML is no longer being actively developed and it remains to be seen if Microsoft (or other partners involved in its development) will further enhance it in the future to keep pace with the evolving web and browser environments.

THE SPATIAL VISION SVG RESEARCH PROJECT

For some time now, Spatial Vision has had a graduate cadet position to which it appoints annually a new graduate selected from an open recruitment process. The graduate cadet is given the opportunity to work across a range of areas within the company and in the second half of their year, undertakes an independent research project. The research project, jointly agreed by the graduate cadet and their Spatial Vision supervisor, is intended to provide the graduate an opportunity to pursue an area of their particular interest and which is also of interest to Spatial Vision.

In 2004, Jess Ngo was awarded the graduate cadet position. Jess had completed a project on Scalable Vector Graphics (SVG) in her final year at RMIT and was interested in furthering this research. Spatial Vision had developed the Tractus component for web-based spatial data editing but had been monitoring the rise in functionality and popularity of SVG and was interested in seeing if it could provide a viable alternative to Java applets for this task.

It was agreed that Jess would undertake research into SVG and its ability to support a web-based spatial data editing capability. If this research showed that such an approach was feasible, Spatial Vision would consider the use of SVG in future projects.
PROJECT OBJECTIVES

Rather than exploring the full extent of SVG’s capabilities by creating a pure SVG solution, the objective for this project is to determine whether it can be used to provide rich interactivity at the browser end. There is no intention to use SVG to format and display the main map layers or features, but rather to have a typical pre-rasterised map image, such as that used in many mapping applications today, displayed within an SVG object which provides rich tools for interacting with it.

In the case of a typical “zoom” tool for example, this would mean being able to dynamically create a rectangular-shaped object “on top of” the static raster image in response to a user clicking and dragging the mouse, in order to define the desired new map extent.

The specific aim of this project is to investigate the capabilities of SVG to enhance the functionality of an existing web mapping application, replacing any use of DHTML and plug-ins. This will involve the development of a prototype, based on an ESRI ArcIMS Image mapservice, to demonstrate a sample of tools common to current applications developed by Spatial Vision. The prototype will enable the user to perform the following functions:

- Zoom in (by drawing a rectangle on the map);
- Pan (by dragging the map image); and
- Calculate areas of polygons (drawn on map by the user).

Note that it was decided not to attempt to implement actual spatial data editing tools since the three simple tools above would demonstrate all of the interface functionality required to support spatial data editing, the only difference being the “back end” functionality which was already well understood and developed for spatial data editing. In essence, we wished to see if SVG would enable us to “swap out” the client-side functionality currently delivered via a mixture of DHTML and a Java applet with SVG.

SCALABLE VECTOR GRAPHICS (SVG)

Since the inception of the Web, raster graphics dominated the Web, being the only format initially natively supported in Web browsers. To overcome this, many workarounds and add-ons have been developed, such as Macromedia’s Flash and Microsoft’s VML. Such approaches have typically been proprietary and have generally required third-party plug-ins and/or specific web browsers.

SVG has been developed to overcome these limitations and has been a W3C (World Wide Web Consortium) recommendation since September 2001 with version 1.1 of the specification released in early 2003. The SVG specification was the result of collaborative efforts from a variety of large-name corporations, including Adobe Systems, Eastman Kodak, Sun Microsystems, Nokia, Hewlett-Packard, Macromedia and Microsoft.

Some issues to consider when using SVG are:

- SVG is a native graphics format that has been specifically designed for delivery over the Web unlike some of the heavier plug-in technologies
- SVG is supported across all popular browsers, via the free Adobe SVG plug-in in Microsoft Internet Explorer and now natively in Mozilla, Firefox and Opera. It is not known if Microsoft will implement native SVG support in Internet Explorer – they have certainly implemented native SVG support in a number of their other applications including Visio.
- SVG code is human-readable, a feature of all XML-based languages. This makes its use easier from a developer perspective but has some commercial implications regarding the protection of intellectual property. There are some means available to protect SVG source code but, especially with dynamically generated SVG, these are not entirely effective.

PROTOTYPE ARCHITECTURE

The SVG prototype was designed to provide a simple test bed to meet the above objectives. It comprised of:

- Web browser (Internet Explorer);
- ArcIMS server to deliver rasterised map images;
• SVG plug-in viewer (Adobe SVG Viewer); and
• Coding languages including SVG, JavaScript, ColdFusion and HTML.

The architecture is shown in Figure 3. An ArcIMS mapserver is used to generate and return map images as pre-rasterised images, based on a map extent based to it from the browser. The browser contains an SVG object which is used to display the map image in the background, allowing a variety of other objects (such as rectangles and dialogue boxes) to be displayed over the map image.

A small amount of Javascript code running in the browser, both within the SVG document and the HTML frames, implements the various tool functions. The Javascript is used to communicate between the frames, implement the dynamic creation of SVG objects such as the zoom rectangle and to format the results of the users drawing and clicks into a request to be passed back to the ArcIMS map server.

Referring to Figure 4, the browser window consisted of the following break-up; a frameset separated into three frames – header, post and map. Each of these frames had a specific purpose:
• header contained all the tools and other relevant buttons;
• post was primarily used to ‘communicate’ with the ArcIMS server; and
• map stored the embedded SVG document.

Javascript provides the communication between the various frames and relies upon SVG’s support of such scripting languages to allow manipulation of elements within the SVG document, as shown in Figure 5 below.
The Zoom Tool developed for this prototype is quite standard in operation. It allows the user to define the area of interest using the mouse, whereby a map showing the new extent is then generated and displayed on-screen (Refer to Figure 6). The Zoom Tool in the prototype utilises the opacity-fill attribute available in SVG to provide an attractive semi-transparent fill for the rectangle.

Referring to the first image of Figure 6, a useful inclusion to the Zoom Tool is the display of the AMG coordinates for X, Y situated in the browser’s status bar. While the mouse is being dragged to form the Zoom box, these X, Y values will continue to be updated, until the mouse button is released.

Figure 6. Zoom-in tool in operation and the resulting map image

AREA CALCULATION

In the prototype, the area calculation tool performs two actions:

- It allows adhoc polygons to be ‘drawn’ on-screen by the user
- The area of the shape drawn is then calculated and the result displayed on-screen

Once the Area Tool is selected, a red text box to display the area becomes visible. To create an adhoc polygon covering the area of interest, the user makes a series of mouse clicks on the map. Similar to the Zoom Tool, the shape depicted on-screen is semi-transparent thus allowing the underlying map features to still be seen (Refer to Figure 7). A double mouse click is required to close off the entered polygon, at which time its area will be calculated and displayed (Refer to bottom-right corner of Figure 7).
Note that calculation of the area including the conversion from image units (ie pixels) to map units is based on translation information returned from ArcIMS.

**PAN TOOL**

In the prototype a conventional pan tool was incorporated. Once the ‘Pan Tool’ button is clicked to make it the active tool, the user is able to click and “drag” the map image to a new position and simultaneously see how far they have panned the map (Refer to Figure 8). This gives the user a good indication of the new map view so once they are satisfied, releasing the mouse button will cause the generation and display of the new map view.

Another common function is an alert message to notify the user that they have attempted to pan beyond the map’s maximum extent. Instead of using the alert message box provided in JavaScript, this was seen as an opportunity to utilise the capabilities of SVG to create a more visually appealing alert (Refer to Figure 9). In this case, Adobe Illustrator was used to design the initial alert message box which was then exported to SVG, a function provided by Illustrator. The resulting SVG code was then taken and modified, extracting the necessary coding and discarding a large amount of irrelevant and code generated by the export process.
CONCLUSIONS

The development of the prototype with each of the tools described above successfully demonstrates that SVG is capable of being used to support and enhance the user’s browser-side interaction in web-based mapping applications. Each of the developed tools in the prototype was at least as functional as their DHTML or Java equivalents and in some cases, provided a more attractive and usable function.

In addition to the usability of the tools themselves, the resulting prototype provided a more integrated solution as no plug-in technology or separate browser windows were required. In summary, SVG is a viable candidate for the development of web-based mapping applications, including those requiring a higher level of user interactivity such as integrated spatial data editing.

Spatial Vision are now considering the use of SVG in upcoming projects.