

Interactive Visualisation of Large Multivariate Datasets on the World-Wide Web

¹JL Wesson & ²PR Warren

¹Department of Computer Science & Information Systems
University of Port Elizabeth, SOUTH AFRICA

²Information Technology Programmes
University of Natal (Pietermaritzburg), SOUTH AFRICA

csajlw@upe.ac.za

Abstract

Information visualisation of large multivariate datasets presents certain unique challenges. Few tools currently exist to facilitate the interactive display of such information on the World-Wide Web. This paper discusses the representation of multivariate data and the development of an API and an example application to view large datasets interactively in 3D in a web-based environment. This application enables the user to dynamically create and view XYZ scatter charts of student placement information using a web browser.

Keywords: H.5.1, H.5.2, H.5.4, I.3.7, I.4.9.

1 Introduction

The World-Wide Web (WWW) contains vast quantities of information that needs to be visualised and yet there are few tools available to a web developer to do this. At most there is a static display of data where the supplier decides on the visualisation and thus essentially the interpretation. The real challenge is the interactive display of the data by the users allowing them to view and analyse the data to suit their specific requirements. This will empower them in ways in which static information, be it in print or on the web, is never able to do. Fortunately, new opportunities to achieve this are presenting themselves. There is a better understanding of what the field of information visualisation entails (Card, Mackinlay and Shneiderman, 1999), (Munzner, 2000). There is also the emergence of Java-based technologies, which hold the promise that application programmers can use well-defined components to create systems with additional functionality. And finally there is the emergence of Web3D tools, such as the virtual reality modelling language (Carey and Bell, 1997), (Roehl, 1998), which would allow developers to display the results on the screen with relative ease.

The goal of this paper is to investigate the extent to which these technologies can be used to facilitate interactive information visualisation of large datasets on the WWW. This hypothesis will be evaluated by the development of

an API and an example application to view large data sets in an XYZ scatter chart on the WWW.

2 Information visualisation

Information visualisation (IV) is defined as the use of computer-supported interactive visual representation of abstract data to amplify cognition (Card, Mackinlay and Shneiderman, 1999). This implies that IV can be seen as a means for exploring data in a way that enhances our ability to understand relationships and identify patterns. Particularly challenging is the situation where large amounts of data are available. Tufte amongst others gives many examples of innovative and clever ways to visualise information but many of these are specific rather than generic (Tufte, 1990). Munzner maintains that the two key advantages of using computer-based systems for information visualisation are interactivity and scalability (Munzner, 2000). Munzner also argues that designing visualisation software by taking the characteristics of a target user's task domain into account, leads to more effective systems that can scale to larger datasets.

Many datasets can be visualised using a number of standard charting methods, and this makes it worthwhile to develop such tools. These datasets include classical statistical data where the data can be represented as a table (relation) consisting of rows and columns (attributes). The rows could represent different individuals and the attributes, different variables associated with the individuals.

Several commercial tools exist for visualising large data sets. These include Spotfire (Spotfire, 2000), Crystal Reports (Crystal), and even Microsoft Excel (Microsoft, 1999). Most of these tools, however, are very large and expensive and little is known about the architecture and design of these tools. Our goal is therefore to investigate how standard technologies, namely Java, VRML and SQL can be used to build an API and a web-based application to view large datasets interactively on the WWW.

Firstly, we need to understand the semantics of the attributes and secondly, the ways these can be arranged into charts. This discussion is based upon the work of (Card, Mackinlay and Shneiderman, 1999).

2.1 Types of attributes

Attributes can be characterized into three basic types:

Copyright ' 2001, Australian Computer Society, Inc. This paper appeared at the Australian Symposium on Information Visualisation, Sydney, December 2001. Conferences in Research and Practice in Information Technology, Vol. 9. Peter Eades and Tim Pattison, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

- **N:** Nominal data, which is only equal to or not equal to other data, i.e. an unordered set. Student number or faculty name are examples that come to mind in an educational context. Both of these can be sorted but there is no inherent meaning in the order. Another name for nominal data is symbolic data.
- **O:** Ordinal data is drawn from an ordered set but where there is no sense of magnitude. An example is the outside temperature, which might be classified as cold, warm or hot.
- **Q:** Quantitative data indicates a notion of scale in such a way that we can do arithmetic and for which we can attach definite meaning to the magnitudes and differences. Time is an example of this type of data. Quantitative data need not be continuous but often is.

Data can always be downgraded from Quantitative to Ordinal or from Ordinal to Nominal by simply ignoring the magnitude or ordering information. The intended meaning is very much in the mind of the person creating the visualisation. While downgrading is always possible

upgrading is not, as the information to do this is not available. There are also transformations that can be applied to the data for instance by binning (e.g. placing ages into age groups), by folding (e.g. transforming quantitative dates into the nominal days of the week) or by aggregating. These transformations can change the characterisation. If one attribute is transformed, additional transformations may be applicable to other attributes in the table.

One might think that only two dimensions can be represented on a two-dimensional (2D) surface but this is not the case. Even in the case of simple 2D scatter plots, the use of symbols to represent additional information has been common for a long time and 2D has, since the inception of art, been used to represent three dimensions (3D). Remembering that we can always downgrade $Q \rightarrow O \rightarrow N$, Table 1 summarizes how the various types can be represented. The person creating the visualisation, however, needs to take care because using a dimension suitable for quantitative data for a nominal type, for instance, might well imply a quantitative interpretation in the mind of the viewer (Card, Mackinlay and Shneiderman, 1999).

Dimension	Type	Range	Remarks
Space (Position)	Q	Many	Axes are traditionally used to represent this kind of data.
Time	Q	Moderate	Animations would best be used to represent a time aspect especially for the evolution of systems.
Colour	N	< 10	Many more colours can be distinguished if adjacent but it is unlikely that as many can be recalled. There are said to be seven colours in the rainbow.
Hue	N	< 10	Position on the colour wheel.
Saturation	O	< 10	How much white has been added to the colour.
Value	O	< 10	The intensity of the colour.
Transparency	O	< 10	
Shape	N	< 10	
Size	Q	< 20	
Orientation	Q	< 20	
Texture	N	< 10	

Table 1 Representation of different data types

Note that the dimensions that are available for quantitative data (e.g. size or orientation) will in fact not always be suitable as is not possible to read values off the graph and the range supported may be insufficient.

The above describes how individual data elements can be represented but these need to be combined in order to understand the relationships between two or more variables. Our objective is to be able to represent large data sets with many dimensions, i.e. multivariate data.

2.2 Choice of charts

There are basically two different chart types, namely linear and hierarchical. Typical linear charts include the normal family of bar charts, various methods of plotting XYZ data such as scatter diagrams, and pie charts. In the traditional pie chart, the area represents the magnitude of the attribute, while in a variation called a star chart, distance is used to represent this. A further variation produces a sequence of star charts to visualise an additional dimension. Hierarchical charts, on the other hand, stress the hierarchical relationships in the data. Trees are the most common form but any other

hierarchical structure can also be used. Typical examples are Guide Balls and InfoCubes (Darville and Van Espen, 2000). Such techniques are particularly suited to allowing users to drill down into the information. There is no need to have two different formats of data, however, as a simple tree walker can convert hierarchically organised data into a table. After sorting the table the converse is simple enough as well.

We can further classify chart types as 2D or 3D. Despite Tufte's concern about the gratuitous use of 3D when no additional information is being conveyed, 3D does allow for a number of different effects to be represented (Tufte, 1990). It allows for an extra dimension to be visualised with relative ease and enables the user to look at the information from various perspectives. In addition some chart types such as the InfoCube are inherently 3D. People live in a 3D world and a 2D representation is to some extent artificial. The tools and techniques for displaying 3D information are now available in the VRML and the Java3D packages and thus we will restrict ourselves to the display of 3D information.

3 Implementation strategies and tools

The first choice is to decide between developing a package which given a database and some parameters will allow the user to visualise the information, and between creating a set of APIs (application programming interfaces) for an application programmer to use. The package approach has several shortcomings. There are too many database formats and it is very difficult to predict the kind of transformations that any given visualisation would require. At this stage it seems better to create an API for programmers to use that is independent of any database access and transformation issues. Any higher-level tools can be built later, if necessary, on top of such APIs.

The next problem is to decide on suitable programming tools for the implementation. The tools must be able to create a virtual world to support the desired level of interaction. The minimum interactivity, given the 3D charts, is to be able to view this world from different perspectives. Regenerating the charts from first principles can create further interactivity if required. Since web applications are involved, the choice is between using Java3D or VRML.

Java3D is an API for writing 3D applications. It provides a rich set of features for creating 3D worlds and can deliver reasonably high performance. It also is well standardised and the standard is likely to remain stable, as it is part of the Java suite. Unfortunately it requires too many lines of code even to set up a simple shapes and much of the detailed control is not required. Java may well be the right way to go in the long term, but while concepts behind the tools are being explored a simpler approach is needed. Conversely the Virtual Reality Modelling Language (VRML) provides a lightweight approach to constructing 3D worlds but it encodes only the content and a VRML browser is needed to display the 3D scene. The scene can be manipulated by the Java2D

package, together with the so-called External Authoring Interface (EAI). The EAI is currently only a proposed standard.

The drawback of this approach is that a third party browser plug-in is required and these may not (and indeed have not) kept up with the rapid evolution of Internet browsers. It is clear that in the long term a standardised solution is required. If a VRML browser does not get built into Internet browsers then visualisation tools will have to create their own classes to view the VRML scene, or alternatively the scene will have to be created in Java3D. In the mean time we can demonstrate the principles with VRML and the Cosmo Player plug-in.

3.1 JavaBeans or Classes

Java has two ways of providing robust and reusable component architectures, namely traditional classes and the emerging JavaBeans (Hamilton, 1997) approach. The idea of JavaBeans is to provide a software component model so that components can be composed together into applications and manipulated visually in a builder tool. This is not required for our primary objective. Our aim is to provide APIs for application programmers to use as part of creating web-based visualisations. The use of JavaBeans is possible but for this kind of application is it not indicated. For instance, Hamilton does not consider JavaBeans suitable for use together with JDBC. It may well be desirable in future to provide end user tools and then this assessment may change.

4 Architecture

This section describes the architecture of the API we built to validate the ideas that we can provide interactive 3D visualisation tools for the web. The basic architecture is shown in Figure 1.

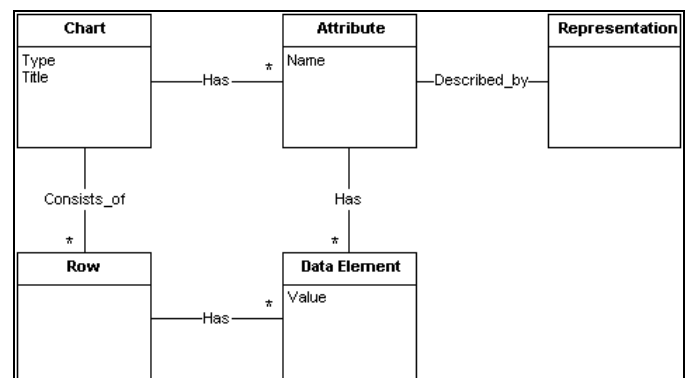


Figure 1. Basic architecture of the API (UML notation)

A chart simply has a title and a number of Rows. Each Row has a number of Data Elements. We chose to pass one row of data over to the chart at a time in analogy with writing one record to a file. The Java vector class makes it easy to pass an array of heterogeneous objects over to a class and this approach is easy to implement. This essentially describes the data; the semantics are described by the Attribute class. This describes the type of data involved in a static and representation-independent way. The way a particular attribute is to be rendered on the screen is described by its Representation.

The Representation constitutes the most difficult part of the design and how this is to be done depends strongly on the kind of chart to be displayed. In the case of an XYZ scatter chart, three of the attributes need to be selected to map onto the axes. After that one attribute needs to be assigned to a shape to plot at the corresponding XYZ position. Finally, the other attributes can be mapped onto any of the other dimensions (see Table 1). The only general way to map a value onto a representation is for the application programmer to specify a function to do this. The prototype takes a more modest approach and a simple table maps a set of discrete values onto their representations. Furthermore, the shape parameter is restricted to those shapes directly available in VRML. It is easy enough to extend this approach. Each data element in a column needs to be of a given class. A method of that class can simply be its representation and inheritance can give the programmer full control over the way in which this is done.

Certain charts are more suitable for certain purposes than other charts. It is conceivable that the tool could assist the programmer in this choice. However at this stage we are simply interested in providing the basic functionality. Any future enhancements will have to be built on top of this primary layer.

The chart interface specifies nothing about the orientation and size of the eventual picture as it is rendered on the screen. The system only specifies the VRML world and not the rendering of that world. This is done by the VRML browser plug in which can be manipulated independently by the end user. In our case the Cosmo browser was used (Cosmo, 2001).

5 StudentView: A 3D tool to visualise student data

We used this API to construct an application to visualise information for students entering the University of Port Elizabeth (UPE). A SQL database was constructed containing the results of several different placement tests for prospective students. The placement test results consisted of the students' marks for Reading Comprehension, Arithmetic, Algebra and Sentence Skills. The data captured included both quantitative and nominal data. The quantitative data consisted of the results for the four different placement tests and the students' average matriculation mark. The nominal data consisted of biographical and registration data such as the students'

gender, race, faculty, home language, age and study programme. The database contained approximately 5000 records for each of the past two academic years.

Prior to the development of this prototype, the Student Placement Unit at UPE had no facility to view overall student placement results. The only tools available were simple 2D graphs that were used to plot an individual student's results and compare these with the acceptable level. The users requested a tool that would enable them to dynamically compare the results for different groups of students, e.g. according to gender, race, faculty, risk category, home language, age and study programme. This facility also needed to be web-enabled so that the different stakeholders at UPE could view the results from different locations.

5.1 User Interface Design

The user interface was designed to provide web-based access to the SQL database to members of the Student Placement Unit. This interface enables the users to dynamically select both what data they want to visualise, as well as how they wish to visualise this data. The user is allowed to select either to view all the student data for a given year or to select a subset based on some selection criteria. For example, the user can select to view only the results for a given age category, risk category, gender, study programme, race, home language or faculty (see Figure 2).

Select	Attribute	Range	Include
<input type="checkbox"/>	Age	16-18	<input type="checkbox"/>
<input type="checkbox"/>		19-21	<input type="checkbox"/>
<input type="checkbox"/>		>21	<input type="checkbox"/>
<input type="checkbox"/>	Risk category	Low	<input type="checkbox"/>
		Medium	<input type="checkbox"/>
		High	<input type="checkbox"/>
<input checked="" type="checkbox"/>	Gender	Male	<input checked="" type="checkbox"/>
		Female	<input checked="" type="checkbox"/>
<input type="checkbox"/>	Study programme	UPEAP	<input type="checkbox"/>
		Non-UPEAP	<input type="checkbox"/>
<input checked="" type="checkbox"/>	Race		<input checked="" type="checkbox"/>
			<input type="checkbox"/>
<input type="checkbox"/>	Home Language		<input type="checkbox"/>
			<input type="checkbox"/>
<input checked="" type="checkbox"/>	Faculty	Low	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Figure 2 Screen to select students in StudentView

The user can then choose which attributes are to be displayed and how these are to be displayed. These attributes include quantitative results for the different placement tests, as well as other nominal data such as race, gender, faculty and degree. Because the final representation is using an XYZ scatter chart, users must select at least 3 quantitative attributes that will be represented using the x-axis, y-axis, and z-axis respectively. A fourth quantitative attribute can be

represented using the size dimension. At most 3 nominal attributes can be selected; these will be represented using other dimensions, such as shape, colour and brightness (see Figures 3 and 4).

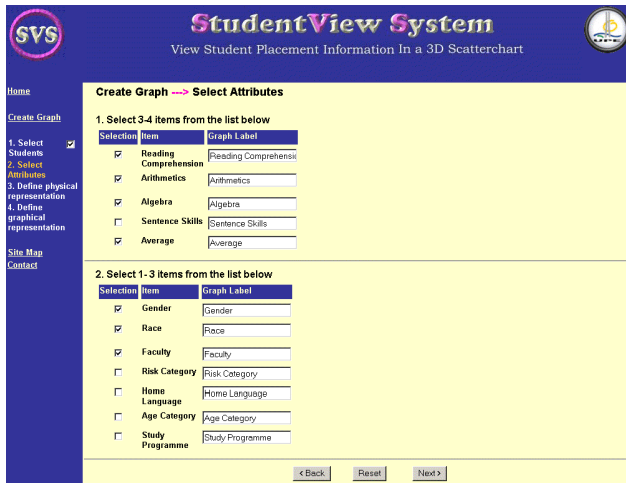


Figure 3 Screen to select attributes in StudentView

Finally, the user can then choose the desired graphical representation of each of the nominal attributes selected previously. For example, if the user has selected to represent race using colour, then he/she can select the desired colours for each of the possible attribute values, i.e. African, Coloured, Indian, White and Other.

Likewise, different shapes can be used to represent the different faculties (see Figure 5).

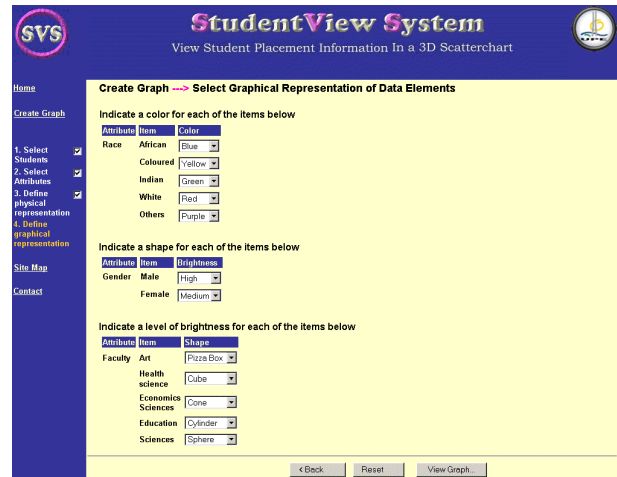


Figure 5 Screen to select graphical representation of attributes in StudentView

5.2 Display

Once the user has created the graph, the final results are then displayed on the web browser in the form of an XYZ scatter chart (see Figure 6).

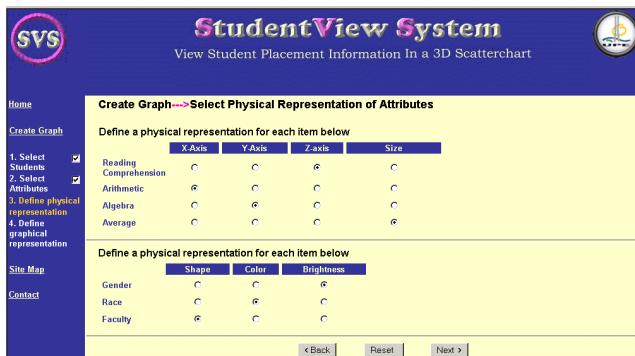


Figure 4 Screen to select physical representation of attributes in StudentView

This example chart depicts the results for 512 students, where shape has been used to represent faculty; colour to represent race; brightness to represent gender; and size to represent the average matriculation marks. The results for the three selected placement tests (Arithmetic, Algebra and Reading Comprehension) are indicated on the X, Y and Z-axes respectively. Note also how the Cosmo Player controls are visible at the bottom of the page. The user can then use the Cosmo Player controls to interactively view the chart from different perspectives or viewpoints (i.e. front, back, right, left, up, down), or using the Cosmo Player navigation controls.

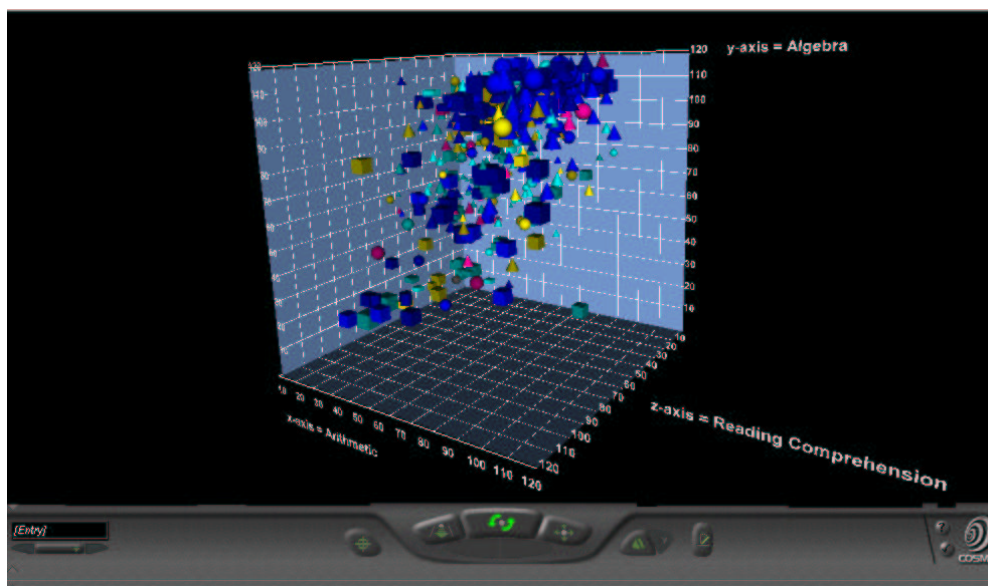


Figure 6 XYZ Scatter Chart produced in StudentView

6 Conclusions and Future Research

IV is a powerful technique that can deliver tremendous results that has been insufficiently exploited on the web. Multivariate data can be represented graphically in 3D using a combination of different techniques to represent the various kinds of information. This representation is based on the underlying types of the data elements as well as the visualisation requirements of the user. We have demonstrated that visualisation tools can be built using Java and VRML technology to facilitate this process and enable the interactive visualisation of this data on the WWW.

The API discussed in this paper was used to construct a prototype tool during 2000 (El ansari and Vanbrabant, 2001). This initial prototype was redesigned during 2001 to develop a system called StudentView to view student placement information on the WWW. StudentView has only been evaluated using informal user testing and a more thorough usability evaluation must still be conducted. This tool, whilst still primitive, offers the promise of providing a novel means for users to view results dynamically and to explore the relationships that exist in the data across the WWW. There are, however, some disadvantages to using a VRML browser to view the chart. These include the problem of legibility of the axis labels as the chart is rotated (see Figure 6).

There is a need to extend this work to include more kinds of charts. Despite many differences in the appearances, there are many similarities in the graph types that would encourage the application of object-oriented techniques. There is also the need to be able to support the programmer in the choice of chart types to display the information. Finally the classes need to be packaged in such a way that the more convenient scripting languages, in particular VBScript and JavaScript, can be used to glue these components together.

7 Acknowledgements

Acknowledgements must go to Abdesamad El ansari and Alain Vanbrabant who implemented the initial API and prototype during their internship at UPE (El ansari and Vanbrabant, 2001). Acknowledgement is also due to Pascal Foamoudjo-Mojuye, a BSc Honours student at UPE, who redesigned the API and the user interface during 2001 in order to improve the usability and efficiency of the system.

8 References

- CARD, S.K., MACKINLAY, J.D. and SHNEIDERMAN, B. (eds) (1999): *Readings in Information Visualization: Using Vision to Think*. San Francisco, California, Morgan Kaufmann Publishers, Inc.
- CAREY, R. and BELL, G. (1997): *The Annotated VRML 2.0 Reference Manual*. Addison-Wesley.
- COSMO (2001): Cosmo Player, SGI. <http://www.sgi.com/software/cosmo/player.html>.
- CRYSTAL Crystal Reports, Crystal Decisions. <http://www.hallogram.com/crstlrep/>.
- DARVILLE, C. and VAN ESPEN, S. (2000): Study of the Virtual Reality Technology used in the Visualization of Business Information. Masters thesis. Institut d'informatique, FUNDP. Namur, Belgium
- EL ANSARI, A. and VANBRABANT, A. (2001): Interactive Visualisation of Large Data Sets with the Java and VRML Technology. Masters thesis. Institut d'informatique, FUNDP. Namur, Belgium
- HAMILTON, G.E. (1997): JavaBeans API Specification. <http://java.sun.com/beans>.

Microsoft Excel 2000. Microsoft Corporation.

MUNZNER, T. (2000): Interactive Visualization of Large Graphs and Networks. PhD thesis. Stanford University. http://graphics.stanford.edu/papers/munzner_thesis/.

ROEHL, B. (ed) (1998): *Late Night VRML 2.0 with Java*, Ziff-Davis Press.

SPOTFIRE (2000): Spotfire DecisionSite: Making High Quality Decisions at e-Business Speed, Spotfire Inc. <http://www.spotfire.com>.

TUFTE, E.R. (1990): *Envisioning Information*. Cheshire, CT, Graphics Press.