# Developing Lightweight, Data-Driven Exploratory Geo-Visualization Tools for the Web

Erik Steiner, Alan MacEachren, and Diansheng Guo

GeoVISTA Center, Department of Geography, 302 Walker, Penn State University, University Park, PA, USA, Phone: 814-865-7491; Fax: 814-863-7943
Correspondence to: maceachren@psu.edu; www.geovista.psu.edu

## Abstract

This paper details efforts to development a set of prototype, web-based geo-visualisation tools. The focus is on design and implementation of lightweight tools that can run in standard web browsers and access data, stored in a remote database. The web geo-visualisation tools incorporate several standard exploratory spatial data analysis methods, including linked brushing and interactive animation. These tools are constructed using Macromedia Flash, a commercial software application that produces content for the web using a publicly available file specification called SWF. The advantages and disadvantages of Flash for geo-visualisation are discussed. One advantage is that it supports links to remote databases. Accordingly, the use of XML as the communication syntax for the geo-visualisation tool-to-database link is discussed. Two solutions to dynamic database access have been implemented, one using a stand-alone Java server and the other using Java servlet technology; the advantages and disadvantages of each approach are discussed.
**Keywords:** interactive maps, visualisation, exploratory spatial data analysis, remote database access, web mapping

## 1 Introduction

Dramatic advances in 'mapping' have occurred during the past decade. Among the most fundamental are the change from maps that communicate a single message, statically, to maps that support exploration of multiple perspectives, dynamically. Although dynamic maps have been appearing on the WWW (world wide web – hereafter, simply web) for several years, most of the innovation in geo-visualisation during this time has been directed to expert users working on desktop (or more powerful) computers. Technological and societal changes are making it

possible to extend from this base toward development of highly interactive geo-visualisation tools designed to support non-expert needs as well as the web-delivery of such tools.

In this paper, a set of lightweight, data-driven geo-visualisation tools for the web that are being developed as part of a 'Digital Government' (DG) research initiative in the U.S are both described and demonstrated. The focus of our contributions to the wider DG effort is on dynamic maps and graphics to support visual delivery and analysis of federal statistical summaries (compilations of numerical data, such as those from the national Census, that are published in aggregate form).

The initial section below provides some background for both the kinds of exploratory geo-visualisation tools being developed and for the DG project they are a part of. Next, the implementation of a web-based prototype using Macromedia Flash as the development platform is briefly presented. The prototype includes dynamic and linked exploratory geospatial data analysis methods (methods that are now relatively standard for systems targeted at expert analysts but, in the implementation presented here, are designed for use by non-expert users). The subsequent section outlines the mechanisms constructed to support data access from a commercial database (Oracle) through the Flash (map-based) interface. We close with a brief discussion of possible extensions to these tools, some of the limitations encountered, and plans for application of an iterative, human-centred design approach to tool development and assessment.

## 2 Background

Interactive maps for the web have become quite commonplace, and substantial investments in both geospatial research and applications are being directed toward serving of geospatial data via the web; see (MacEachren, 1998) for a recent review. There have also been several successful web-implementations of exploratory geo-visualisation methods; see (Andrienko & Andrienko, 1999; Dykes, 1997; Harrower, MacEachren, & Griffin, 2000). Still, it remains a challenge to design web-based interactive geo-visualisation tools that meet professional cartographic design standards while being flexible enough to achieve the core geo-visualisation goal of supporting a multi-perspective approach to data exploration. This challenge generally requires that developers (and often users) master formal computer programming languages; and even the best of current tools impose serious limitations from the perspective of graphic design – limitations that are likely to impede the transitioning of multi-perspective geo-visualisation methods from expert to non-expert user.

The goal of our particular DG project is to develop 'quality graphics' that support the understanding and flexible analysis of statistical summaries produced by eight federal government agencies in the U.S. The research has a dual focus on provision of relatively sophisticated exploratory data analysis tools to support

expert work in generating these statistical summaries (e.g., to support population projections by the Bureau of the Census) and to support access to derived information by the public (with the target audience here ranging from the ordinary citizen interested in health-environment issues to the university researcher conducting a demographic or economic analysis).

The Flash-enabled tools described below are intended to support this dual focus. First, Flash is being used to provide both an environment for formal cognitive experimentation and for rapid prototyping. The rapid prototyping allows dynamic data exploration methods to be quickly implemented and assessed, before committing resources to final implementation as part of a suite of Java-based tools being constructed within GeoVISTA *Studio* (MacEachren et al., 2001) and/or Illumitek's nViZn. Second, we are experimenting with Flash as a final web-delivery mechanism that supports user-friendly but flexible exploratory geo-visualisation tools that can be used effectively by non-experts.

In sections 3 and 4 below, we describe prototype, Flash-based, web geo-visualisation tools we have implemented using a client-server approach. The first section focuses on design and implementation of interactive exploratory maps that run in standard web browsers and that support dynamic linking between visual representation forms and user construction of temporal and non-temporal animated sequences. Then, the next section details the process of linking these interactive maps dynamically to a remote database.

## 3 Design of Dynamic Maps

We have created several dynamic maps using Flash in an effort to determine capabilities and limitations of this. Here we illustrate, and describe briefly, three of these prototypes.

### 3.1 Linked Geographic Brushing

In our initial prototype, we implemented a choropleth map and scatterplot display using state-level data of the United States. The interface was designed in an earlier version of Flash (4.0) that supported data access only through loading URL Encoded text files. Later prototypes (discussed in sections 3.2 and 3.3) implement more sophisticated database support. With our initial tool, once the data are loaded, they are linked to their geographic entities (states) and to a scatterplot representation. This functionality is illustrated in Fig. 1 (below). The figure shows a screen capture from a custom dynamic application built using Flash's ActionScript (a scripting language that is similar, conceptually, to JavaScript while emphasising graphic applications). Flash's graphic design tools allow the designer to construct the visual elements from which the interface is composed, while ActionScript supports control of the global characteristics of the display objects (colour, size, location, transparency, etc.). In the case of the choropleth

map shown, state elements were given a lightness value that corresponded to their data value, and in the case of the scatterplot, individual points were given positions according to their values on two separate attributes.

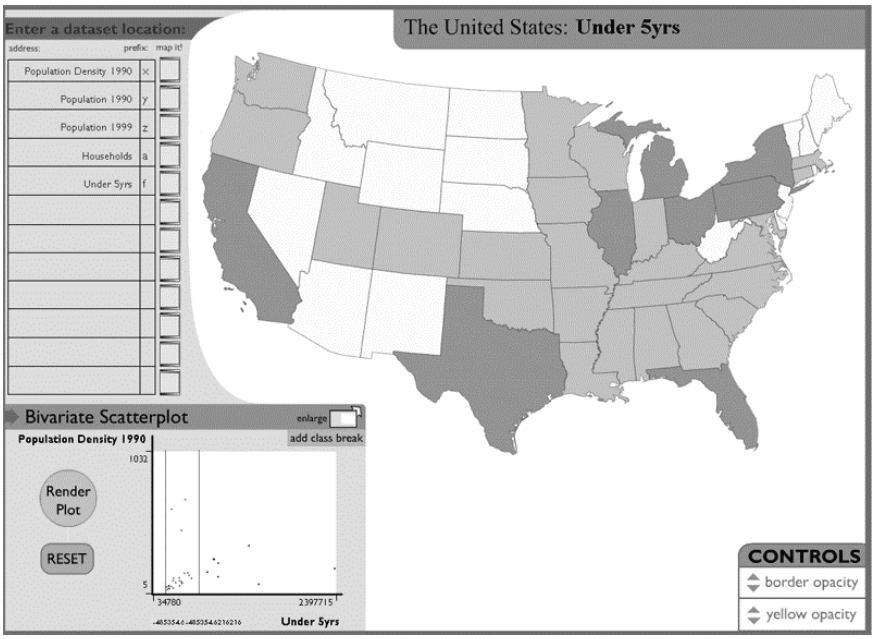


**Fig. 1.** Prototype 1: Linked geographic brushing interface

The prototype demonstrates the concept of linking a scatterplot and choropleth map dynamically, so that manipulation of one has an impact on the other. For previous research on the topic of linked geographic brushing; see (Dykes, 1997; Haug et al., 1997; Monmonier, 1989). Here, we implemented a one-way communication from the scatterplot to the map in which user interactions within the scatterplot are propagated to the map. Specifically, users can define class breaks on the x-axis of the scatterplot (the axis depicting the variable mapped) by dragging a vertical line that represents each break point. These breaks are then immediately propagated to the map display, with the colour assigned to each state altered to reflect its new class assignment. Thus users can define classes based on the characteristics of one variable's relationship to another and evaluate the impact of those class break choices on the spatial pattern of that initial variable. An interactive version of the interface may be accessed at the following address: http://www.geovista.psu.edu/publications/Beijing01/SteinerICA01/loadusa/loadusa5.html.

Our success with the Flash 4.0 environment demonstrated the feasibility of developing linked displays that include lightweight and dynamic statistical maps (the example above involves a 93KB download to the web browser). One

drawback that we encountered in our initial testing was a lack of flexibility related to loading new data into the interface. Flash 4.0 included very limited support for robust data structures. The highly customised nature of the individual components of this first prototype (scatterplot, choropleth map) also constrained the future application of these representation forms to new interfaces.

### 3.2 User-Controlled Animation Sequences

A second prototype interface we designed combines the linked brushing concept with user-controlled animation sequences and more sophisticated data access. The interface is designed to allow users to load and explore county-based thematic data attributes through a cartographic display. Variables are loaded using eXtensible Markup Language (XML) syntax (supported in Flash 5.0 and higher). XML is rapidly becoming a standard information exchange language; it is supported by commercial databases, with efforts to develop a geospatial variant (GML) well under way (Open GIS Consortium, 2000; Zaslavsky, 2000). Support for XML makes it possible for Flash to access a database efficiently. In our case, an Oracle database server stores the relevant datasets (see section 4 for details on database access issues).

The dynamic environment illustrated in Fig. 2 is designed to interpret the structure of an incoming XML file and, subsequently, dynamically construct

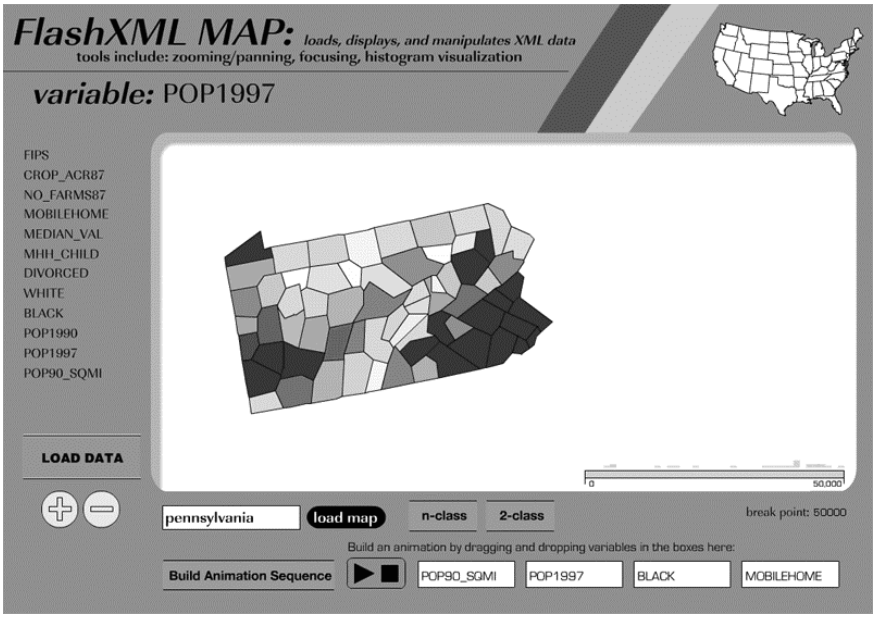


**Fig. 2.** Prototype 2: XML-based map incorporating animation sequences

visual and data elements from this code. In this prototype, once the data are processed by the Flash Player, they are stored locally and may be explored and visualised without further communication with the server.

In prototype 2, three visual representation forms are generated upon data input: a choropleth map, a histogram display, and roll-over buttons for each variable. The number and names of the variables are defined based on the XML metadata. The prototype includes standard zooming and panning functions (for the map) and the three elements of the interface are dynamically linked. For example, if a user clicks or does a 'mouse over' on a variable button, the attribute selection is immediately propagated to the map and histogram; if the user interacts with the histogram, the changes are immediately propagated to the map.

Flash is an object-oriented environment in which each display entity (e.g., county) controls aspects of its own display. Labels for the elements in the variable list (left column of the interface) are generated from metadata within the XML import file. On import, the data associated with each variable are distributed to the geographic entities (e.g., counties). When the user chooses a new variable (by mousing-over its label), each geographic unit now 'knows' to refer to the selected locally stored attribute. Thus, each map object accesses data simultaneously, without referring to the original dataset or making a new call to the server. Similarly, the histogram object recognises the shift to a new variable and will immediately update its form based on the user's selection.

The histogram supports linked geographic brushing in the form of a moveable class break similar to the scatterplot described in the first prototype. The user may click and drag along the base of the histogram display to create a two-class representation of counties above and below a certain class break; see Fig. 3. Users can also define a break point for a single variable and then jump among the different variables to compare the spatial distributions of the variables based on this cut-off criterion. The dynamic interaction between a statistical distribution representation (histogram) and a cartographic representation (choropleth map) may allow users to explore geographic datasets in greater depth with the possibility of generating novel hypotheses.

Users can visualise each individual variable alone or may choose to rapidly shift between variables to compare attribute values and distributions, an example of non-temporal animation. For more information on non-temporal animation, see: (DiBiase, *et al*., 1992; Egbert & Slocum, 1992; Peterson, 1993). Building such a sequence is accomplished by dragging variable labels from the list in the left margin to the slots in the “build animation sequence” row. As the animation is run, users can interact with the histogram display to produce a dynamic brushing effect with a single class break that is transmitted to each variable in the animated sequence. The same tools will work for temporal data sequences, although the drag-and-drop method for building sequences is not practical for long time sequences. In such cases, supporting drag-and drop for sequence end points (with other data accessed automatically) may be practical.

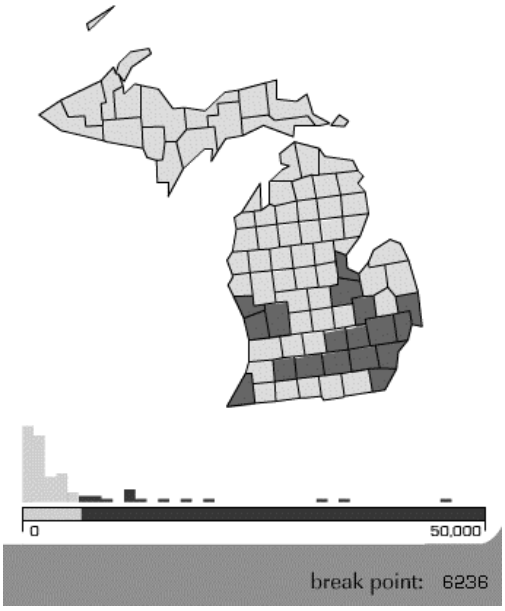


**Fig. 3.** Prototype 2: Histogram-Map link with draggable class break

This prototype demonstrates the feasibility of using dynamically-loaded XML files to create dynamic visual representation forms using Flash. While the interface remained lightweight (114KB), the data loading process proved to be a performance bottleneck (taking perhaps 5 seconds to load Pennsylvania). Once the data were loaded, however, the system response was immediate and satisfactory.

### 3.3 U.S. by County - Interacting with Detailed Maps

The first two prototypes described, dealt with both a limited number of data variables and limited number of geographic entities. To further explore the capabilities and limitations of Flash as an environment for web-based geo-visualisation (particularly in support of the missions of our Digital Government partner agencies), we created a new application for county-level data of the entire United States.

Generating a base map of the U.S. to accept the XML input was a labour-intensive process. As explained in more detail below, it involves manual labelling of each geographic entity (more than 3000 here). Once completed, of course, the map is reusable.

While we streamlined the previous ActionScript code to perform only absolutely necessary operations on the dataset, there was a substantial improvement of system performance with this county-based map. An XML file

containing “fips” codes (a standardised coding scheme used in the U.S.) with a single attribute took about one minute to load and the system response once loaded was unsatisfactory for linked brushing and animation. This is a recent prototype and we are currently evaluating the application in an effort to speed up the processing.

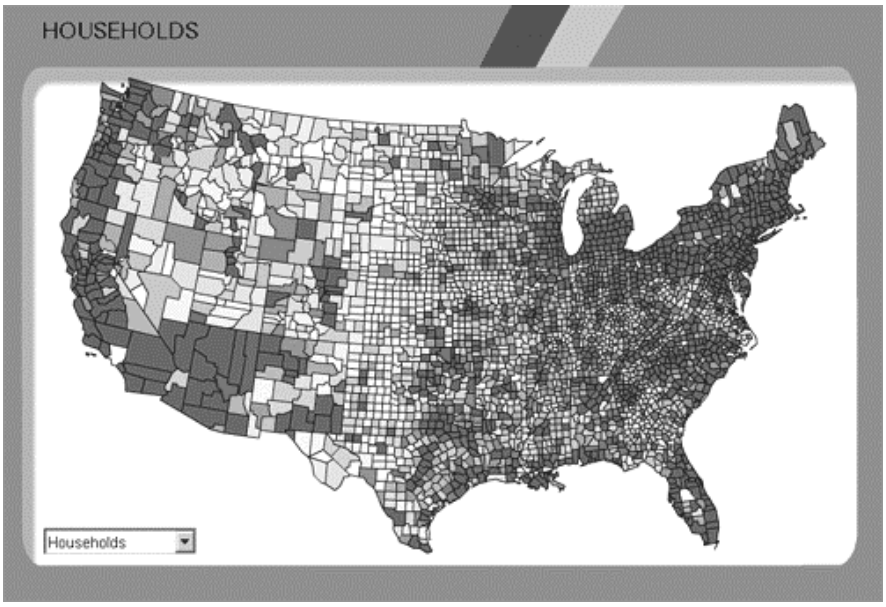


**Fig. 4.** Prototype 3: U.S. by county, loaded through large XML file

Flash provides support for interpreting and constructing XML files to allow efficient and robust client-server communication with small datasets. Flash's basic XML parser allows the designer to build interpreters for the expected XML file in order to store attribute names, values, and metadata. The following section details the database side of the prototype designs we have implemented.

## 4 Database Access

To support access to and visual exploration of a wide range of data using the tools described above (and their extensions), we are building a database of demographic, health, and related statistics on a dedicated Oracle database server. The initial implementation of each example above relied on pre-processed data stored as XML files, with data embedded in the client application at download time. Creating a dynamic link between the web-based geo-visualisation client and the database server, in contrast, offers several advantages. Among them: 1) the

user can freely select and focus on any interesting subset of data for exploration; 2) with a centralised database, we eliminate many problems regarding consistency checks, maintenance, and updates of data; 3) through a standard interface (here we adopt the XML format as the communication syntax), the visualisation modules and database management are separated, which can ease the evolution of both components (e.g., allowing us to easily replace the Flash-based geo-visualisation modules with ones written in Java). Due to the limited ability for Flash applications to process data (particularly if we are to achieve the desired lightweight client that is sufficiently responsive for real-time interaction), we also need the server to do most of the data query and processing tasks and then serve the client with a small ready-to-visualise dataset.

We evaluated two technical solutions to this dynamic connection. One is through Java Servlet technology. The other is to develop a dedicated Java server. While there are other ways to achieve the dynamic connection between Flash and a database (namely through ColdFusion MX, Macromedia newest server technology) we chose to test two available solutions in our lab. Both are described and compared below.

### 4.1 Connection through Java Servlets

A servlet is a Java class used to extend the capabilities of web servers. Servlets provide a component-based, server- and platform-independent method for building web-based applications, without the performance limitations of CGI programs. So, a servlet can be thought of as an 'applet' (a Java class for web browsers) that runs on the server side—without a 'face'. It works as follows (Fig. 5).

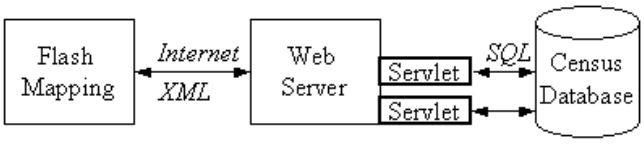


**Fig. 5.** Connection through Java Servlet

The Flash client can send a request, such as:
http://rangoon.geog.psu.edu/servlet/OracleConn?tablename=counties&statename=Pennsylvania&attribute=pop1990,white,black

Here, http://rangoon.geog.psu.edu/ is the HTTP address of the web server, OracleConn is the name of the servlet, tablename=counties &statename= Pennsylvania &attribute= pop1990, white, black is the parameter string for this request. The OracleConn servlet can parse these parameters and compose an SQL query:

Select POP1990, WHITE, BLACK from COUNTIES where statename= 'Pennsylvania'.

The servlet will then connect to the database through JDBC (Java Database Connectivity) APIs, execute the above query, and return the data. In our implementation, in addition to retrieving data, the servlet also pre-processes the data (e.g. calculates the mean value, median value, and other information) as required by the Flash client. Then the retrieved data and the derived information will be packed in XML format and passed to the client. From the Flash client's perspective, this request is exactly the same as downloading an XML file from the web server—it does not know or care how the file is generated. To be aware of which data are available in the database, the Flash client needs to retrieve associated metadata from the database before issuing any query requests. The metadata is also passed in XML format.

### 4.2 Connection through a Dedicated Java Server

Another choice for implementing the connection between the web geo-visualisation tools and the remote database is to build a dedicated Java server. Adopting this strategy, we built a mini server that does not require embedding the service in a web server (as we did with the Java servlet). The communication protocol is still XML. With our own server, we have the flexibility and freedom to design a more complex protocol than the parameter string used by Java servlets. For this implementation, the Flash client will first build a socket connection with the server. Then all the communication (including query, metadata, and data) between the Flash client and the Data server will be achieved through XML messages sent back and forth through this socket.

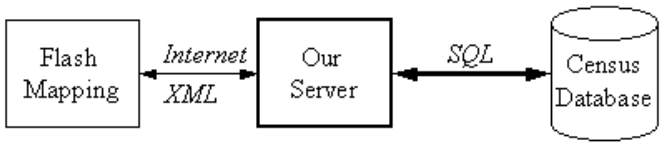


**Fig. 6.** Connection through a dedicated server.

For example, our server runs on the machine rangoon.geog.psu.edu at port 2001. The Flash client first builds a socket connection to this server. Then it can issue a query such as:

<select>pop1997, white, black</select> <from>COUNTIES</from>

<where> <statename>Pennsylvania</statename> </where> <order> pop1997 </order>

<derive> mean, median </derive>

The communication protocol can be more complex than this; this is dependent upon the agreement between the client and the server. The server then parses the above request into a SQL statement:

*Select pop1997, white, black from COUNTIES where statename = 'pennsylvania' ORDER BY pop1997*

Once parsed, the server will calculate the mean and median values of the data (as <derive> tag pair required) and pass it to the client in XML format.

### 4.3 Comparison of Two Techniques

There are both advantages and disadvantages for the two alternative forms of client-server connection described above. These are outlined for each below.

With a Java servlet, it is easy to implement and deploy a new service. However, the request-response communication is limited to a parameter string. Thus, it is hard to express a complex request. The servlet methods seem suitable for usability experiments, each of which can be simple and often need quick implementation. Another advantage of a Java servlet is that it is embedded in web servers and can be accessed easily through standard http protocol and the standard port 80, which is accessible through almost any firewall. As security issues become increasingly important in universities and government agencies, this advantage may become an overriding one.

A stand-alone Java server, while being no more difficult to build than a servlet, makes flexible and complex services possible. One example where this is important would be a collaborative environment in which distributed users could interact and observe interactions on the same dataset. A drawback of this approach is that many firewalls are configured to limit access to a small number of standard ports (ftp, http, etc.), which means the server used in the example above (port 2001) cannot be accessed outside of the local firewall.

## 5 Discussion

The DG project reported here is focused on improving the quality of maps and graphics available in geo-visualisation tools for the web and on addressing the usability of these tools through implementing and assessing a series of interface prototypes. Our broad goals are to support sophisticated exploratory graphics for experts and comparable statistical summaries in a form that is accessible to, and usable by, non-expert public users through web-based interfaces. We have now implemented several prototype interfaces in Flash (the three above are representative). We are now moving on to the tasks of assessing the effectiveness of specific geo-visualisation concepts and to determining the extent to which Flash can support both rapid prototyping and cognitive experimentation.

Developing flexible and robust geo-visualisation tools for the web is a challenge. Meeting this challenge often requires trade-offs between tool

functionality and ease of construction. Developing effective map-based interactive graphics 'from scratch' using a formal programming language such as Java or C++ requires considerable programming skill. Such development is often hampered (even for those with such skill) by the limitations of these languages when faced with implementing subtle display variations needed to support good cartographic design. Furthermore, public access to these tools is limited by web browser versions and the presence of special plug-ins on the end-users' machine.

The web increasingly supports vector-based interactive display formats in SVG (Scaleable Vector Graphics) and SWF. SVG is an XML-based open standard for displaying vector graphics (not currently accompanied by high-level authoring tools that incorporate interactivity). An open XML standard is promising for the flexible display of vector graphics, but at this point, the feasibility of rapid-prototype SVG maps and devices is questionable. SWF and its accompanying authoring environment (Macromedia Flash) offer a lightweight tool (~15% of SVG file size) who's plug-in is supported on the majority of computers (estimates suggest that 95% of current users have at least the 4.0 plug-in installed in their web browser; most computers now come pre-installed with the plug-in).

Designing in Flash holds some distinct advantages over existing development techniques in both prototyping and producing functional geo-visualisation interfaces, not the least of which is design time. Flash designers with limited programming knowledge can produce functional and graphically rich interfaces very quickly for prototyping and for testing conceptual extensions. The ActionScript programming interface is intuitive enough for inexperienced programmers, while flexible enough to support complex object-oriented development by expert designers. These qualities make Flash well suited to graphics prototype development and may support sophisticated geospatial products in the future. Flash also dramatically improves the graphical flexibility of the designer to apply existing cartographic theory to interactive displays. Furthermore, Flash applications are lightweight and run on the web through the small plug-in that has achieved very wide distribution. The transition from development environment to the web is as simple for designers as choosing 'Publish' from the Flash-authoring interface. Finally, our examples illustrate the ability of Flash to support a database connection through a dedicated server or a Java servlet. Once a connection has been made to the database, the data may be stored locally and thus support multi-perspective interactions without further database queries.

Although Flash has many features that make it attractive for web-based geo-visualisation applications, we have also encountered some serious limitations. The main drawback of designing geo-visualisation applications in Flash is the inability of the software to recognise any existing spatial data formats. The geographic base for the maps produced in these three prototypes required a time-consuming manual step—assigning a label to each display object (county). Now that the geographic boundaries for U.S. states and counties have been transformed for use with Flash, we are able to prototype county- and state-based data. However, each new ‘geography’ requires tedious manual encoding. While the format for Flash SWF export files used in web browsers is public, the Flash software and its

ActionScript language is not; and it does not support automated import of objects and conversion to movie clips (the format required in order to program subsequent behaviours). We also observed a considerable processing slow-down for loading and displaying large datasets. This limitation may become increasingly restrictive as we begin to evaluate our county-based US map (3000+ entities) while adding further interactive controls and animation. The most recent release of Flash (FlashMX) and its associated plug-in appear to dramatically improve XML parsing speed. Unfortunately, these tools were not available to test at the time of our experimentation.

Despite the disadvantages listed here, Flash-based geo-visualisation is proving useful for our cognitive and usability experiments. We are currently running a cognitive map animation experiment and a usability assessment of conditioned choropleth maps with two additional prototypes. Designing and testing rapid prototypes in an environment such as Flash has the potential to speed up development and improve the graphical integrity of exploratory geo-visualisation tools generally. The rapid prototypes allow us to explore the viability of geo-visualisation concepts and assess implementation feasibility before committing scarce resources to their development in a more robust and extensible Java environment.

### Acknowledgements and Disclaimer

This paper is based upon work supported by the National Science Foundation under Grant No. 9983451. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. Thanks go to Luis Mejias for his considerable efforts in building the geographic database of the U.S.

## References

Andrienko GL, Andrienko NV (1999) Interactive maps for visual data exploration. International Journal of Geographic Information Science 13(4):355-374

DiBiase D, MacEachren AM, Krygier JB, Reeves C (1992) Animation and the role of map design in scientific visualisation. Cartography and Geographic Information Systems 19(4):201-214, 265-266

Dykes JA (1997) Exploring spatial data representation with dynamic graphics. Computers & Geosciences 23(4):345-370

Egbert SL, Slocum TA (1992) EXPLOREMAP: An exploration system for choropleth maps. Annals of the Association of American Geographers 82(2):275-288

Harrower M, MacEachren AM, Griffin A (2000) Design, implementation, and assessment of geographic visualisation tools to support earth science education. Cartography & Geographic Information Systems 27(4):279-293

Haug D, MacEachren AM, Boscoe F, Brown D, Marrara M, Polsky C, Beedasy J (1997) Implementing exploratory spatial data analysis methods for multivariate health statistics. In: Proceedings of GIS/LIS '97, Cincinnati, OH, Oct. 28-30, 1997

MacEachren AM (1998) Cartography, GIS and the world wide web. Progress in Human Geography 22(4):575-585

MacEachren AM, Hardisty F, Gahegan M, Wheeler M, Dai X, Guo D, Takatsuka M (2001) Supporting visual integration and analysis of geospatially-referenced statistics through web-deployable, cross-platform tools. Paper presented at the Proceeding, dg.o.2001, National Conference for Digital Government Research, Los Angeles, CA, May 21-23

Monmonier, M (1989) Geographic brushing: Enhancing exploratory analysis of the scatterplot matrix. Geographical Analysis, 21(1):81-84

Open GIS Consortium (2000) OpenGIS Web Map Server Interface Implementation Specification

Peterson, MP (1993) Interactive cartographic animation. Cartography and Geographic Information Systems 20(1):40-44

Zaslavsky, I (2000) A New Technology for Interactive Online Mapping with Vector Markup and XML. Cartographic Perspectives 37:12-24