OpenedEyes: A framework for information visualization based on open web technologies and web standards

Caio Sacramento de Britto Almeida[1], **Antônio Lopes Apolinário Júnior**[1]

[1]Universidade Federal da Bahia (UFBA)
Instituto de Matemática – Departamento de Ciência da Computação – Salvador – BA – Brazil

{caiosba,apolinario}@dcc.ufba.br

Abstract. *Extract knowledge from a dataset is not trivial. Information visualization shows up to help on this task by trying to optimize the user information gathering using suitable graphical representations. Many of the existing tools for this task use proprietary technologies or are proprietary software themselves. The proposal of this work is to present OPENEDEYES, a free (as in freedom) JavaScript framework that uses only web standards and open technologies to implement interactive information visualization applications to be run in a web browser. From the framework core, four visualization modules were developed, implementing traditional and new techniques that confirmed the flexibility and extensibility of the framework, which can be used to analyze any kind of abstract data, including data about free software projects.*

1. Introduction

Advances in technology allow computers nowadays to store a large amount of data. Researchers from the Berkeley University estimates that every year 1 exabyte of data is produced, most of them available in digital format [Keim 2002]. Being a valuable source of information is a promise that such datasets carry, while extracting these informations is often a hard task.

There are tools developed to help on data analysis. Visual exploration involves directly the user and has an advantage over other methods that is to be intuitive and able to handle easily heterogeneous data. As a consequence, they often lead to faster analysis and better results [Keim 2002]. Visual representations and interaction techniques take advantage of the human eye broad bandwidth pathway into the mind to allow users to see, explore, and understand large amounts of information at once [Thomas and Cook 2005].

Regarding interactive visualizers, the proprietary technology Adobe Flash [Adobe 2011] is used by some tools. But Flash goes against the grain of the open standards since it requires a plug-in to be installed by the user, which is considered a usability and accessibility problem, among other implications [Nielsen 2000]. On the other hand, the web standards were born as recommendations proposed by the World Wide Web Consortium (W3C) as an attempt to promote the Internet's evolution and to guarantee that all web technologies work well when joined. For years, W3C referred to these specifications as recommendations, and this may have contributed to a weak adoption by the companies that developed web browsers. When the Web Standards Project [Project 1998] was launched, in 1998, those recommendations were renamed to web standards, and then the support to those standards became a vital ingredient to any browser or Internet device.

The web browsers modernization allows web standards-based rich internet applications to be built, even visualization tools [Kosara 2011].

The need to extract knowledge from a large dataset using free (as in freedom) tools motivated the development of a web-based information visualization framework using only web standards and open web technologies. The proposal is that this framework is composed of many interactive visualization modules, each one suitable for a certain kind of dataset, and also offers the basic infrastructure that allows it to be extended with the development of new visualization modules. As a result, it was possible to develop a fully functional framework with four visualizers developed upon it. Those visualizers are completely different since they implement different techniques suitable for different numbers of dimensions. The bubble chart module supports up to four dimensions, the great circles map represents relationships between geographic-related data, the choropleth map allows a variable to be compared across regions on a map, and finally, the parallel bars chart, an implementation of a new technique purposed by this work, allows n dimensions to be represented in a bidimensional space. The following sections will present related works, the reference model, the framework and the results.

2. Related works

Perhaps the first framework for information visualization was the Information Visualizer [Card et al. 1991]. Later, Microsoft Excel was released, and it also generates charts based on a dataset, but beyond running on desktop, it's a proprietary software. Regarding end user visualization applications, Tableau [Tableau 2012] and Spotfire [TIBCO 2012] are desktop applications which offer advanced information visualization tools to analyze large amount of data.

Many visualization tools were developed in the recent years to allow developers build their own visualizations. Among the Java-written ones, Processing [Fry and Reas 2011] is a visual programming language and environment for that ones who want to create images, animations and interactions but works in a lower-level since it requires some programming skills. Besides this, there are many possibilities to build complex visualizations. Prefuse [Heer et al. 2005] defines itself as a toolkit for interactive visualizations creation. The Prefuse framework provides both Java and Flash tools, is freely licensed, but its strength is tree and graph structures besides also requiring coding. ManyEyes [Viegas et al. 2007], written in Java, is a website which offers a service that allow users to upload datasets, create visualizations, publish and comment, but can not be downloaded and used as a product, just as a service.

Fusion Charts [FusionCharts 2011] is also another Flash tool for interactive visualization, which is largely used worldwide, including lots of visualization techniques, most of them traditional and simple techniques such as pie, bar and line charts. Besides not including complex visualizations, it's a paid product. Part of its code is written in JavaScript and HTML 5.

Among the JavaScript solutions, most of the existing solutions also work in a lower-level. Raphael [Baranovskiy 2011] makes it easy to create cross-browser SVG [W3C 2011d]. ProcessingJS [Resig 2011] is a JavaScript port of Java Processing, so it works in a similar way as the Java tool. The solutions for charts creation are many, but include only simple charts and most of them can't be extended.

As exposed above, the existing tools can be divided into two main groups. In one group, web applications that implement simple techniques or require advanced knowledge. In the other group, desktop applications that implement more advanced techniques. Both groups could also be divided into free software and non free software. So, OPENEDEYES would be placed in an intersection between those two groups since it is web based, free and implements advanced techniques without requiring advanced knowledge. This way, lots of features are joined in only one product. It's written in JavaScript and based on web standards, avoiding proprietary technologies such as Flash or non-standard technologies such as Java. It runs on web browsers, making it available for all platforms. Works on both lower level and high level, so users who have programming skills can extend it by developing new visualizers, while novice users can just generate a visualization for some dataset by using one of the available visualizers. It's full available for download and licensed as free software, so anyone can download it, modify and use. Advanced techniques are available, such as a new technique introduced by this work called *parallel bars*. Conceptually, OPENEDEYES was based on the reference model to be presented in the following section.

3. Reference model

Some components in the information visualization process are common to many visualization techniques, regardless the data domain. These components are present in reference models [Card et al. 1999, Chi 2000, Haber and Mcnabb 1990], which serve as a conceptual framework for structuring infovis applications [Heer et al. 2005]. This work is based on the reference model shown in Figure 1 [Mazza 2009], modified to show how each component is mapped to a web technology.

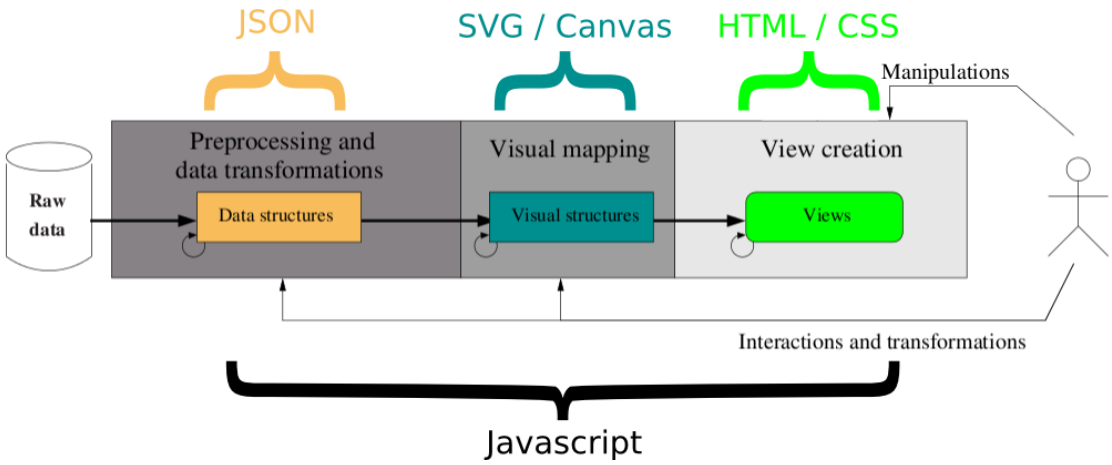

Figure 1. Adapted reference model for this work based on [Mazza 2009]

The process begins with the raw data. Pre-processing turns data into a JSON [Crockford 2006b] structure. These JSON data tables can be static (stored in a file system) or dynamically returned by a data server. Data will be graphically represented as visual structures, by a process called visual mapping. Visual structures in OPENEDEYES are SVG elements (vectors), or a Canvas bitmap (HTML 5 element) [W3C 2011c]. Finally the views are created, which are HTML documents [W3C 2011c] formatted using CSS [W3C 2011a]. The transitions between each stage of this sequence is done by JavaScript,

as the user interactions after the view creation. These interactions may not perform new requests to the server.

4. The framework

Regarding its extensibility, depending on the use case, OPENEDEYES can behave as white box or as black box. White box frameworks, or architecture driven, are those where instantiation is achieved through adding new classes. These classes should be included in the framework source code, so it requires some level of knowledge about the framework internals. On the other hand, black box frameworks, or data driven, are those that do not require much knowledge about the framework internals as instantiation is achieved through defining configuration parameters [Markiewicz and de Lucena 2001]. OPENEDEYES behaves as a white box framework when a developer creates a new visualizer for it and as a black box framework when a user creates a new view for some dataset using one of its visualizers. The OPENEDEYES execution flow is presented in Figure 2.

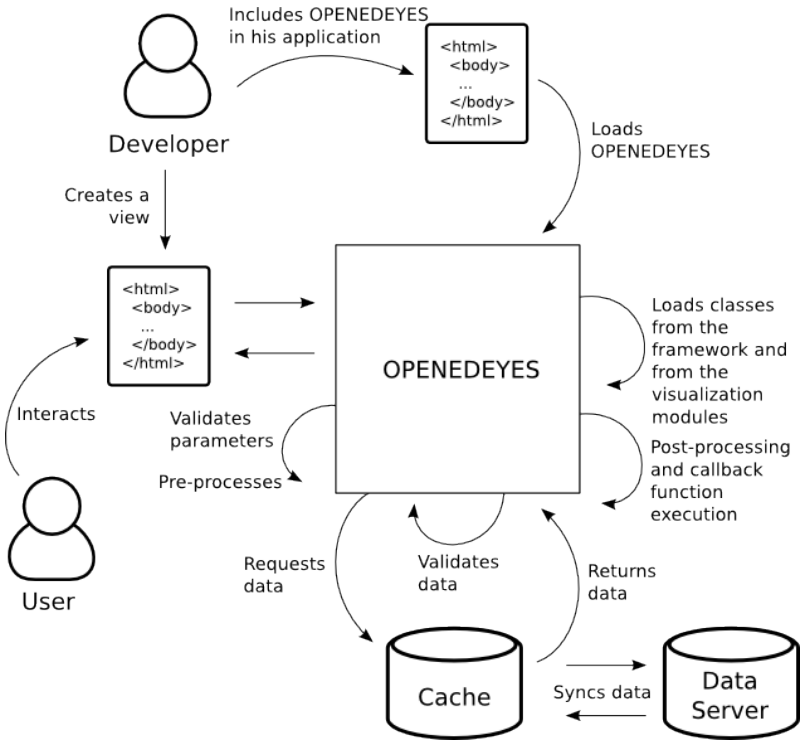

Figure 2. Global execution flow of OPENEDEYES

The developer - responsible for creating the visualization - includes OPENEDEYES in his application and defines the configuration parameters and interface controls. The user - responsible for understanding some dataset from the visualization - uses those controls to interact with the view generated by OPENEDEYES. The framework is responsible for handling the current state of visualization, current values of configuration parameters, data request and data preprocessing. Interface controls added by the developer communicate with this visualizer by message passing, calling public methods provided by OPENEDEYES. These methods allow changing configuration pa-

rameters, setting which variable is represented on each dimension, enabling and disabling dimensions, filtering the dataset, zooming in and out, and so forth.

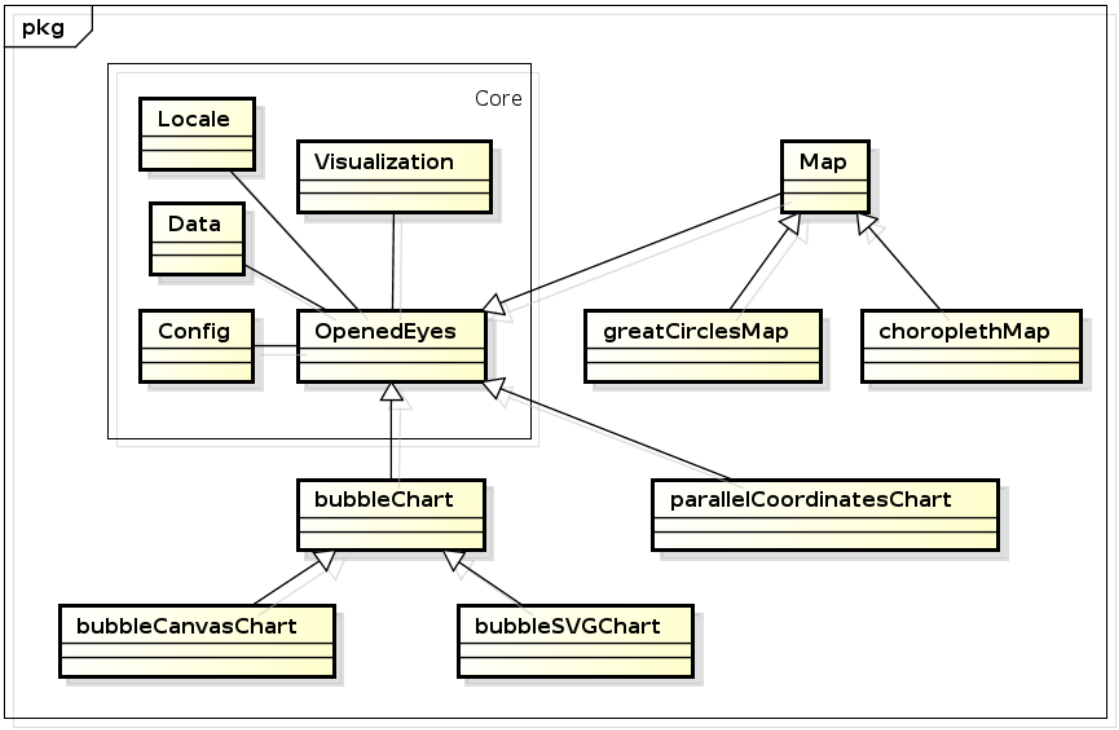

Figure 3. Simplified OPENEDEYES class diagram

A simplified OPENEDEYES class diagram is represented in Figure 3, hiding the methods. Five classes (wrapped in the rectangle in Figure 3) form the framework core. The other classes form the visualization modules set. This set is not limited since it includes only the visualization modules already implemented in this work as an attempt to show the expansion capabilities of the framework. A new visualization module could be implemented by creating a new class that inherits from the OpenedEyes class. The core classes are:

- Data: Manages input data. As mentioned previously, the data table for OPENEDEYES is a JSON structure, that can be returned by AJAX [Garret 2005] or by JSONP [Ippolito 2005]. The choice for JSON instead of another popular format, such as XML [W3C 2011b], is that JSON is light-weight (faster and cheaper to be transfered through network), easy to be read or written (can be manipulated in lots of programming languages, making it easy to be sent by a data server), subset of JavaScript (language on which OPENEDEYES is written), among other reasons [Crockford 2006a]. Tests also suggest that JSON requires less resources and is faster than XML [Nurseitov et al. 2009]. XML has the advantage of being a format for universal representation of data, simple, human-readable and self-explained [Nurseitov et al. 2009], advantages not so important as the JSON advantages, in this work.
- Visualization: Defines helper methods for visualization. Visual structures can be elements from an SVG document or part of a bitmap drawn on a Canvas element.

The main differences between those two approaches is that SVG, while being actually XML, represents each scene element independently as an XML element, which allows this element to be controlled, what usually results in more smooth transactions between two states of the visualization. On the other hand, when many visual structures are present, the SVG approach can lead to performance failures, and so in this case Canvas might be a better choice. Since it's a single bitmap, it can represent complex visualizations, although it does not allow each scene element to be controlled independently, what means that any change on the visualization requires a full redrawing of the scene.

- Config: Manages configuration parameters.
- Locale: Handles translations.
- OpenedEyes: It's the main class of the framework.

The visualization modules are interactive and implement non-trivial techniques. Multiple inheritance and abstract classes are used to allow extension of the framework. For example, classes OpenedEyes, bubbleChart and Map are abstract classes, but Map and bubbleChart inherit from OpenedEyes. The Map class defines tasks that are common to many geographic visualizations, while bubbleChart calculates values for visual structures allowing its child classes to represent them in different ways. Another visualization module implemented is a variation of the parallel coordinates chart [Inselberg and Dimsdale 1990], proposed by this work, that consists in replacing each line for a polygon which height is proportional to the represented value. Each polygon represents a subset of the input dataset instead of a single element. The following section will show a case study that was made using these visualizers with real data.

5. Results

The following subsections explain each visualization module developed in the scope of this work.

5.1. Bubble Chart

The Bubble Chart module implements a variation of the scatter plot, where data points are replaced by bubbles that define a new dimension. In a bubble chart, each element is represented by a bubble which area is proportional to some variable from the dataset. In this implementation, each bubble has four visual properties (color, size, horizontal position and vertical position), so this chart may have from one to four dimensions (since any dimension can be disabled at any time). There are two classes,the first one uses SVG and so offers a richer experience as any modification in the chart is reflected in a smooth, animated way. The other one uses Canvas and so might be a better choice when the input dataset has lots of elements.

Each OPENEDEYES visualization module includes a default interaction interface, although new controls can be added. This default interface has four select menus where each of them defines which variable will be represented in each dimension. It's also possible to filter the dataset by clicking on some bubble, that will be removed and all the chart will be redrawn.

On the example on Figure 4, each element from the dataset is represented by a bubble. There is an interactive timeline that animates the chart by filtering data through time.

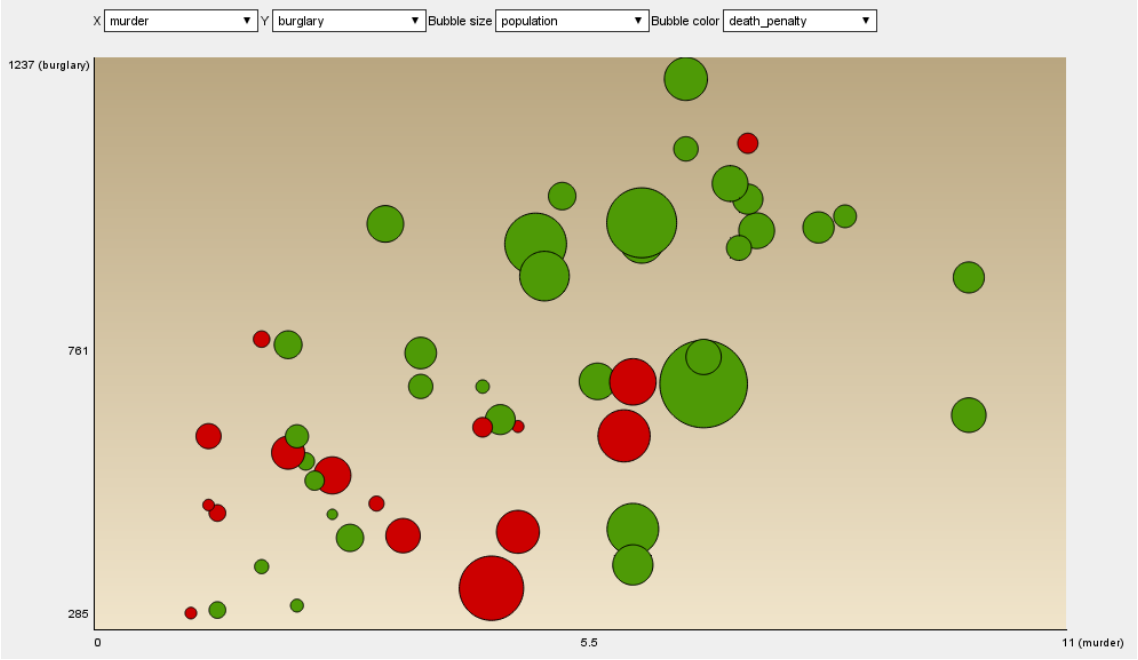

Figure 4. Bubble chart running

5.2. Parallel Bars Chart

Parallel coordinates chart can represent, in a bi-dimensional space, *n* distinct dimensions of dependent attributes. Each attribute is represented by a bar, which are equally spaced and positioned parallel [Inselberg and Dimsdale 1990]. Parallel coordinates chart represents each element from the dataset as a line that crosses each attribute bar. The problem is that this visualization can lead to poor performance when many elements are present.

A variation of this technique, called *parallel bars* (Figure 5), is proposed in this work. Instead of representing each element as a line, each subset of elements is represented by a polygon, which height is proportional to the value at that bar, so the subset is filtered as the polygon crosses the bars. This way less lines are represented without information loss. Global analysis is possible since each line groups elements with the same characteristics.

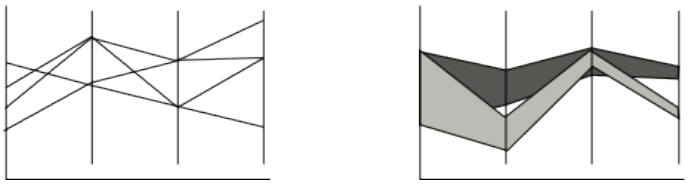

Figure 5. Main difference between the way that data is represented in parallel coordinates (left) and in this new proposal (right)

The main difference between this visualizer and the three others is that this one requires the definition of not only the variables that will be plotted in each dimension, but also which are these dimensions. Each dimension is represented by a bar that is related to some attribute from the dataset. The default interface allows bars to be added, removed and reordered. It also allows new polygons to be added, removed or manipulated, by just

clicking on the value for some attribute. Another great difference is that this one usually will demand new data requests. A local cache is available to avoid unnecessary requests.

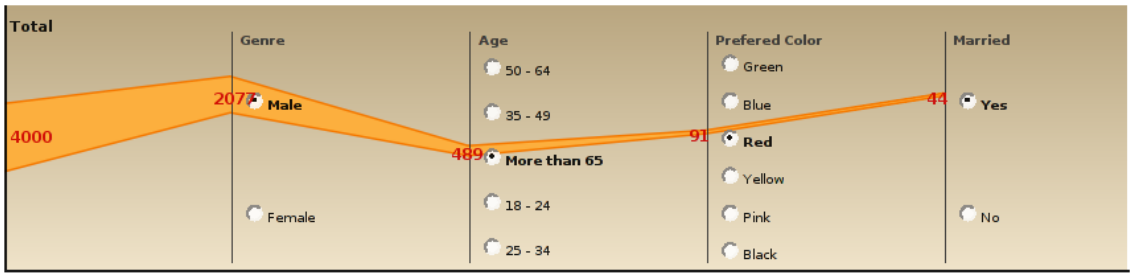

Figure 6. Parallel bars running

This way a big dataset can be filtered as the bars are being crossed. For example, from Figure 6, using fake data, it's possible to see that the whole dataset has 4000 elements. From this total, 2077 are men, of which 489 have more than 65 years. From those, 91 has red as the prefered color, and yet 41 from those are married.

But the example on Figure 6 has only one polygonal line. The example below on Figure 7 represents data about crimes against the human rights on the Internet. It's also possible to see the control box that is available for all visualizations of this type. Through this control box, it's possible to interact with the visualization, by choosing what dimensions will be active and how they will be ordered. A timeline is also present, and so the data to be visualized can also be filtered by time.

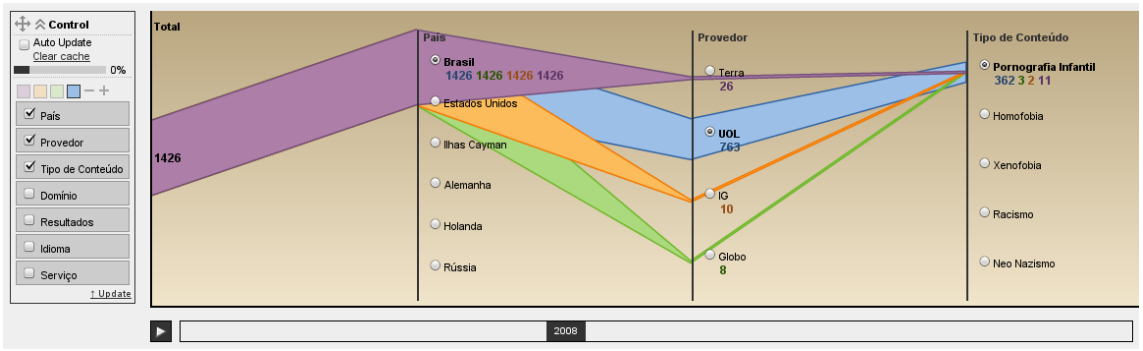

Figure 7. Parallel bars with four lines, seven dimensions (from which three are active), and a timeline

Although this kind of visualization supports infinite active dimensions theorically, in practice this is not feasible (nor useful) since many active dimensions can lead to poor performance and confusing analysis, which is the opposite to the goal of this visualization.

5.3. Choropleth Map

Choropleth map is a conventional way to plot geographic density data. This technique uses a sequence of color tones or shadows to apply over the regions. Areas with higher values get darker colors, while areas with lower values get lighter colors [Cuff and Mattson 1982]. In this OPENEDEYES implementation, the map regions are

colored according to value of the represented numeric variable. The colored regions define a dimension of this visualization. The other dimension available is defined by bubbles over regions, which areas are proportional to the represented variable. This alternative is present because in some cases colored maps are not the best way to visualize data, since bigger regions get more attention than the smaller ones, and that can lead to misinterpretation [IBM 2007].

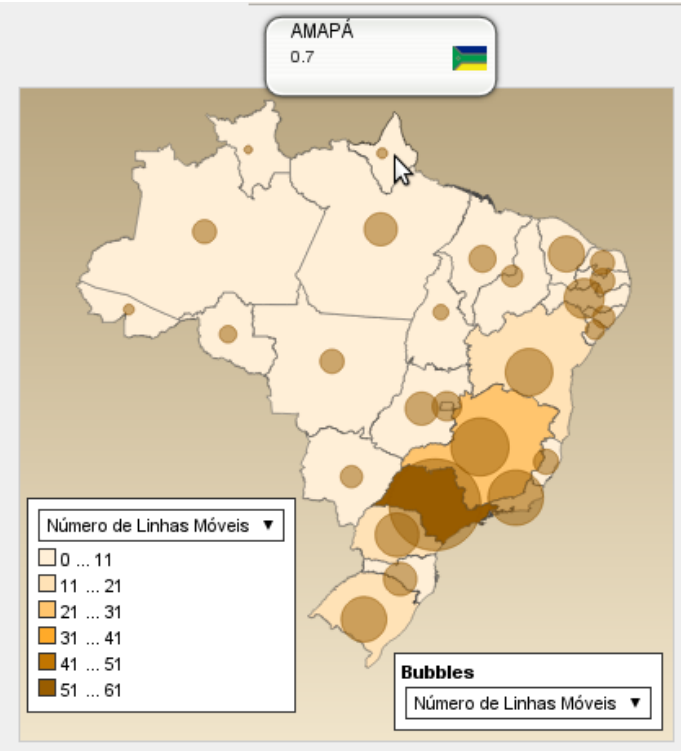

Figure 8. Choropleth map of Brazil about mobile phone subscribers

The interaction with this visualization can be done by selecting from the legends which variable will be represented in each dimension (region color and bubble). Any of those dimensions can be disabled.

This visualization uses map templates, which must be SVG files in which each region is an element identified by an unique id. This same id must be referenced in the dataset, and then this way data can be mapped. The example on Figure 8 uses the map of Brazil. The world map is also available by default, and other maps can be added as well, since they follow the requirements mentioned above.

Many times, analyzing data from a region level is not enough, and so deeper analysis is necessary, from a geographic coordinates level, for example. For this case, the module to be presented on the next section was developed.

5.4. Great Circles Map

A great circle is a section of a sphere that contains the diameter of the sphere. The shortest path between two points in a sphere is a segment of a great circle [Weisstein 2011]. As the Earth is not a perfect sphere, the shortest distance between two points is not exactly a great circle, but in this work an approximation is made and so the Earth is considered

a perfect sphere, as the goal is not to measure distances, but notice relationships between points.

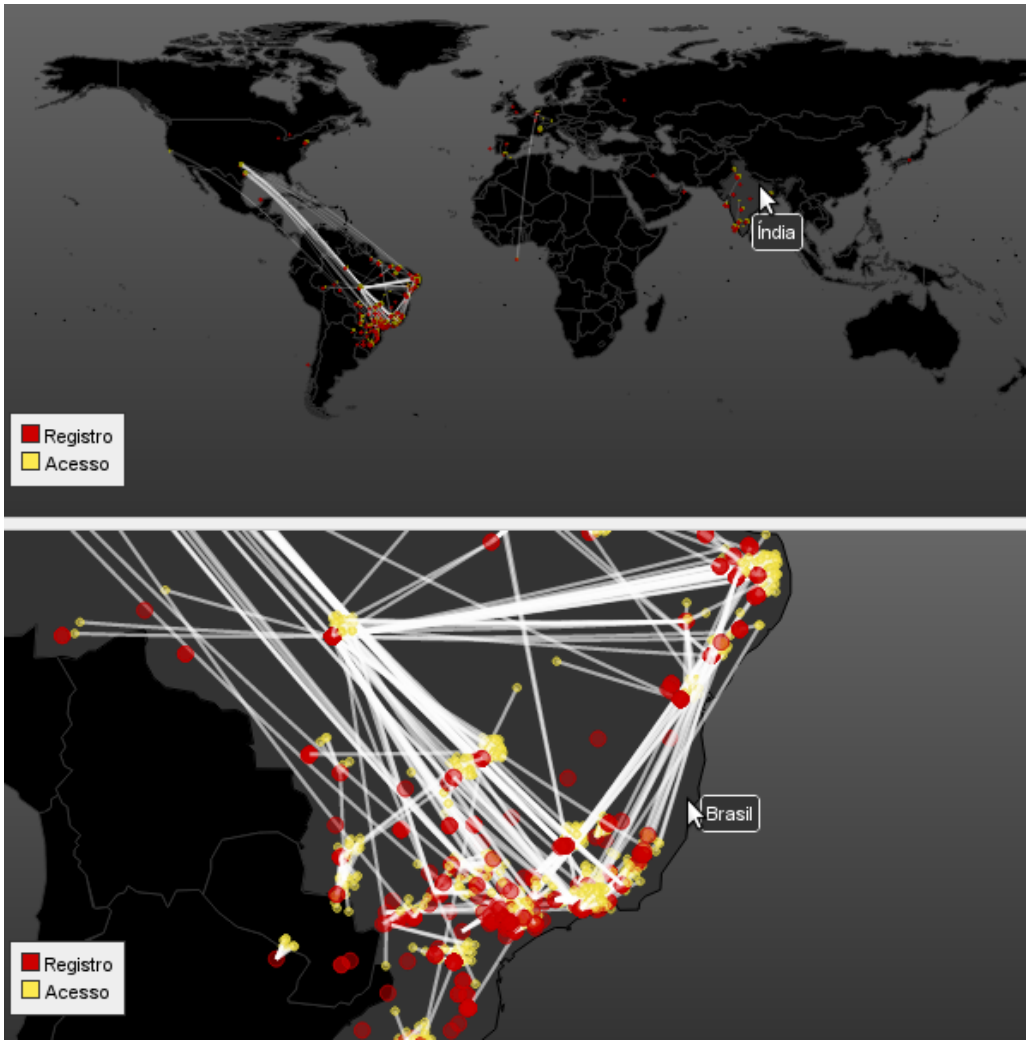

Figure 9. Great circles running

The goal of this visualization is to represent data associated with geographic coordinates and also to represent connections between those points, using great circles. Each element from the dataset is a point marked on the map. The color of the point is also a dimension and so may be associated to a variable from the dataset. The connections between points, using great circles, is also a dimension, so in order to use this dimension, there must be a variable in the dataset whose value is a reference to another element from the same dataset.

Interaction is achieved by passing the mouse over a point, which shows the attribute values for that element, and by zooming and dragging the map. At last, a timeline is also available, so data can be visualized through time in an animated way.

The empiric results obtained in this section were able to prove that the framework is flexible and usable, being capable of building information visualizations with different relationships, as well as implementing various visualization techniques.

6. Conclusion

This work presented OPENEDEYES, a JavaScript, web standards-based free framework for information visualization on the web. Its architecture is modular, defined by five core modules, responsible for common actions and basic structure, and four visualization modules, which implement advanced visualization techniques to fit distinct datasets. As a framework, its hybrid behaviour, acting sometimes as a black box framework or as a white box framework, makes it usable for developers and for users, from novice to advanced. Interaction is such an important ingredient in visualization, and so all developed visualization modules include a default interaction interface but also allow new controls to be added. OPENEDEYES comes to fill a gap in the visualization tools world by offering tools which are, at the same time, web based, interactive, simple but robust, and even innovative. Most important, it's freely licensed and provides a structure that allows new visualizers to be added. Future works include studying how this implementation scales for larger datasets, whether there are performance failures, how can they be mitigated, how those visualization tools can be used to analyze data about free software projects, and evaluate the usability of the visualizers for users and developers, since for the time being only the authors used it.

References

Adobe (2011). Flash. http://www.adobe.com/br/products/flash.html.

Baranovskiy, D. (2011). Raphael. http://raphaeljs.com/.

Card, S., Mackinlay, J., and Shneiderman, B. (1999). *Readings in information visualization: using vision to think*. The Morgan Kaufmann series in interactive technologies. Morgan Kaufmann Publishers.

Card, S. K., Robertson, G. G., and Mackinlay, J. D. (1991). The information visualizer, an information workspace. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*, CHI '91, pages 181–186, New York, NY, USA. ACM. http://doi.acm.org/10.1145/108844.108874.

Chi, E. H. (2000). A taxonomy of visualization techniques using the data state reference model.

Crockford, D. (2006a). JSON: The fat-free alternative to XML. http://www.json.org/fatfree.html.

Crockford, D. (2006b). RFC 4627 - the application/json media type for javascript object notation (JSON). https://tools.ietf.org/html/rfc4627.

Cuff, D. and Mattson, M. (1982). *Thematic maps: their design and production*. University paperbacks. Methuen.

Fry, B. and Reas, C. (2011). Processing. http://processing.org/.

FusionCharts (2011). FusionCharts. http://www.fusioncharts.com/.

Garret, J. J. (2005). AJAX: A new approach to web applications. http://www.adaptivepath.com/publications/essays/archives/000385.php.

Haber, R. B. and Mcnabb, D. A. (1990). Visualization idioms: a conceptual model for scientific visualization systems. *Visualization in Scientific Computing*.

Heer, J., Card, S. K., and Landay, J. A. (2005). Prefuse: a toolkit for interactive information visualization. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '05, pages 421–430, New York, NY, USA. ACM. http://doi.acm.org/10.1145/1054972.1055031.

IBM (2007). Many Eyes: Maps. http://www-958.ibm.com/software/data/cognos/manyeyes/page/Maps.html.

Inselberg, A. and Dimsdale, B. (1990). Parallel coordinates: a tool for visualizing multi-dimensional geometry. In *Proceedings of the 1st conference on Visualization '90*, VIS '90, pages 361–378, Los Alamitos, CA, USA. IEEE Computer Society Press.

Ippolito, B. (2005). Remote JSON - JSONP. http://bob.pythonmac.org/archives/2005/12/05/remote-json-jsonp/.

Keim, D. A. (2002). Information visualization and visual data mining. *IEEE transactions on visualization and computer graphics*, 7(1).

Kosara, R. (2011). The state of information visualization, 2011. http://eagereyes.org/blog/2011/state-of-infovis-2011.

Markiewicz, M. E. and de Lucena, C. J. P. (2001). Object oriented framework development. *Crossroads*, 7:3–9. http://doi.acm.org/10.1145/372765.372771.

Mazza, R. (2009). *Introduction to information visualization*. Springer.

Nielsen, J. (2000). Flash: 99% bad. http://www.useit.com/alertbox/20001029.html.

Nurseitov, N., Paulson, M., Reynolds, R., and Izurieta, C. (2009). Comparison of JSON and XML data interchange formats: A case study. *Scenario*, 59715:157–162. http://www.cs.montana.edu/izurieta/pubs/caine2009.pdf.

Project, T. W. S. (1998). About the web standards project. http://www.webstandards.org/about/.

Resig, J. (2011). ProcessingJS. http://processingjs.org/.

Tableau (2012). Tableau. http://www.tableausoftware.com.

Thomas, J. J. and Cook, K. A. (2005). *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. National Visualization and Analytics Ctr. http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0769523234.

TIBCO (2012). Spotfire. http://spotfire.tibco.com/.

Viegas, F. B., Wattenberg, M., van Ham, F., Kriss, J., and McKeon, M. (2007). Many Eyes: a Site for Visualization at Internet Scale. *IEEE Transactions on Visualization and Computer Graphics*, 13:1121–1128.

W3C (2011a). Cascading Style Sheets. http://www.w3.org/Style/CSS/.

W3C (2011b). Extensible Markup Language. http://www.w3.org/standards/xml/.

W3C (2011c). HTML 5. http://www.w3.org/TR/html5/.

W3C (2011d). Scalable Vector Graphics. http://www.w3.org/Graphics/SVG/.

Weisstein, E. (2011). Great circle. http://mathworld.wolfram.com/GreatCircle.html.