

Developing an interactive Road-Accident-Information-System using SVG

Master Thesis of Christian Spanring

Ao. Univ. Prof. Mag. Dr. Georg Gartner
Department of Geoinformation and Cartography
TU Vienna

Motivation

Accident research forms the basis for planners to take measures in the general area of transport planning and of course in its special area of road safety planning. The spatial representation of accident research results is often inevitable for planning decisions because important connections and interactions of planning activities are mostly clarified in the spatial visualization. GIS systems represent an integral part of processing spatial distributed data.

Within this work it shall be shown how an highly specialized GIS system can be made accessible to a broader, for the first step primarily planning, public.

As communication platform the internet will be used. Via that platform the accident data (accident accumulation spots) from the austrian road safety board will be published in realtime from the internal specialized GIS system to an online information system.

Considered and used techniques

To publish and visualize cartographic material online there are basically 2 possibilities:

- raster images
- vector images

Raster images (jpeg, gif, etc) have the advantage that they can be displayed at almost every PC without the necessity of installing special software or extensions. Also, compared to vector graphics, the display quality is in some occasions higher. Color graduations can be carried out more easily and often more attractively.

Nevertheless I consider the suitability of raster graphics for cartographic applications, where a high user interaction with the pictorial material is given, as very limited.

Within the use of vector graphics one gets access to every single "real" map element (points, lines and

polygons) and can modify and create their attributes. The use of raster graphics allows the identification of map elements only through the pixel color similarity and pixel position within the graphic. On the other hand one can easily embed raster graphics within vector graphics and thus benefit from the partly higher display quality of raster graphics as mentioned above.

The main disadvantage of vector graphics, especially in the case of online applications, is that there is no native display support in most web browsers. SVG (Scalable Vector Graphics) stands at this point for the main exception. SVG is XML based (one can create graphics by describing them in a simple text editor) and thus its highly suitable for web applications.

Unfortunately one still has to use browser plugins to view SVG and is therefore dependent of partly special implementations of the SVG standard by some plugin manufacturers. A native browser support for SVG is built within the mozilla browser. SVG support is already available in the source code and can be made available during the compilation process of the mozilla browser.

There are some tools available that handle the export of SVG from GIS systems. In this work the export was done by the freely available Avenue script "shp2svg" to produce the appropriate SVG files out of an ESRI ArcView 3.3 project.

The export process delivered about 3 MB of SVG data, which would be considerably too much for an online application. The user would have to download 3 MB of data before he could use the map. For still far common modem users this circumstance would be hardly reasonable and therefore would endanger the acceptance of the application. As well the complete performance of the application would suffer with this quantity of basic data, e.g. every single interaction would be sluggishly going on.

The idea to transfer only the data the user needs was obvious. Basically, database systems offer this functionality - to query, select and submit data.

Theoretically numerous database systems would fulfill that functionality. Therefore criteria for a short list were:

- easy integration within web applications (interface and support of commonly used scripting languages)
- freely available
- support for spatial data

Because of their support for spatial data Oracle Spatial and ESRI ArcSDE (e.g. together with MS SQL Server) were considered, however, left because of a too high expense.

In web applications a very popular and freely available database is MySQL. Version 4.1 has also a built-in support for spatial data. In addition the installation on most systems and, due the availability of numerous administration tools, the configuration are very easy.

However, after completing some tests and comparing the functionality based on the documentation the decision was made in favor of PostgreSQL with the PostGIS spatial data extension. Beside the capability of spatial data storage and indexing, PostGIS allows far-reaching spatial operations. In addition, the

implementation of "OpenGIS - Simple Features Specification for SQL" has progressed further than in MySQL.

The decision to use a database server as source for the SVG output brought a complete change to the concept of connecting the GIS system with the web application. Generally it is possible to use the database as back-end for the GIS system but since the main focus of this work is the web application and there is no out-of-the-box support for linking ESRI ArcView to PostGIS, I didn't took any steps to directly connect the database and the GIS system. As said, basically it's possible and there are a few (open source) projects dealing with that topic. Here, the Avenue script "shp2sql" was used to create some SQL dump files and import them later into the database. The more elegant and in a production environment for sure necessary way would be connecting the GIS system directly to the database, but for this master thesis the effort to do that would be to high.

Since all the basic data now were stored in a database there must be found a way to produce SVG output. For this one can modify the PostGIS installation and add "AsSvg" to the functionality. It would add SVG compliant output for PostGIS. But at the time of that decision "AsSvg" was unknown and so unfortunately it was not considered.

The chosen way was to use a server side scripting language to query and select the data in the database and deliver the geoinformation as well as the attribute data as plain text, in SVG and XML format to the web browser. PHP (Hypertext Preprocessor) was the favored scripting language because of its simplicity for small web applications and its well integration with the (already used) Apache web server.

As development environment served a PentiumIII/800 running the linux distribution Fedora Core 1 and an Apple PowerBook G4/400 running Mac OS 10.3. Due the availability of binary packages by the PostGIS community the installation of PostGIS on Fedora Core 1 was no problem at all. Installing PostGIS on Mac OS X was slightly more difficult because all the binaries had to be compiled from source. Configuring the database and web server was due the availability of very detailed information and documentation no problem on both systems. For coding, the very comfortable and freely available development tool Xcode from Apple was used.

On-going testing during the development was made in the browsers Mozilla-Firefox and Safari on Mac OS X, Internet Explorer 6 on Windows and Mozilla on Fedora Linux. All Systems were equipped with the Adobe SVG Viewer 3 browser plugin.

Example: main features and highlights

The most essential feature of that work is surely the on-the-fly production of SVG code. The spatial operations of the database server are consulted to select only those elements of the map which are within the maps view-box at the time of interaction. SVG properties and attributes are submitted via Javascript to PHP functions, which handle all the database connection and return plain text XML data which is parsed by Javascript again and embedded into the SVG map. Javascript enables the access to every single map element and PHP is responsible for getting the right geoinformation out of the database. Due those operations the complete performance of the map application and therefore the usability were

increased fundamentally.

Another decisive factor for performance and usability is the scale depending smoothing (reducing vertexes) on lines and polygons. By geometry calculations based on the Douglas-Peucker algorithm already on the server, fewer data has to be transmitted to the client (SVG viewer). The step of line smoothing is necessary in that work because very detailed GIS data is used for the output.

For reasons of platform independency the intention was to avoid a problem that has shown up evaluating various cartographic web applications which were using SVG: Embedding several SVG documents (as Objects) in on HTML document may results in problems apart from the Windows platform. Due platform depending implementations of the DOM (Document Object Model) it can lead to problems if several SVG documents try to communicate among each other. This problem was avoided by embedding several SVG documents in on "container" SVG document instead in one HTML document, similar to the concept of frames in HTML. The combination of several SVG documents in different extents and scales is inevitable developing cartographic applications which try to fulfill the approach of using detailed GIS data as source.

Therefore it was ensured that this application is identically working on at least the tested platforms (Windows, Linux, Macintosh) and browsers (Internet Explorer, Mozilla, KHTML/Safari with Adobe SVG Viewer 3). Because of the use of only one "container" SVG document, scaling the whole application to another screen extent is at least theoretically in a few additional working steps possible.

The possibility to use the database server as GIS back-end data source was already mentioned. This would create the possibility of being able to publish all the GIS data through the web front-end without any further working steps, only by storing them in the database out of the connected GIS system.

Beside the commonly used and well known cartographic tools like zoom, pan, show/hide layers, orientation map, scale-bar the application offers the possibility to access the map elements due a attribute-query mask and a (graphical) select tool. Through the select tool all attributes of a certain map element can be invoked and displayed in a system of tables. Due linking of table entries (e.g. spatial operations like "find within extent" of map elements) one can navigate through all map elements and their attributes without leaving the table system.

A built-in search mask allows the user to search/find specific map elements (accident accumulation spots) and displays the search result as high lighted map elements in the map and the table system. By default the most important attributes are shown in the so called Info-bar above the map and as well as a label beside the mouse pointer.

The tools and information areas are ordered in the user interface by their priority form the upper left corner down to the right.