Using SVG and XSLT for graphic representation

Andres Baravalle
Department of Computer Science - University of Turin
Italy

Biography

Andres Baravalle is a full-time Ph.D. student at Turin University, Department of Informatics. His actual research topics are artificial intelligence, human-computer interaction, usability. He took his degree summa cum laude in Communications at Turin University. His degree thesis discusses about different technologies for information storage (databases, XML etc.) and about languages and techniques for multimodal web interaction with desktop and mobile users. Economical aspects concerning implementation are also covered. Among his key qualifications he lists a deep knowledge of the main languages and technologies related to web development. He worked at several projects as consultant. Among them he worked to develop multimodal interaction systems based on XML and server side technologies for the web sites costameno.it and loescher.it. Publications "Remote web usability testing: a proxy approach", to be published at HCI2003, Creete. "Remote web usability testing", Measuring Behaviour 2002, Amsterdam. "A usable web for long-stay hospitalised children", HCI2002, London.

Marco Gribaudo
Department of Computer Science - University of Turin
Italy

Biography

Marco Gribaudo is currently a researcher at the Department of Computer Science, University of Turin.

Vitaveska Lanfranchi
Department of Computer Science - University of Turin
Italy

Biography

Vitaveska Lanfranchi is a full-time Ph.D. student at Turin University, Department of Informatics. Her actual research topics are artificial intelligence, human-computer

interaction, usability. She took his degree summa cum laude in Communications at Turin University. Her degree thesis discusses about different languages and techniques for multimodal web interaction with desktop and mobile users. Among her key qualifications she lists a deep knowledge of the main languages and technologies related to semantic web. She worked at several projects as consultant. Among them she worked to develop multimodal interaction systems based on XML and server side technologies for the web sites costameno.it and loescher.it. Selected publications: "Remote web usability testing: a proxy approach", to be published at HCI2003, Creete; "Remote web usability testing", Measuring Behaviour 2002, Amsterdam: "A usable web for long-stay hospitalised children", HCI2002, London.

Tiziana Sandri
Communication Science - University of Turin
Italy

Biography

Tiziana Sandri is currently student at Communication Science, University of Turin.

Table of Contents

Using SVG and XSLT for graphic representation

Introduction

In a world based on communication, the way in which information is presented plays a role as important as the actual content being transmitted.

In many occasions an impressive presentation may be the key for the success of a particular strategy. This is also true for scientific data, obtained from complex experiments or computer simulations. Many software and technology exist to aid the construction of visually appealing presentations. Most of them however, focus on the production of graphs such has function plots, histograms, diagrams and so on. Even though this kind of diagrams are extremely useful for the experts involved in a particular area, the may be very hard to understand for people not as much keen on that particular subject.

For this reason alternative representation forms exists, such as pictograms, where the information is visualized by real images, rather than by abstract representations.

The main problem with the realization of such kind of visualizations is that they usually have to be drawn by hand by the presenter after the final data have been obtained. If, for some reason, a mistake in the data is discovered and the corresponding values have to be recomputed, the pictograms must be redrawn. Also, if the data changes rapidly, the time required to draw them with a pictogram representation may be too much to follow the changes. Usually, to avoid this kind of problems, the user that whishes to use such type of representations has to write ad hoc programs that produce the images starting from the data.

In this paper we present an alternative approaches based on XML, XSL and SVG that can help the realization of pictograms for rapidly changing data.

Data are written in a XML using a very simple name-value representation, while a stylesheet, written in XML, specifies the way in which the pictograms are to be constructed. The data and its pictogram style definition file are combined by an XSLT stylesheet, to produce an SVG representation of the final image.

The proposed approaches as several advantages:

- Interoperability. The use of standard technologies such as XML, XSLT and SVG makes the approach portable on different platforms with a small effort.
- Efficiency. The time required to update a pictogram due to the change of data is comparable

to the one achieved using special purpose software.
- Portability. The SVG image produced may be easily included into many form of presentations, such has texts, slides and web sites (in this case, even on the fly), and are compatible with the majority of authoring tools.

State of the art

Scientific visualization has become a strategic issue in human-computer interaction: in many situations the main problem is not to gather experimental data, but to interpret and display the results in a suitable form [1] .

Raw data are often difficult to be interpreted by the intended audience of a research, or they are simply not as fancy as the author would like. Converting raw data into pictures and animations, easily understandable and readable, while seems not adequate to some authors, it is a pragmatic approach that can be a good choice in many cases, as a more appealing presentation can be the method to convey the results of a research to a wider audience.

Unfortunately the interest of software companies in scientific visualization for research purposes seems to be low, probably because the target of users is restricted and often unwilling to invest economical resources for the presentation of their work.

Undoubting generic office applications, as Microsoft Excel (included in Microsoft Office) or Calc (included in OpenOffice) are the most used software for graph representation, at least between non-specialized users. This kind of software targets non-specialized users, both for the number of features and for complexity that can handle. Researchers need a more specialized and powerful software and can sacrifice the simplicity of use of the software for visualizing information in the most suitable way.

Since the visualization job does not constitute only the final phase of a research, but it is also a fundamental element of every stage of the experimentation, is particularly important that also the most complex visualizations can be managed in simple way and real time.

These features have been implemented in softwares like Vis5d, Aspen 2000, Por 2000.

Vis5d is a software developed by SSEC (Space Science and Engineering Center) of the university of Winsconsin-Madison; it offers the possibility to animate in real time and to visualize large amount of data in 3D. Aspen 2000 is a software for the analysis of bulkheads and walls that allows to precisely visualize cross section with elevated degree of detail and to edit all the graphical details. Por 2000 is a software for planning and verification of buildings; it allows graphical modification of the elements and provides visual control for editing elements.

All these software allow to collect data, to analyse them and then to create bidimensional or three-dimensional graphs: the graphs can be bubbles, histograms or cakes but they are all abstract diagrams that do not have any connection with the nature of the data.

The diagrams produced adapt to whichever type of numerical data but do not provide an intuitive point of view over the described data: the true nature of the described data it is not immediate or easy to understand.

Otherwise in meteorological and oceanographic fields several applications of scientific visualization have been developed. Researchers in these fields deserve clear and complete comprehension of studied phenomena, need a platform able to handle a huge quantity of data, analyzing them through appropriate visualization techniques.

Moreover it lets visualizing a lot of different variables at the same time, without deterioratating the quality of the produced images, and without increasing the complexity.

Some reserachers suggested to adopt XML based language to semantically describe complex data **[2]** : since XML is a powerful language that allows creating custom tags and conveys a semantical values to the contained data, it is often use to describe data that can be difficult to understand, as chemical or mathematical formula **[3]** **[4]** .

In mathematical or chimical fields the visualization of the prodoced results it is very important **[5]** : some recent reseraches proposed to adopt an XML dialect for describing data and to transform them in other XML based language, specialized for visualization as VRML **[6]** or SVG.

SVG has been used as visualization language also for different type of data, as Census Data **[7]** , reverse engineering data **[8]** or medical data **[9]** .

In this paper we present an XML based framework that can be used to produce graphical visualisation of scientific data. The approach rather than producing ordinary histogram and function diagram graphs, tries to represent the information in a more graphical appealing and easy to understand way. The proposed framework is able to maintain the value of the data strictly separated from the visual form of its representation (positions of element, colors, visual representation etc.).

Since XML can be used for describing complex data information, we represent every level of the graphic representation with an XML structure.

To describe our architecture we defined the following XML dialects, each one with different markup tags, reflecting the semantical values of the elements.

- **Data definition level**. Used to define the value of the data that can be used in the graphic representation.

- **Data representation level**. Used to define the graphic representation, it defines how the values expressed by the data definition level are represented.

Both data representation and data definition files are based on a DTD to impose the constraints.

XSLT is then used to output a SVG file derived from the two files describing the graphic representation.

Visualization model

Data representation level is the core of the system, and defines a powerful language to represent:

- **Source primitives**. Used to define for the source of the graphic elements, for example bitmap image files or vectorial SVG images.
- **Modification primitives**. Used to define the modifications that can affect a graphic element, for example rotation, scaling or repetition.
- **Disposition primitives**. Used to define the possible dispositions along x, y and z axes, for example to impose a order in the representation of elements.
- **Action primitives**. Used to define the possible actions that can be activated by graphic elements for different user behaviors. For example a mouse action can activate a link to a different resource, or can change the value of any of the other primitives of the data structure, as image source or disposition, or can show a tooltip.

Source primitives

The main idea is that numeric data can be easily understandable if graphically represented in a well-known context.

Then we decided to associate to every numeric data an image that visually represents a contextual reference to the described subject. Dynamic images, superimposed on the static image, help representing variations of numeric data.

It is very important to choose an adequate image that can be intuitively understood by any user, according to her cultural background. A knowledge of the user cultural background and, if possible, usability tests, can help to decide which graphical elements to use.

In the example below we wanted to represent the lowest and the fastest speed of a car.

Figure 1: Speddometer

We choose to represent the speed of the car using a static image of a speedometer: the lowest and fastest speed are rendered using two dynamic hands.

Modification primitives

Elements are provided to describe the possible modifications of the images, such as scaling, rotating and skewing.

This lets modify basic properties of the images, in order to render the relation between them and therefore between the data easy andmeaningful.

The most intuitive operation is to change width and height of the images in order to represent an increment of a phenomenon. The example below represents a cloudy sky.

When the weather conditions change, it is possible to represent this variation simply widening the image of the cloud over the sun:

To define these operation we used a mathematical formula that establish in which way the selected dimension change proportionally to the variation of the numeric data:

$h = s(d-c)$

where h represents the dimension to be modified (height, width, rotation angle), *s* is the scale, the constant used to multiply the dimension; *d* represents the image dimension, *c* represents the starting point of the represented unit of measure. The formula is stored in our formalism.

Disposition primitives

Disposition primitives permit to repeat images or to put them in a sequence, for example from the smaller to the bigger data, or from the older to the newer data.

Action primitives

Actions primitives permit to associate responses to specific user behaviors.

For example it is possible to underline or highlight an image or part of it, or to add tooltips and links.

Implementation

The proposed approach may be implemented in several different ways:

1. Using a high-level programming language: a software tool that allow to visually build the pictograms, starting form the data, may be written in general purpose programming languages that support XML such as Java or C++. The tool may use the proposed XML document to store the style definitions and to import the data.

2. Using the scripting languages of some advanced graphic format: for example, an XML aware browser plug-in, such as Macromedia Flash, can be use to read both the data and the pictogram style XML specification. Using the scripting language of the graphic format, the pictogram may be generated by parsing and interpreting both files.
3. Using a generic XML transformation process to create a graphic file in a suitable format: for example, we can use XSLT to produce an SVG representation of the pictogram obtained by combining the data with its style.

Each of the alternative has its own advantages and disadvantages. Using an interpreter (such as in case 1 and 2), the interactive features (such as the one defined by the Action Primitives) may be easier to implement. Also, high-level programming languages are the one that can provide the most efficient implementation. However, an interpreter in a programming language (case 1) is more expensive to develop than a script in an advanced graphic format (case 2), which in turn is more expensive to develop and debug than a text transformation specification (case 3). XSLT transformation may be easier to implement server side, and simplify the distribution of the produced pictograms.

We have tried to prototype several implementations: a Flash movie that can read data and style specifications and represent them, and an XSL style-sheet that combine them to produce an SVG output. In this paper we will concentrate on the latter.

We decided to adopt SVG because it meets two main requirements:

- SVG is data oriented and XML-based: it is simple to generate an SVG image file from a native XML.
- SVG is interactive: it allows to manipulate data and to associate responses to specific user behaviors.

The transformation of XML representations of scientific data into SVG is made throught an XSLT stylesheet.

For every defined primitive we created a XSLT transformation that converted XML into SVG code:

- Source primitives are converted into SVG <img> elements.
- Modification primitives arecimplemented through the use of transform attribute, with rotate, scale or matrix values.
- Disposition primitives that allows repeating images are implemented through the use of the element <pattern>.
- Action primitives are implemented adopting the EcmaScript language.

Examples

In these chapter we present some examples realized simulating real situations.

The first example concerns measuring pressure in weather forecast.

We want to represent the maximum and minimum temperature in a city.

Below is listed the XML file that defines numeric values:

```
<?xml version="1.0" standalone="no"?>
<?stile stiletemp.xml ?>
<!DOCTYPE datafile SYSTEM "data.dtd">
<datafile>
        <data>
                <value id="tempmin" num="2" />
                <value id="tempmax" num="6" />
        </data>
</datafile>
```

We choose to represent the contextual static image with a thermometer and the dynamic data with two vertical lines, blue for the minimum, red for the maximum.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE stile SYSTEM "astile1.dtd">
<stile>
        <image src="termometro.svg" imgx="20" imgy="20" />
                <addimage addsrc="tempmin.svg" addimgx="30" addimgy="500" />
                <addimage addsrc="tempmax.svg" addimgx="30" addimgy="500" />
        <upimage link="tempmin" imageref="tempmin.svg" center="0" scale="5" vali
        <upimage link="tempmax" imageref="tempmax.svg" center="0" scale="5" vali
</stile>
```

We used the element <upimage> to represent the direction of the dynamic images (their numeric values are positive numbers).

Below is the resulting image:

The second example concerns measuring human corporeal pressure.

The data file contains the systolic and diastolic pressure values:

```
<?xml version="1.0" standalone="no"?>
<?stile astile.xml ?>
<!DOCTYPE datafile SYSTEM "data.dtd">
<datafile>
        <data>
                <value id="pressmin" num="75" />
                <value id="pressmax" num="120" />
        </data>
</datafile>
```

We choose to represent pressure trough an analogical sphygmomanometer, and the maximum and minimum values are represented with two hands that rotate according to their numeric value. When a user clicks on the image, a web page containing further data is visualized.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE stile SYSTEM "stile.dtd">
<stile>
        <image src="sfigmo1.svg" imgx="30" imgy="30" />
                <addimage addsrc="lancettablu.svg" addimgx="75"
                addimgy="175" />
                <addimage addsrc="lancettarossa.svg" addimgx="75" addimgy="175"
                <rotateimage center="0" scale="2" link="pressmin" imageref="lanc
                        xcenter="75" ycenter="175" />
                <rotateimage center="0" scale="2" link="pressmax" imageref="lanc
                        xcenter="75" ycenter="175" />
        <infodata url="press.htm" imageref="pressmin" />
</stile>
```

The last example regards the number of university students that actively follow a course.

An XML file represents the number of students associated to the different courses:

```
<?xml version="1.0" standalone="no"?>
<?stile astile.xml ?>
<!DOCTYPE datafile SYSTEM "data.dtd">
<datafile>
        <group>
                <gdata gid="1corso" gnum="60" />
                <gdata gid="2corso" gnum="100" />
                <gdata gid="3corso" gnum="70" />
                <gdata gid="4corso" gnum="120" />
        </group>
</datafile>
```

We decide to represent the context with the image of a lecture hall and to represent the different number of students with the same image of a student but differently scaled.

We also choose to highlight the highest frequency by surrounding the correspondent image with a green rectangle.

Below is the resulting SVG code:

```
<?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 xmlns="http://www.w3.org/2000/svg" width="350" height="400">
        <image xlink:href="aula.gif" width="300" height="200"
                transform="scale(0.666667 0.68) translate(12 35.2941) translate(
                />
        <image xlink:href="stu.gif" width="50" height="90"
                transform="translate(88.3333 110) translate(211.667 -3.33333) tr
                />
        <image xlink:href="aula.gif" width="300" height="200"
                transform="matrix(1 0 0 1 100 150) translate(-95 58.3333) scale(
                />
```

```
        <image xlink:href="stu.gif" width="50" height="90"
                transform="matrix(1 0 0 1 225 205.667) translate(6 150) translat
                />
        <image xlink:href="aula.gif" width="300" height="200"
                transform="matrix(0.633333 0 0 0.71 155 179.334) translate(-224.
                />
        <image xlink:href="stu.gif" width="50" height="90"
                transform="matrix(1.64 0 0 1.44444 210 185.668) translate(12.19
                />
        <rect x="213" y="333" width="142" height="160"
                 style="fill:none;stroke:rgb(64,255,64);stroke-width:3"
                transform="translate(1 493) scale(0.764344 0.764344) translate(-
                />
        <polygon
                points="303.439,105 327,119 303.439,133 303.439,124.49 285,124.4

                transform="rotate(228.885 306 119) translate(0.657568 -0.753395)

                 style="fill:rgb(255,255,255);stroke:rgb(0,0,0);stroke-width:1"/
        <rect x="319" y="113" width="142" height="82"
                 style="fill:rgb(255,255,192);stroke:rgb(0,0,0);stroke-width:1"
                transform="translate(390 113) scale(1 0.768293) translate(-390 -
                />
        <rect x="1" y="5" width="490" height="484"
                 style="fill:none;stroke:rgb(0,0,0);stroke-width:1"
                transform="translate(1 493) scale(0.764344 0.764344) translate(-
                />
        <text x="326px" y="129px" style="fill:rgb(0,0,0);font-size:10;font-famil
                transform="translate(1 493) scale(0.764344 0.764344) translate(-
                >stu.svg</text>
        <text x="324px" y="147px" style="fill:rgb(0,0,0);font-size:10;font-famil
                transform="translate(1 493) scale(0.764344 0.764344) translate(-
                >x:80 y:120</text>
        <text x="324px" y="164px" style="fill:rgb(0,0,0);font-size:10;font-famil
                transform="translate(1 493) scale(0.764344 0.764344) translate(-
                >number of student:60</text>
</svg>
```

Conclusions

In this paper we presented an approach to produce pictogram representation of numerical data obtained, for example, from scientific computer programs. How a pictogram has to be drawn is specified using an XML based language, and the data to be represented itself are specified using XML. An implementation using XSLT to produce SVG representation of the pictogram has been proposed and several examples have been presented to prove the applicability of the technique. Future direction include the extension of both XML dialects to include more sophisticated data types, such as tables and non numerical information. Also we plan to implement the approach using different technologies, such as Java, to allow the graphical construction of the pictogram styles as well as their visualisation.

Bibliography

[1]

Domik, G. *Scientific Visualization*. In: Proceedings of Ed-Media 93, Orlando, AACE, Charlottesville, VA (1993), 153-160.

[2]

XML for visualization. http://www1.bcs.org.uk/DocsRepository/03700/3784/brodlie.pdf

[3]

Carlisle, D. (editor) *Mathematical Markup Language (MathML) Version 2.0.*http://www.w3.org/TR/2003/WD-MathML2-20030411/

[4]

Murray-Rust, P. *Chemical Markup Language*. World Wide Web Journal, 1997, pp 135-147.

[5]

Ihlenfeldt Wolf-D. and Engel Klaus. *Visualizing Chemical Data in the Internet -- Data Driven and Interactive Graphics.* Computers and Graphics, 1998, Vol. 22, No. 6, pp. 703-714.

[6]

Arun, B.& Ganguly, A.D. *XML To Be, VRML To See.* http://www.acm.org/crossroads/xrds6-2/xml2b.html

[7]

Chang, Yi-Hong & Chuang, Tyng-Ruey . *Online Aggregation and Visualization of Census Data: Population Mapping with SVG, XML, and Free Software.* In: Proceedings of SVGOpen 2002, Zurich, Switzerland.

[8]

Kienle,H.M., Weber,A., & Müller, H.A. *Leveraging SVG in the Rigi Reverse Engineering Tool.* In: Proceedings of SVGOpen 2002, Zurich, Switzerland.

[9]

Lewis, C.T., Karcz, S., Sharpe, A., Parki, I.A.P.*BioViz: Genome Viewer. Development of an SVG GUI for the visualization of genome data.* In: Proceedings of SVGOpen 2002, Zurich,

Switzerland.