In Journal VIRTUAL, special edition *Advances in Computer Graphics in Portugal*, ISBN: 972-98464-5-6; ISSN: 0873-1837. Oct. 2004. Online version: http://virtual.inesc.pt/aicg04

VGLib2D – Class library for the 2D graphic visualisation of large volumes of data

Sofia Gameiro Luís Almeida
Centre of Computer Graphics (CCG)
Coimbra Department
Centro de Empresas de Taveiro
Estrada de Condeixa
3040 – 912 Coimbra, Portugal |
{Sofia.Gameiro,Luis.Almeida}@coimbra.ccg.pt

Adérito Marcos
CCG / University of Minho
School of Engineering
Department of Information Systems
Azurém Campus
4800-058 Guimarães, Portugal
marcos@dsi.uminho.pt

Abstract

This article presents the implementation of a class library in C# language, which uses the recent .Net technologies, aimed at the creation of documents or images in Scalable Vector Graphics (SVG) format, for the representation, visualisation and printing of large volumes of two-dimensional (2D) data.

The goal is to take advantage of SVG for applications implementation effects that, for instance, are Web based and where the graphical representation of large volumes of data, both dynamically and in real time, is necessary, along with its immediate availability for the purposes of visualisation, printing and downloading.

Keywords

SVG (Scalable Vector Graphics), XML (Extensible Markup Language), 2D Information Visualization, C#, .Net

1. INTRODUCTION

Without a doubt the World Wide Web is the biggest information repository and circulation vehicle of today. Its ease of access, low cost of utilisation and simplicity of navigation has turned it into the most suitable tool for information searching all over the world.

Contributing to the growth of this phenomenon is the constant evolution of new solutions for the representation and access to information of all kinds and formats. Such an evolution ranges over important issues such as look and design, navigational simplicity, interaction and animation possibilities and speed of download among other matters. Simultaneously, it is essential, whatever its format, that all information should be searchable.

With respect to textual information both its visualisation and searchability are optimised but graphical information, however, is not as easily searched or visualised. Such a reality is the result of some still existing limitations concerning Web technology, particularly regarding the presentation of 2D images (non-animated or video).

SVG is an XML technology that appeared to overcome this situation. Besides allowing the representation of graphical information in a compact, generic and portable format [Eisenberg, 02], its textual nature (implicit in its XML format) greatly facilitates the search of all represented information [Probets, 01].

Thus, the library implemented version - the VGLib2D – aims at the creation of SVG documents and images in a suitable way for the 2D representation and visualisation of large volumes of data. The authors, therefore, claim that SVG is the most suitable format for graphical information representation whenever this information has a strong vector component or is totally represented by vectors, for these and other reasons presented below.

In the following sections of this article a brief reference is made to the current technologies, which is followed by an approach to SVG technology – its characteristics and potentialities – and by some of the technical issues concerning implementation of the library. Finally, an actual case of the implemented library is presented and reference to future work is made.

2. CURRENT TECHNOLOGIES

.Net platform [Microsoft, 03], which is a set of technologies created to simplify the development of the WWW and Microsoft Windows applications [Duthie, 02], was chosen as the development platform. Technologically current, these software technologies based on a set of XML Web Services (small re-usable applications written in XML that is a language aimed at the unification of information representation and circulation), have been shown to enable a high level of software integration.

In this context, the class library was implemented using C#, which is a fairly recent programming language from Microsoft® that has been specially designed for the .Net platform. Based on the Object Oriented Programming paradigm, the C# language derives from C and C++ and it closely resembles C++ and Java. Moreover, being based on the vast class library available in the .Net Framework, C# promotes software reliability, consistency and efficiency [Drayton, 02].

Although .Net platform does not have specific name spaces (or sets of specific classes) for working with SVG, as it is such a new technology, it was possible to use its robust support for XML, which thereby secures

the creation of any desired type of SVG image on the fly [Wahlin, 02]. All programming development was made using Visual Studio .Net from Microsoft®.

3. SVG TECHNOLOGY

3.1. Introduction

One of the most exciting XML technologies of today is Scalable Vector Graphics (SVG). This format, presenting an XML based syntax, can be used for the creation and visualisation of images (especially vector images or combined images with embedded raster elements in the vector image) from pictures to reports and graphics in a variety of devices [Wahlin, 02].

For the creation of these images, SVG allows the use of all the specific attributes and elements defined in the SVG Document Type Definition (DTD), which has been published in World Wide Web Consortium – the W3C site([W3C, 00a]). Although there is a vast list of elements and attributes to learn, it is relatively easy to work with SVG and visual editors of companies such as Adobe® and JASC can export images in SVG format [Wahlin, 02]. For its visualisation in a browser a specific plug-in is necessary and the Adobe® SVG Viewer [Adobe, 01] is the most known one.

In Text 1 and the accompanying Figure 1, an example of the SVG basic elements used to draw forms and text is presented, which shows a simple image [Wahlin, 02]:

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C// DTD SVG 1.0//EN"
"http://www.w3.org/TR/2001/REC-SVG-
20010904/DTD/svg10.dtd">
<svg width="300" height="150">
<rect x="31" y="36" width="240" height="59"
style="fill:#02027a;stroke:#000000;stroke-
width:1"/>
<text x="56px" y="67px" style="fill:#ffffff;
font-family:Arial; font-size:18"> My First SVG
Document </text>
</svg>
```

Text 1: Example of an SVG document

Figure 1: Graphical visualisation of the document using Adobe SVG Viewer

3.2. Main characteristics of SVG

It is claimed that the form of graphical representation in the Internet will be revolutionised by the SVG format, and its success results from some of its characteristics [Probets,01] [Eisenberg, 02] [Mouza, 02] [Marriott, 02], which were decisive in its choice for the implementation of VGLib2D, namely;

- **Vector Format**;
- **XML Syntax;**
- **Text file**, which allows search engines to index SVG files or SVG embedded components in HTML files (HyperText Markup Language);
- **Platform and device independent;**
- **Allows for interaction**, as it possesses a number of event-oriented procedures, which permit a certain degree of interaction with the user;
- **CSS (Cascade Style Sheets) and XSL (Extensible Stylesheet Language) Integration**;
- **Flexibility**, which concerns attributes and elements that allow a separation between the graphical presentation and the geometric structure of the elements;
- **DOM (Document Object Model) Interface**, which allows the modification of a document from the server side and, when this happens, the SVG browser immediately transmits the changes to the graphical display; and
- **Allows for animation.**

In spite of all that has been presented above, SVG also presents some limitations that must be recognised, namely;

- **The adaptation of the client side, according to different visualisation conditions, is not immediate.** Although SVG is currently used in different devices ranging from PDAs (Personal Data Assistants) and mobile phone to big conference room display devices, it requires the prior knowledge of the device characteristics, to enable the adaptation of the visualisation window to the device display;
- **The textual format of the file can, in some situations, result in large dimension files**. This problem can be overcome at the implementation level, if some precautions are taken into account, which are meant to assure the file optimisation.

These are, however, minor limitations for which solutions have already been found, as will be shown later in this article.

3.3. Areas of SVG application

Supporting the strong impact of SVG in graphical representation and visualisation technologies today is the increasing recurrence of several and very different knowledge areas for this new technology.

Only four examples will be presented in this article. They have been selected from the large number available and are considered to be representative of the flexibility of application of this format, as well as of its success and dissemination concerning Computer Graphics.

In addition to reviewing the state-of-the-art of SVG technology, the aim is also to present the potentialities of SVG in a more concrete way, based on real examples.

3.3.1. SVG and cartography

SVG technology has been particularly attractive to developers and users of Geographic Information Systems (GIS). In fact, cartography is, no doubt, a perfect area for the appliance of SVG, as maps are, by definition, vector representations of 2D areas that are distributed in layers. The SVG specification includes that same concept of layers or groups, which is essential for information representation using GIS.

Some of the actual advantages of SVG are its image manipulation (pan, zoom, etc.), interaction, animation and layer control functionalities that cover all essential GIS features, and the fact that it is an open source solution (thus reducing implementation costs) and a standard (SVG is an open recommendation developed by a cross-industrial consortium). Not being a substitute for a complete Geographic Information System, SVG personalises a new way of representing quality geographic information on the Web [Seff, 02].

An example of the application can be found (see Figure 2) in custom mapping [Custom, 03]:

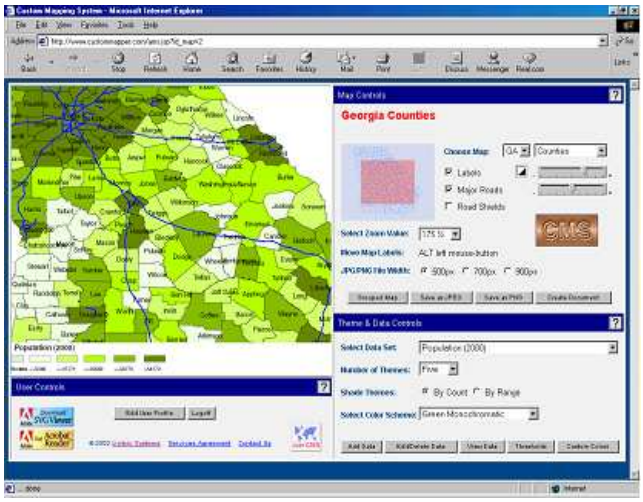

Figure 2: Custom Mapping System v1.1 developed by Limbic Systems, Inc

3.3.2. Chemical Markup Language (CML™)

Chemical Markup Language (CML™) [CML, 03] is a new approach to molecular information management that takes advantage of XML technologies.

Supporting extremely complex information structures, it acts as an exchange or archive mechanism. In addition, it is machine-independent, which consequently guarantees portability of data between different operating systems and machine architectures.

In CML2SVG. URL [Adobe, 03] the following example can be found in which information filed in CML™ is manipulated via an Extensible Stylesheet Language (XSL) in order to create an SVG file that is ready for visualisation and interaction (see Figure 3).

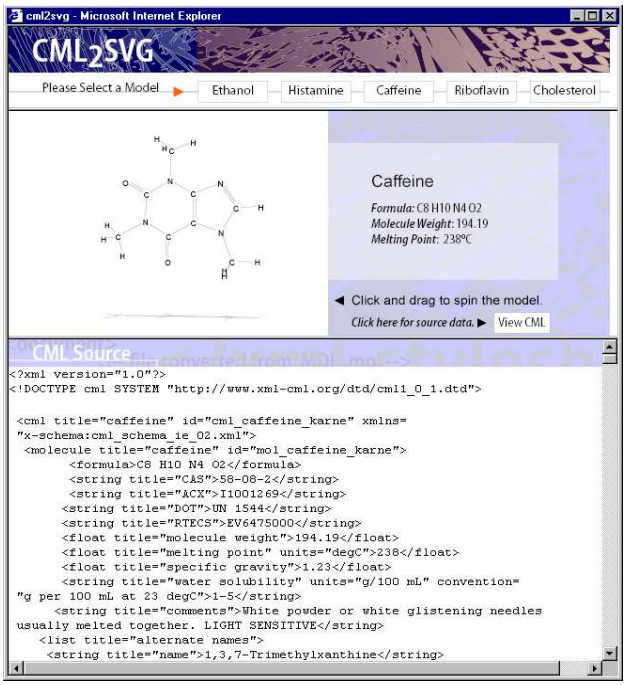

Figure 3: 3D Visualisation of a caffeine molecule in SVG format obtained from the CML™ format

3.3.3. SVG and Mobile Devices

Mobile devices such as PDAs, PalmPilots, SmartPhones and others help their users in the management and optimisation of "to do" activity sequences.

In this context, SVG format has been used for the representation and visualisation of all graphical information associated with these activities and tasks. This implies the interaction of a set of visualisation techniques such as the spatial representation of the place where they occur, temporal representation, textual representation using lists, continuous text and hierarchies, priority representation (using colour) among others [Bieber, 03].

3.3.4. Laboratory Notebook

This last example is probably the most surprising one, as it highlights the extraordinary applicability and versatility of the SVG format.

It presents a new concept - the "augmented laboratory notebook," which is a working space intended to make the connection between information on paper and in a digital format in a laboratory context.

Although it is not intended to go deeper into technical issues related to the implemented prototypes, it is important to state that each notebook concerns a set of pages and that each page contains several (different) layers of information.

SVG was the chosen format for the representation and posterior visualisation of all graphical information implicit in the use of this device that, as the authors claim, is capable of acquiring the proportions of an authentic set of augmented reality tools for working over paper [Mackay, 02] (see Figure 4).

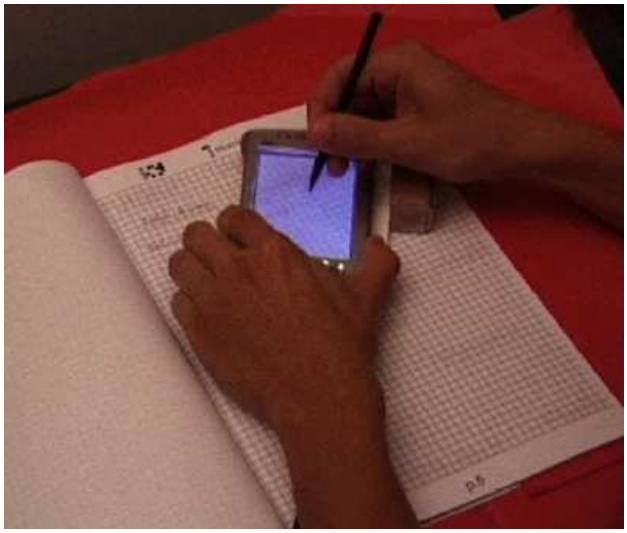

Figure 4: One of the implemented prototypes, the a-book [Mackay, 02].

4. THE LIBRARY

4.1. Introduction

As already stated, the .Net platform does not present specific name spaces for working with SVG. This is also, at the moment, an intrinsic limitation of other existing programming development languages and platforms. Thus, all implementation with respect to the generation of SVG files is still made at a very low level. This, which is the result of the relative novelty of SVG, is also in a way worrying, if we take into account the advanced state-of-the-art of this format, as well as the large quantity of applications being developed with SVG support.

It was precisely with the aim of overcoming this limitation that the authors decided on the implementation of VGLib2D - a high level class library that allows the creation and writing of SVG files. More than this, VGLib2D proposes to take advantage of SVG potentialities, as cited above, in such a way as to allow the graphical representation, downloading and printing of large volumes of data at the desired speed and quality.

VGLib2D understands low level classes for the creation and properties definition of the basic SVG elements, such as *SVGCircle* that is used for the creation and definition of circles, *SVGLine* used for the creation and definition of lines, etc., as well as higher level classes that work as an interface for the creation of SVG documents such as header writing, elements writing, commentaries, etc., which are based on the aforementioned low level classes (see Figure 5).

The purpose of this division is to obtain a clear separation between the SVG core (SVG elements and attributes) and the library interface, which is accessible to the programmer and intended for the creation and management of all graphical entities to be represented.

VGLib2D was developed in order to work in a similar way as the .Net Framework name spaces and in this way it assures the integration and communication between the two.

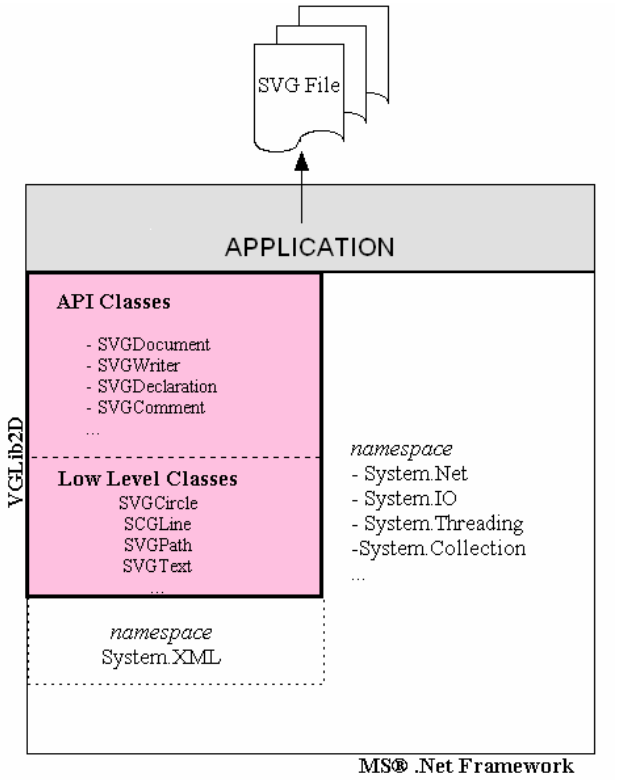

Figure 5: VGLib2D – Architecture and Communication with the .Net Framework

4.2. Implementation strategy

As a starting point for the library implementation, the authors took into account the System.Xml namespace present in the .Net Framework Software Development Kit and included in the class library FCL (Framework Class Library) from Microsoft®.

This namespace offers support for the management of XML documents according to a set of standards defined by the World Wide Web Consortium (3WC). Its classes implement objects that respect the XML1.0 specification ([W3C, 98]) and the DOM Core Level 1 and Core Level 2 ([W3C, 00b]) [Drayton, 02]. The aim is to validate the XML syntax of all SVG files and the creation of a coherent and structured DOM that allows future editions (thinking in advance of the future implementation of interactivity functions, among others).

For the validation of the created SVG files, the authors used the software XMLSpy® [XMLSpy, 03]. Currently there are also SVG validators available, as in the case of the service provided by W3C and available from the W3C [W3C, 03]. This service was created in the context of a W3C project[1] and, as far as SVG is concerned, it implies the validation of the following items: XML definition, SVG DTD, attributes grammar (as presented in the specification) and stylesheets specification. SVG files created using VGLib2D and tested using this service presented no errors.

[1] "Quality Engineering Solutions via Tools, Information and Outreach for the New Highly-enriched Offerings from W3C: Evolving the Web in Europe"

Once that a SVG file is nothing more than a variation of an XML file, and considering that this implementation is based on the System.Xml namespace, the authors came to the conclusion that the VGLib2D architecture should reflect, in a certain way and within certain limits, the architecture of the referred namespace. Therefore, the interface classes such as *SVGWriter*, *SVGDocument*, *SVGDeclaration*, etc. present exactly the same methods as the ones that can be found in the parallel classes from the System.Xml namespace, together with other specific methods aimed at the SVG format and its particularities. This is the case of such methods as *SetWidth*, *SetHeight*, *SetUnits*, *DrawLine*, and *DrawCircle* among many others. The intention is to extend the utilisation philosophy inherent in the .Net Framework namespaces to the VGLib2D. This will assure once again the uniformity of the interfaces, so a .Net platform experimented programmer should have no difficulty in using the implemented library.

The developed library aimed for the maximum degree of abstraction and for a consistent hierarchy, guaranteeing in this way its applicability for a diversity of applications.

For the creation of the SVG files there are some optimisation questions that should be taken into account, such as the preferred use of certain attributes like the *path* in substitution for multiple uses of the *line* or *polyline*, the *H* and the *V* for horizontal and vertical lines, among others. Similar considerations apply to curves [Probets, 01]. After some debate, the authors opted to delegate to the user the responsibility for optimisation management (which implies some degree of decision-making concerning the selection of methods). Nonetheless, VGLib2D possesses all the necessary mechanisms for this optimisation.

4.3. Characteristics

VGlib2D allows for the creation of SVG documents of various dimensions, as defined in all metric units considered valid by this format specification.

Furthermore, after allowing the definition of style sheets and of general styles for the document, which can be automatically adopted by all created elements that refer to these styles, VGLib2D also allows the definition of graphic elements with their own style independently from the general styles.

All elements or entities that are part of the SVG format specification are considered in VGLib2D, so it is possible to define and edit attributes and to apply geometric operations, such as translations, rotations and scalings, to all these elements.

Examples of these supported elements and the respective attributes are lines with attributes such as width, colour, opaqueness and style of stroke (continuous, dashed, etc.); text with attributes such as size, width, font, alignment, letter and word spacing, orientation (vertical, horizontal, going along a path, from bottom to top, right to left, etc.); among many others whose exhaustive enumeration is not possible in the context of this article.

The interface for the definition of the graphic elements is intuitive and very well documented. All functions obey a simple and clear syntax, as can be seen in the following example that allows for the definition of a continuous line:

- *DrawLine (SVGWriter filePointer, string id, double x1, double y1, double x2, double y2, int width, string colour);*

In this case we have the *filePointer*, which is the pointer for the created SVG file, *id* is the identifier of the created line, *x1*, *y1*, *x2* e *y2* are the line co-ordinates and *width* and *colour* are the width and colour of the line, respectively.

If the intention is to apply in the line creation a pre-defined style, the correct function to use would be the following where *style* is the identifier of that style:

- *DrawLine (SVGWriter filePointer, string id, double x1, double y1, double x2, double y2, string style);*

Finally, VGLib2D allows for the insertion of raster elements, the definition of patterns and gradients, Bézier curves, the use of filters and the definition of geometric areas with or without shading.

Until now no real time animation and interaction classes and functions have been contemplated. The library architecture was, nonetheless, built in such a way as to include this for possible future implementation.

4.4. Application contexts

VGLib2D is intended for the development of all and any kind of application that considers SVG as the suitable format for all information representation and visualisation providing that this information is possible for 2D graphical “translation” (which implies the possibility of using 2D graphical techniques and symbols for the correct representation of the selected information). This means that it can be used in the development of SVG applications in all previously mentioned areas with the added virtue of pushing all SVG creation and management programming to a higher level. Moreover, it is intended for successful optimisation, as new functions and classes are implemented.

VGLib2D was tested and refined during the development of a Web application for the management of the Technical Timetables of Trains for REFER EP, in which it has proved to be crucial in the implementation of the graphical output component.

One of the required components of the application implied the visualisation of Circulation Graphics, which are technical maps where the circulation of a given number of trains is represented along a given time interval and a sequence of stations. These maps are intended to respect current timetables, their validity period (the temporal period for the circulation of the trains) and their frequency regime (the week days on which they circulate) among other relevant issues. It was particularly in this module that the VGLiv2D was used,

with the main purpose of generating Circulation Graphics in SVG format.

This application, accessible to the user through a simple Web browser, presents a Client-Server architecture based on the Asp.Net technology and on a SQL (Structured Query Language) relational database, as can easily be seen in the schematic representation in Figure 6.

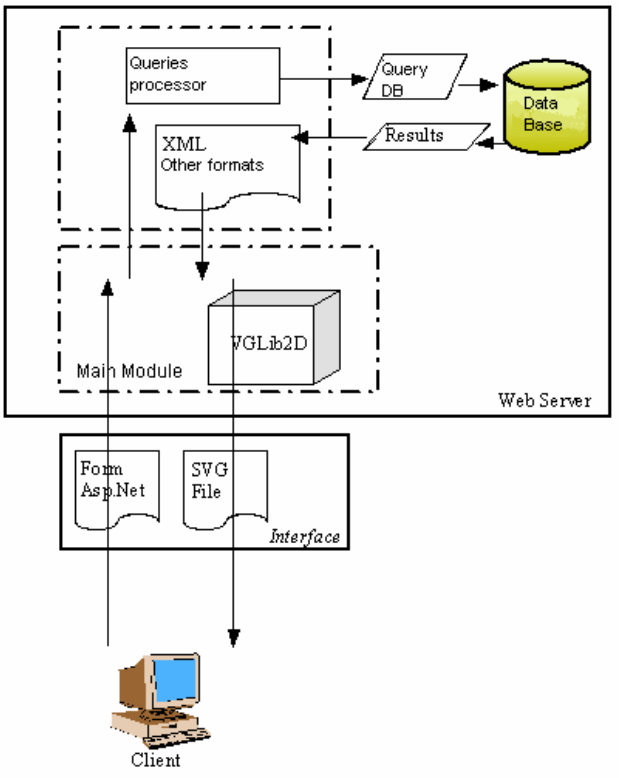

Figure 6: Circulation Graphic Module Architecture

The user interface consists of a Search Form (Asp.Net form) where the station lines and respective stations intended for consultation should be selected along with the daily/monthly time interval, temporal hour interval, train validity period, frequency regime, etc. (see Figure 7).

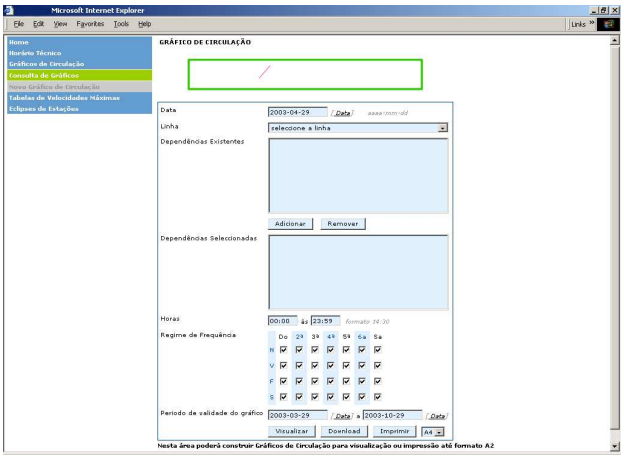

Figure 7: Search Form

Next, the application accesses the database through Data Queries corresponding to the data introduced by the user and it returns all trains presenting the requested characteristics, plus all attached data. Each train is defined by a set of generic information (first identification number, second identification number, circulation start date, circulation end date, date of train insertion on the database, description, operational identifier, commercial identifier, etc.) and a set of nodal or station information corresponding to the path of the train (station identifier number, line identifier number, arrival hour, departure hour, stop time, etc.).

Both the generic information and the nodal specific information are very complex and the number of trains can, in case of a long search time intervals (e.g. around one year), reach two thousand. Thus, all this can result in a very impressive amount of information to represent.

All resulting information is submitted to pre-processing, being, after, graphically "translated" and dynamically represented by the use of VGLib2D. The resulting SVG file is then automatically available for visualisation, printing and downloading.

A train, in its graphical representation, is no more than a line going through a specific set of stations (vertical axis) in a determined period of time (horizontal axis). In this way functions from the VGLib2D were used for the creation of polylines and paths for the representation of the trajectories of trains through the several stations of their routes. Additional objects were used for the legends, once that all trains have been associated with the generic information mentioned above. The use of colours is also a must, as it works as a classificatory element for different types of trains.

It is important to say, nonetheless, that the actual Circulation Graphics only take advantage of a very small parcel of SVG elements and have the use of a much reduced subset of its potentialities.

Besides visualisation, Circulation Graphics are specially oriented for printing purposes. In fact, this was one of the main requirements to be fulfilled, once that its visualisation and analysis on screen is not satisfactory due to the complexity and large amount of information frequently present in each graphic. Therefore, it was necessary to assure the possibility of creating large dimension SVG files, which range from A4 size (14,8cmx21cm) to A0 (84,1cmx118,9cm), which are more suitable for printing by special plotters.

Although the creation of SVG files of these dimensions presented no problem, their printing was only made possible up to A2 size (42cmx59,4cm) due to the limitations of the plug-in, which forced the printing to be made via a browser. It is foreseen that this limitation will be overcome with the new version of the plug-in that promises to offer new printing possibilities.

Relevant factors for the use of VGLib2D were the facts that the desired application was to be a Web application and that .Net platform was selected as the development platform.

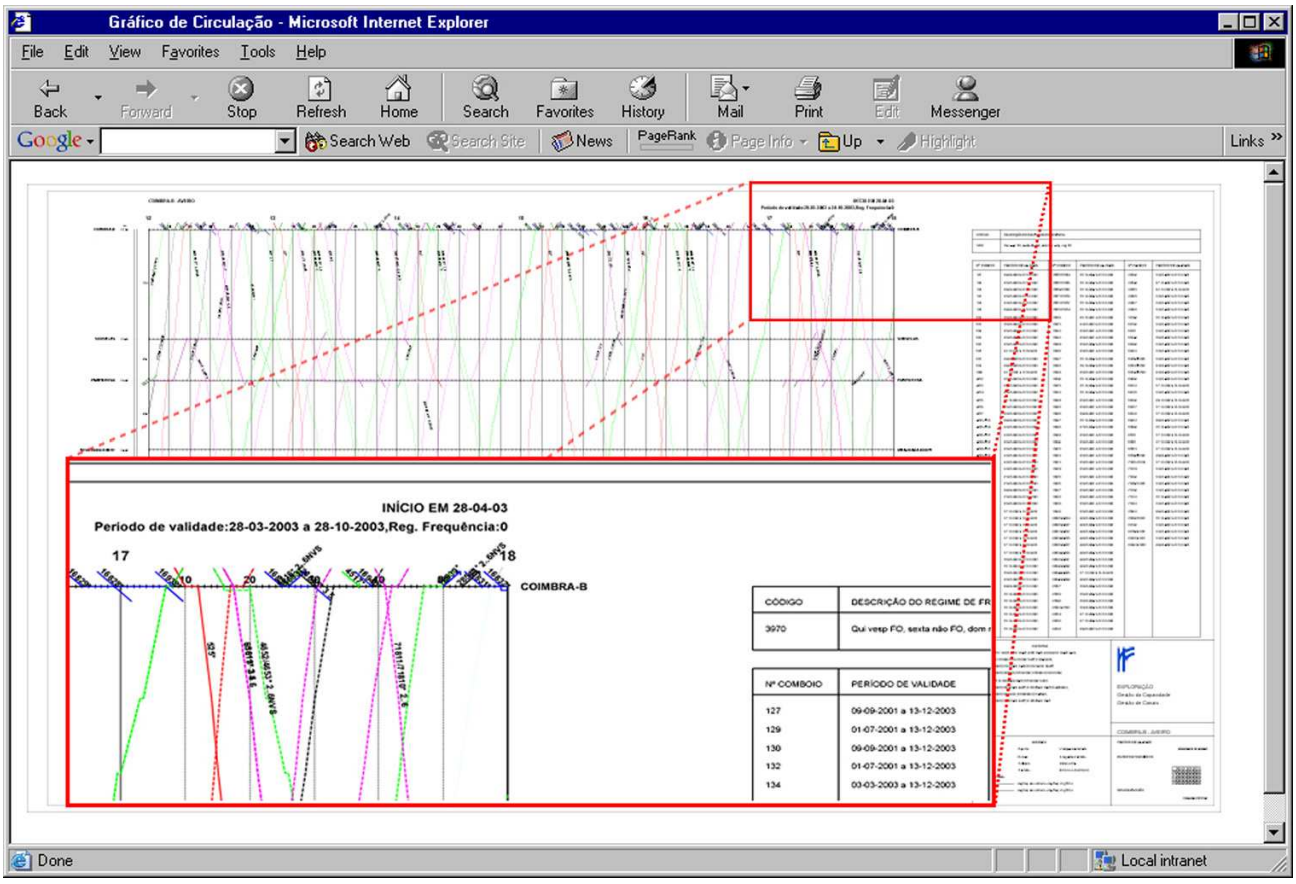

Figure 8: Example of a Circulation Graphic

There was, nonetheless, one other intention in the choice of SVG as the adopted format for the graphic representation of information: on one hand, with SVG, and by using the group element, it is possible to include some options for the interactive visualisation of the information, such as to hide, highlight or visualise only specific subsets of information from the total presented in the SVG file. This type of function can be compared to the use of layers, which is a visualisation technique widely used to allow for the separation and grouping of all available geometrical information in isolated layers, which makes its reading and interpretation easier.

On the other hand, SVG format allows for the future implementation of dynamic Circulation Graphics in which the user can see, in real time, the circulation of trains. This kind of approach fits into a recent perspective in which the traditional notion of documents as being mainly composed of static entities has been changing in order to allow for the constant insertion of dynamic informative artefacts. These are the so called Live Documents, where, with the new emerging technologies such as SVG, there is a real chance for enriching informative documents with multimedia components, thus allowing access to additional information, interactivity and co-operative work [Weber, 02].

One other aspect that led to the use of VGLib2D was the great amount of information being manipulated and graphically represented. As stated above, some of the Circulation Graphics can present information related to more or less two thousand trains. This results in files with a highly complex DOM, which are subject to a considerable set of optimisation measures. Just the same, in comparative terms and serving as an example, it is important to indicate that the document of Figure 8 with respect to what can be considered to be readable dimensions (in such complex maps where the range of time intervals exceeds 12 hours, it is considered that a paper size inferior to A2 is not legible) and with a suitable printing quality (150dpi), in the raster formats JPEG and TIFF present, respectively, 1674KB and 41769KB of information, while in SVG format it only requires 210KB. The choice of SVG as the output format to the VGLib2D, has also proved to be of crucial importance in this matter.

Finally, it is important to say that SVG files can be compacted into the SVGZ format with the simple use of the Gzip tool [Gzip, 03] and, in the same way, they are suitable for interpretation and visualisation by the plug-in.

5. CONCLUSION AND FUTURE WORK

VGLib2D has proved to be of great utility in the implementation of SVG-based applications oriented towards the 2D graphical representation of large volumes of data.

It has filled the gap present in the .Net Framework with respect to the availability of a higher level interface for SVG document generation. The real case presented in this article has reinforced this idea, just as it has highlighted once again the great flexibility of the SVG format and, consequently, confirmed the idea that this was the most suitable output format for VGLib2D.

As far as future work is concerned, one of the main issues for consideration is to provide VGLib2D with support functions for an extension of SVG, the Constraint Scalable Vector Graphics (CSVG), that allows higher flexibility in the description of the SVG graphical elements. With CSVG, an image can contain objects in such a way that its properties and position are specified in relation to other objects and not in absolute terms. CSVG allows the delay of the layout of the defined objects in the document until the rendering is done by the client application, which results in a higher flexibility in use of different visualisation devices [Badros, 01]. In this way, one of the biggest limitations of SVG will be overcome.

In the further distance, the authors predict the implementation of new classes for animation and real time interaction functionalities.

6. ACKNOWLEDGMENTS

Special thanks are due to Rede Ferroviária Nacional - REFER EP, Meticube, Lda. and noLimits Consulting, S.A. who collaborated with the Centro de Computação Gráfica by implementing, among other components, VGLib2D in the development of the circulating material information management platform now in daily use by REFER EP.

7. REFERENCES

[Adobe, 01] Adobe® SVG Viewer 3, 2001. URL: http://www.adobe.com/svg

[Adobe, 03] CML2SVG. URL: http://www.adobe.com/svg/demos/cml2svg/html/index.html

[Bieber, 03] Gerald Bieber, Volker Leck. Scalable Vector Graphics for SaiMotion. *Computer Graphik Topics – Reports on Computer Graphics, 1/2003 Vol.15.~* http://www.inigraphics.net/publications/topics/2003/issue1/1_03a08.pdf

[Badros, 01] Greg J. Badros, Jojada J. Tirtowidjojo, Kim Marriott, Bernd Meyer, Will Portnoy and Alan Borning. A constraint extension to scalable vector graphics. *Proceedings of the tenth international conference on World Wide Web*. April 2001, 489-498.

[CML, 03] Chemical Markup Language (CML™). URL: http://www.xml-cml.org/

[Custom, 03] Custom Mapping System V1.1. Limbic Systems, Inc. URL: http://www.custommapper.com/index.jsp

[Drayton, 02] Peter Drayton, Ben Albahari and Ted Neward. C# in a nutshell – A desktop quick reference. O'Reilly & Associates, Inc. 2002.

[Gzip, 03] The gzip home page. http://www.gzip.org/

[Mackay, 02] Wendy E. Mackay, Guillaume Pothier, Catherine Letondal, Kaare Bøegh and Hans Erik Sørensen. Interaction in the real world: The missing link: augmenting biology laboratory notebooks. *Proceedings of the 15th annual ACM symposium on User interface software and technology*. October 2002, 41-50.

[Marriott, 02] Kim Marriott, Bernd Meyer and Laurent Tardif. XML Applications: Fast and efficient client-side adaptivity for SVG. *Proceedings of the eleventh international conference on World Wide Web*. May 2002, 496-507.

[Microsoft, 03] Microsoft® .Net™ http://www.microsoft.com/net/

[Mouza, 02] Cédric du Mouza and Philippe Rigaux. GIS and the internet: Web architectures for scalable moving object servers. *Proceedings of the tenth ACM international symposium on Advances in geographic information systems.* November 2002, 17-22.

[Probets,01] Steve Probets, Julius Mong, David Evans and David Brailsford. Hypermedia and Graphics 2: Vector graphics: from PostScript and Flash to SVG. *Proceedings of the 2001 ACM Symposium on Document engineering*. November 2001, 135-143.

[Seff, 02] George Seff. SVG and GIS. 2002. URL: http://www.directionsmag.com/article.php?article_id=198

[W3C, 00a] World Wide Web Consortium. DTD - W3C SVG 1.0 Specification - Candidate Recommendation 20001102. URL: http://www.w3.org/TR/SVG/svgdtd.html

[W3C, 00b] World Wide Web Consortium. The Document Object Model. 2000 URL: http://www.w3.org/DOM

[W3C, 03] World Wide Web Consortium. QH D22 P2 SVG Validator: http://www.w3.org/2003/08/qh-d22-p2.htm

[Wahlin, 02] Dan Wahlin. SVG: .Net Graphics Courtesy of XML. *XML & Web Services Magazine.* August/September, 2002.

[WebDraw, 03] Jasc® WebDraw™. URL: http://www.jasc.com/products/webdraw/

[Weber, 02] Anke Weber, Holger M. Kienle and Hausi A. Müller. Live documents with contextual, data-driven information components. *Proceedings of the 20th annual international conference on Computer documentation.* October 2002, 236-247.

[XMLSpy, 03] XMLSpy http://www.xmlspy.com/