Efficient Map Visualization on the Web

MARCELO GATTASS, CARLA CRISTINA F. FERREIRA, ALEXANDRE S. VILAR AND MARK S. GLASBERG

Tecgraf – Computer Graphics Technology Group, Computer Science Department, PUC-Rio,
Rua Marquês de São Vicente, 225, 22453-900 Rio de Janeiro, RJ, Brasil
gattass,carla,asv,mark@tecgraf.puc-rio.br

Abstract. The use of Geographic Information Systems in the web requires the transmission of maps as figures. These figures can only be encoded as raster images if a very low resolution is used, otherwise they produce very large files. An important feature of these figures is that they usually have polygonal lines with a great number of points. General-purpose vector formats generate files which are also too large to be transmitted through the web. TWF is an active and scalable vector file format that is capable of storing these map figures in small files. This paper presents a set of software components that constitute a simple framework to support map visualization over the web based on this format. Examples are shown to evaluate the proposed strategies and conclusions are drawn based on these examples.

Keywords: GIS, Map, Web.

1 Introduction

Geographic Information Systems, GIS, are becoming increasingly important for most human activities, and the Internet is the main medium of dissemination of such information. Current web technology, however, does not properly support map visualization. Web servers present maps using figure encodings that were not designed to deal with geometric representations of geographical objects.

A common strategy to transmit maps over the web is to first transform the representation to a raster format. GIF (Compuserve Graphics Interchange Format) is a raster format commonly used in the web because it presents compression [1] and produces small files if the image resolution is low. Some web systems produce images with resolutions around 640x480 in order to fit them in a monitor's window. With this resolution, image sizes usually vary from 5 to 30 Kbytes. When the user wants to see some detail it is necessary to get a new image from the server representing the map in the new scale. Some systems provide maps in predefined scales, while others create a new image in the resolution specified by the user. In both cases, the interactivity of the system may be seriously compromised, because the new image has to be transmitted through the web, which is usually a low-bandwidth and high-latency medium. Figure 1 illustrates the scheme used by GIS web servers that employ this strategy.

The representation of maps as low resolution images also presents other problems. When geometrical representations of geographical objects such as lines and regions are transformed into raster images, these objects become a set of pixels and lose their identity. After this process, the map is no longer scalable and the user cannot select an object

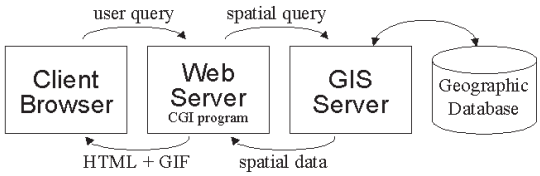

Figure 1: GIS visualization with image file

by pointing to its representation. It is easier to identify the extension of a geographic entity, like a river or a road, by selecting the graphic primitives that compound it rather than selecting an image area.

The W3C (World Wide Web Consortium) [2] has established some requirements [3] that scalable graphics must follow to be usable in the web. These requirements are divided into: general, graphical, interaction and miscellaneous. The graphical requirements define that the graphics must have a full vector representation with curve elements, text strings with scalable fonts, RGBA color, grouping and layering, template objects/symbols, coordinate systems and transformations, raster data, and a few others. Interaction includes zoom and pan, links, event handling, object selection, clipboard, and animation.

The ISO format for figure storage is CGM (Computer Graphics Metafile) [4]. This format has a significant following in technical illustration, electronic documentation and geophysical data visualization, among other application areas. For such reasons it has been the first choice for vector format, not only for important commercial products such as Intergraph's GeoMedia Web Server [5], but also for the W3C itself with WebCGM [6]. The scheme used by

the GeoMedia web server is the same one illustrated in Figure 1 except that a CGM is used instead of a GIF file. With CGM the maps became scalable and most requirements of W3C are met. With these requirements accomplished, most user interactions are efficiently handled locally. Furthermore, these maps can be printed with good quality.

The codification of maps in the CGM format, however, generates files that are much larger than the GIF files they replaced. The main reason for this large size is the lack of a special scheme to deal with the great number of polygonal points. A simple vertex count on map data shows that they can easily amount to the hundreds of thousands. If each vertex requires two floating-point numbers for its coordinates, the file will be very large. Thus, the problem of reducing the size of the file that represents a map requires a study on how to store polygonal lines with a large number of points in a small file.

The problem of producing small files that digitally represent maps is somewhat related to generalization in Cartography, in which simplified versions of the graphical representation of the objects are produced. Dettori and Puppo [7] have presented a summary of the Cartographic generalizations and have mathematically formalized the operations that transform the map representations in simpler and more efficient ones to be used in visual communication. The generalization of polygonal lines is defined as simplification.

Polygonal lines are usually simplified according to some geometric criteria such as those proposed by Douglas-Peucker [8]. Such criteria use a tolerance normally defined based on the cartographic map scale. If, for example, 0.1 mm is an acceptable error in the map drawing and the map scale is 1:500.000, thus a tolerance of 50 meters in the map coordinates is acceptable. The goal in such criteria is to produce the same visual representation with a reduced number of polygonal points, that is, perceptual criteria.

These type of criteria, designed for map printing process, do not take into account the scalable nature of the vector representations on web browsers. In this new environment the size of the visualization surface, window on the monitor screen, is not as controlled as the size of a paper in a plotter. Different monitors may yield different sizes. Furthermore, if zoom capabilities are offered to the user, he or she may perceive differences depending on the zoom factors.

There are several possibilities for simplification criteria in the web environment. A common denominator in current visualization hardware is that they are all based in some resolution. Monitor screens vary between 75 to 100 dpi (dots per inch) while printers and plotters may reach 2400 dpi. For this reason, a simple and direct perceptual criteria can be established if the maps are embedded in the integer coordinate systems. A map entity can be generalized if the embedding of its geometric entities in the integer grid (rasterization) yields the same pixel colors. That is, the simplification criterion is defined based on the visualization surface resolution of the device used (canvas). This resolution must take into account the zoom capability locally offered to the user.

Another factor to reduce the file size is the process of coordinate quantization. If, for example, a grid of 4096x4096 positions is used, then each coordinate should not require more than 12 bits. Furthermore, a vertex of a polygonal line that represents a natural object is likely to be close to its predecessor, suggesting that relative positions may be stored with less bits than absolute coordinates. Both considerations must be present in the codification of the coordinates in the file.

Even after the simplification and codification processes have taken place, there may still be room for reduction of the file size. This is mainly due to global redundancies such as repeated chains of coordinates. Lossless dictionary-based compression algorithms, such as LZW [9] and LZO [10], can be used to further reduce the file size.

Ferreira, Gattass and Figueiredo [11] have proposed a file format, called TWF, to fulfill these requirements based on studies on compact polyline representations [12]. This representation is based on two processes: codification of the polyline into a file and interpretation of the file.

The codification process, with compression, has the following steps:

- Quantization: coordinates are transformed to the integer Cartesian coordinate system (Z^2).
- Simplification: zero-size line segments created in the previous item are eliminated.
- Codification: figure primitives are codified in a compact way.
- Compression: classic lossless general-purpose compression algorithms are used.

It is worth noting that this proposal does not include cartographic generalization, i. e., it is assumed that the data has already been generalized before being submitted to this process.

The interpretation process consists of only two steps:

- Decompression.
- Primitive interpretation.

This paper extends the TWF proposal into a simple framework to support map visualization over the web, as seen in Figure 2. The TWF framework includes, besides the format specification, authoring and viewer components.

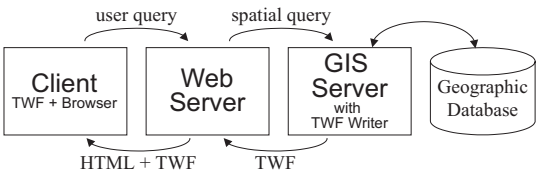

Figure 2: GIS web visualization with TWF

The format specification is outlined in the next section and can be found in www.tecgraf.puc-rio.br/twf.

There are two types of authoring tools: one for programmers and another for end-users. The programmers tool is an ANSI-C library presented in section 3 and the authoring tools are extensions on commercially or publicly available software. We have recently developed a GeoMedia [13] extension based on the COM (Component Object Model) [14] technology. This extension exports GeoMedia data as TWF files, mapping each GeoMedia feature data to a TWF layer.

Two different viewers are described here: an applet based on the Java 1.2 platform [15] and a Netscape plug-in based on C/OpenGL [16]. The first one has the full TWF functionality and the second yields a very efficient implementation. Section 4 discusses both viewers.

The last two sections present some examples, conclusions and suggestions for future works.

2 The TWF Proposal

TWF (Tecgraf Web Format) is a file format designed to represent a drawing that contains lines, regions, text and images to be sent through World Wide Web technology for displaying, browsing and printing. Its main goal is to allow drawings to be coded into small files without converting vector primitives into raster images. Text, lines, curves and regions retain their scalable nature and can be directly selected by the user in the browser screen.

A TWF file is composed of a header and a series of layers. Each layer is composed by functions, and each function has a unique identifier and a variable number of arguments, as shown in Figure 3.

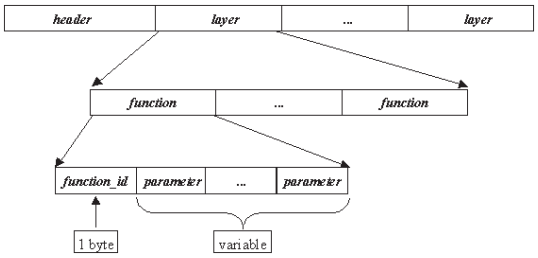

Figure 3: TWF file structure

The different functions and their semantics hold the real functionality of the Tecgraf Web Format proposal. A viewer is supposed to execute the functions as they are read to render the drawing.

Name-value pairs containing arbitrary data can be inserted anywhere in the file. This could be used, for instance, to convey external information about the different primitives of the drawing to the viewer. This is implemented by the Extension function.

All primitives belong to one and only one layer, and all layers are named, with the exception of a default control layer. This is implemented by the Layer function.

Primitives may be grouped into objects. Objects can be selected by the user and associated to an action. This is implemented by the Object function. The TWF Applet currently implements an association of objects to Universal Resource Locations (URLs). When the user clicks on a region of a drawing, the web browser goes to the associated location as if a link had been activated. Other viewers could have more complex associations and, since the behavior and action descriptions are separated from the file itself, the same drawing could be employed for several different uses.

Raster images may be embedded once into a TWF file and used as textures for several areas or scaled, rotated, translated and drawn as many times as desired. This allows for an efficient use of raster images in a TWF file. Raster images are usually represented in raw format in a TWF file, but any MIME type representation supported by the viewer, such as the JPEG or PNG formats, may be used. This functionality is implemented by the SetImage, SetMIMEImage, Image, and FullImage functions.

Primitives may be grouped into shapes, which are the TWF equivalent of macros, template objects or symbols. Shapes are defined once and may be instantiated and transformed as many times as desired. This functionality allows for greater flexibility and reduced file size when the same group of primitives appears several times in the same drawing. It is implemented by the BeginSetShape, EndSetShape, Shape, StampShape, and RepeatShape functions.

Alpha compositing to achieve blending and transparency effects with graphics and images is supported. The rules used are a subset of the rules described in Porter-Duff [17]. This functionality is implemented by the SetBlendMode function.

Colors may be defined by color channels (RGB or RGBA) or by indexed palettes. The 256-color palette might be changed in any point of the file with the SetPalette function, which relaxes the common restriction some file formats have of a single 256-color palette.

Polygonal lines are compressed with techniques similar to the ones presented by Ferreira [12]. Furthermore, polygonal lines that define part of the boundary of more

than one region do not need to be stored twice, as would happen in most vector file formats. This is specially important for thematic maps where each line usually bounds two regions. An example is given in Figure 4.

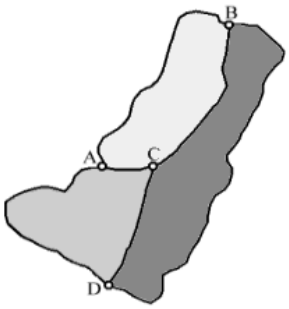

Figure 4: TWF complex region representation

To represent the figure in Figure 4, on most vector file formats, one would have to do something equivalent to the following commands: draw AC, draw AD, draw AB, draw CB, draw CD, draw BD, fill ABC, fill ACD, fill BCD.

This kind of representation is very wasteful, because the same polygonal line must be repeated several times in the file. When large polygonal lines are involved, this results in unnecessarily large files. The points in the CB edge, for example, would have to be listed three times in the file: once to draw the edge CB, once in the filling of the closed area ABC and once in the filling of the closed area BCD.

TWF drawings use an alternative representation through which closed areas, called faces, are defined by a list of edges. Edges are polygonal lines which are defined only once and then referenced as many times as necessary, with no need to relist their points. The drawing above could be represented in TWF with no repetition of points, and this allows for great savings in file size, specially for drawings that represent thematic maps.

It is important to note that the same data can have different representations and codifications in the TWF format. That is, both representations of Figure 4 can be used. This flexibility is necessary to facilitate the convertion from other file formats.

For further gains in file size, TWF allows the file to be optionally compressed with the zlib compression library [18]. According to its creators, zlib is designed to be a free, general-purpose, legally unencumbered – that is, not covered by any patents – lossless data-compression library for use on virtually any computer hardware and operating system.

3 The TWF Writer

The TWF Writer is a library that allows programmers to create TWF files. It is composed by three modules: compilation, encoding and compression. The compilation is responsible for converting a high level description of a drawing into a series of TWF functions. The second component encodes each one of these functions in a file. Finally, compression uses zlib [18] to compress the encoded file.

The compilation module is a client of the encoding module. A programmer could use the encoding module directly and produce a valid TWF file. The need for the compilation module came from the complexity of producing small files from a general purpose description of graphical objects. The programmer can describe each geometric representation of the geographical object in its own coordinate system. The mapping from this system to the TWF integer grid produces repeated vertexes that are eliminated in this step (simplification process).

The goal of this component is to allow near optimal files in different quantization resolutions to be easily generated. All the work of this component is controlled by a few operation parameters and performed internally by the TWF Writer, in order to make it simple to use.

As a simple example of the control performed by this module, each polygonal line must have it vertices represented in the smallest point size and each change in point size requires a new TWF function to be stored in the file. As a new polyline is being defined with the begin polyline, vertice, ..., vertice, end loop, this module decides which size to store the first absolute point and all relative points in. More complex optimization tasks are the ones that involve semantic transformations such as the one illustrated the example discussed in Figure 4.

The TWF library is written in ANSI C, in order to obtain portability. Furthermore, the API (Application Programming Interface) of the TWF Writer was designed in a way that it does not force the developer to use specific data structures to the function parameters. They are mapped only to basic C language data types. As a side benefit, it is very simple to write bindings of the TWF Writer to embedded scripting languages, such as Lua [19].

The API allows the writing of several files simultaneously as easily as writing a single file.

The TWF Writer can be used to implement converters that read data from other vector graphics files to TWF files. Another important goal of the TWF Writer is to support the creation of CGI (Common Gateway Interface) programs, to generate TWF files dynamically and to supply them to clients through a web server.

4 Viewers

A viewer for TWF files is supposed to comply with the TWF standard and render the drawing correctly. Moreover, since TWF files contain scalable vector graphics, specific functionalities must be available in the user interface.

The user must be able to navigate the drawing by performing panning, and zooming in and out of specific regions. The layers can be either hidden or selected as visible by the user, to allow him or her to see only the information desired.

One could imagine a drawing of a city plan to contain its streets, hotels and restaurants in three different layers, and that would allow the user to conveniently disable the rendering of information he or she does not need. Zooming in on a specific street to be able to read its name and learn the exact address of a hotel, or zooming out in order to get a broader view of the city are two typical uses of drawing navigation.

Two different approaches have been taken when implementing viewers for TWF files. The TWF Applet is a Java 1.2 applet based on the Java 2D API. The TWF Netscape Plug-In is a Netscape Navigator^{TM} plug-in program based on OpenGL^{TM}.

4.1 The TWF Applet

The TWF Applet [20] is a platform-independent solution. It allows any Java 1.2 web browser to view TWF files, as the code is automatically and transparently downloaded from the server and executed on the client as needed.

The Java 2D API presented itself as a very complete and high-level 2D API, which allowed the straightforward and complete implementation of all the TWF features.

On the downside, Java is an interpreted language, and the Java 2D API has poor performance when compared to native graphics libraries. The fact that Java uses garbage collection also adds a few pauses when rendering occurs, and that affects the perceived performance.

Another problem with this platform is that Java 1.2 support is still not available in the two most popular web browsers, Netscape Communicator^{TM} and Internet Explorer^{TM}, unless the user downloads and installs the Java Runtime Environment, which contains the required Java Plug-In.

The TWF Applet includes support to the association of link behavior to objects defined in a file. The web master or CGI program can insert parameters in the HTML page where the applet is loaded to associate URLs to object names. The user may then navigate the drawing and click on a primitive that is part of an object, which will cause the TWF Applet to point the browser to the correct URL.

The TWF Applet also features a partially progressive display, in which each layer is rendered on screen as soon as it is fully downloaded and interpreted. The user is free to work with the partial drawing while it is loading, as the applet uses a background thread to download and interpret the TWF file.

4.2 TWF Netscape Plug-In and TWF Viewer

The TWF Netscape Plug-In [21] is currently restricted to the Win32 versions of the Netscape Navigator^{TM} web browser. It requires the user to download and install the plug-in.

OpenGL^{TM} has presented a series of difficulties to implementing some features of TWF, because it is a low-level graphics library with a series of performance-oriented simplifications. The filling of concave polygons and limited text support are some of the problems still to be solved.

On the bright side, OpenGL offers excellent performance and hardware acceleration support for the rendering, and this allows the viewing of very large TWF files with acceptable response times. OpenGL is also a portable graphics library, which would allow the easy porting of the plug-in to other platforms.

The object support is not yet functional in the TWF Netscape Plug-In, but there are plans to implement functionalities equivalent to those of the TWF Applet.

There is also a stand-alone program called TWF Viewer [22] which uses the same OpenGL-based renderer to view TWF files.

5 Examples

The original files used in tests are either CGM or BIN. The BIN is a plain binary format where each polygonal is represented only by its number of points (integer) followed by the point coordinates (float). It only contains polygonal lines. The CGM format used here is binary with integer coordinates (32 bits). The figures used in tests are:

- Madison Isolines [23] (Figure 5)
- Municipal Districts of Brazil [24] (Figure 6)
- Vegetation of Brazil (Figure 7)

The original Vegetation of Brazil map is a CGM file, and Municipal Districts of Brazil and Madison Isolines are BIN files. The Madison Isolines map (Figure 5) is so dense that it is difficult to visualize the isoline details. Table 1 shows some information about the original files.

Images and figures have been generated from the original files into various formats (TWF, CGM, DWF [25] and GIF) in two resolutions (4096x4096 and 512x512). For comparison, we present all file sizes with and without the compression storage by the zlib [18]. Note that this compression does not yield any significant change in the case where the original format has compression embedded in it (GIF, DWF). Table 2 presents the size of each one of the resulting files.

Data	Format	Size (Kb)	Number of points	Number of polygonals	Average # of points per polygonal
Madison Isolines	BIN	5212	665674	2840	234
Municipal Districts	BIN	5666	716933	16586	43
Vegetation of Brazil	CGM	830	196977	2781	71

Table 1: Data Characteristics

	4096x4096				512x512			
Data	**TWF**	**CGM**	**DWF**	**GIF**	**TWF**	**CGM**	**DWF**	**GIF**
Madison Isolines (Kb)	1229	2645	1221	1512	375	2645	432	8
zlib compressed (Kb)	724	1885	1090	1512	213	894	339	8
% GIF	48	125	72	100	2662	11175	4237	100
Municipal Districts (Kb)	460	3005	521	337	177	3055	156	16
zlib compressed (Kb)	284	878	451	337	86	230	138	16
% GIF	84	261	134	100	538	1437	862	100
Vegetation of Brazil (Kb)	336	839	311	438	122	839	112	24
zlib compressed (Kb)	198	480	276	438	62	198	91	24
% GIF	45	110	63	100	258	825	379	100

Table 2: Format Comparison

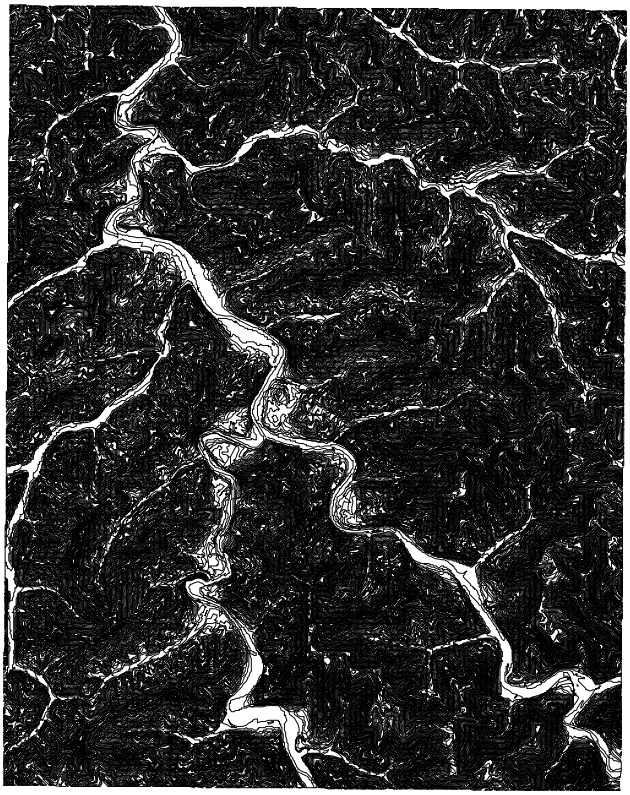

Figure 5: Madison Isolines

Figure 6: Municipal Districts of Brazil

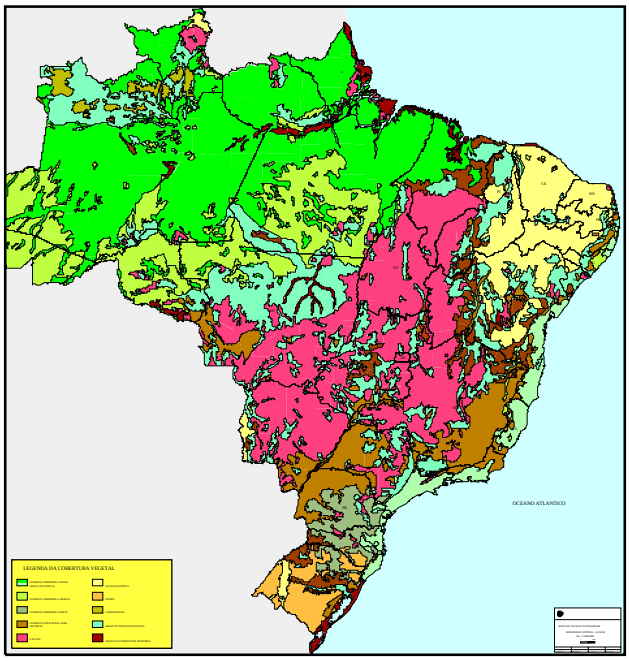

Figure 7: Vegetation of Brazil

6 Conclusions

This paper investigates figure vector formats for map visualization on the web. Embedded in this investigation there is a format proposal, called TWF, to replace currently available vector and raster formats. The primary goal of the format is to produce high quality scalable figures in small files.

The results obtained so far indicate that not only studies in simplifications procedures are important to produce reasonable size files. Quantification and compression are also important steps to reduce the file size.

Even with only simple optimizations implemented, our TWF writer produced the smallest files when compared with other scalable-vector formats. TWF files are even smaller than GIF files if a good resolution is required. Only in small resolutions do GIF files present better results than TWF. In this situation, however, the figures are low quality and can not be scaled.

As for future works we intend to study map generalizations that are more appropriate to the scheme we are proposing. We also intend to investigate optimizations procedures that would automatically re-write a TWF file format, without changing it pictorial content, to a smaller file.

We are also in the process of writing a CGI program to act as a GIS web server that would allow users to navigate geographic databases through the web. The goal is to dynamically generate TWF files directly from high level user queries and to transfer them through web servers to client viewers. Difficult topics in this subject that we require investigation are: user interface models, distributed data base models and spatial query languages.

7 Acknowledgements

Luis Henrique de Figueiredo has also advised the M.Sc. dissertation of Carla Ferreira on this subject. Paula Frederick and Anselmo Paiva wrote the GeoMedia extension. Gilberto Câmara and many Tecgraf GIS team members provided important contributions and suggestions for this work. Tecgraf is a Laboratory mainly funded by PETROBRAS.

References

[1] J.D. Murray and W. Vanryper. *Encyclopedia of Graphics File Formats*. O'Reilly & Associates Inc., 1994.

[2] World Wide Web Consortium. www.w3.org.

[3] World Wide Web Consortium Requeriments. www.w3.org/Graphics/ScalableReq.

[4] ISO/IEC 8632:1992. *Information Technology - Computer Graphics - Metafile for the Storage and Transfer of Picture Description Information*, 1992. Part 1 - Functional Specification (ISO8632-1), Part 2 - Character Encoding (ISO8632-2), Part 3 - Binary Encoding (ISO8632-3), Part 4 - Clear Text Encoding (ISO8632-4).

[5] GeoMedia Web Map. www.intergraph.com/software/geo_map/geo_web.asp.

[6] W3C WebCGM Profile. www.w3.org/Graphics/WebCGM/.

[7] Enrico Puppo and Giuliana Dettori. Towards a formal model for multiresolution spatial maps. In *4th International Simposium on Large Spatial Databases*, pages 152–169, 1995.

[8] D. H. Douglas and T. K. Peucker. Algorithms for the resolution of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10:112–122, 1973.

[9] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory IT-23*, 3:337–343, 1997.

[10] Markus F. X. J. Oberhumer. LZO real-time data compression library. www.widsau.idv.uni-linz.ac.at/mfx/lzo.html.

[11] C. C. F. Ferreira, M. Gattass, and L. H. Figueiredo. Um estudo sobre arquivos vetoriais para a

visualização de mapas na web. *GIS Brasil 99 - V Congresso e Feira Para Usuários de Geoprocessamento da América Latina*, 1999.

[12] Carla Cristina Fonseca Ferreira. Um estudo sobre arquivos vetoriais para visualização de mapas na web. Master's thesis, Departamento de Informática PUC-Rio, 1998.

[13] Geomedia. www.intergraph.com/geomedia.

[14] Dale Rogerson. *Inside COM*. Microsoft Press, 1997.

[15] The Source for Java Technology. java.sun.com.

[16] OpenGL. www.opengl.org.

[17] T. Porter and T. Duff. Compositing digital images. *SIGGRAPH*, pages 253–259, 1984.

[18] zlib Compression Library. www.cdrom.com/pub/infozip/zlib/.

[19] The Lua Programming Language. www.tecgraf.puc-rio.br/lua.

[20] TWF Applet. www.tecgraf.puc-rio.br/twf/applet/.

[21] TWF Netscape Plug-In. www.tecgraf.puc-rio.br/twf/plugin/.

[22] TWF Viewer. www.tecgraf.puc-rio.br/twf/viewer/.

[23] Madison Isolines. ftp://spectrum.xerox.com/ds9/map/dlg/1:24,000/optional/.

[24] Malha municipal digital do Brasil - situação em 1991 e 1994. Base de dados em CD.

[25] Whip! and DWF - A Primer. www.autodesk.com/products/whip/primer.htm.