A Framework for Interactive Web-Based Visualization

Nathan Holmberg†, Burkhard Wünsche‡, Ewan Tempero‡

Department of Computer Science
University of Auckland,
Private Bag 92019, Auckland, New Zealand
Email: nhol021@ec.auckland.ac.nz†,{burkhard,e.tempero}@cs.auckland.ac.nz‡

Abstract

As the power of end user web browsers increases, the delivery of sophisticated visualizations of information via the web becomes possible. However no technology exists that offers the kind of interactions that a stand-alone application can deliver. Technologies such as Java3D, VRML, X3D and SVG incorporate powerful rendering capabilities but make it difficult to interact with the underlying source data. Other technologies such as SMIL can also offer synchronization but then lack this rendering context. We present a framework within which these technologies can be evaluated.

We also address the question of how to integrate these technologies with existing and well-understood web technologies such as Javascript, CSS, Web Services, and PHP, to provide interactive web-based visualization applications. We then describe a generalized framework for determining how to choose the right set of technologies for a web-based visualization application.

Keywords: Visualization, Web3D, web-application, interactive

1 Introduction

A *web application* uses a standard web browser as its user interface and obtains its functionality (potentially) from a remote host. As the power and ubiquity of the Internet continues to grow, the use of web applications more advanced than simple conveyance of static textual information is inevitable. One of the most obvious applications is one that harks back to the original intention of the Internet, that of sharing scientific information.

We are particularly interested in providing *visualizations* of scientific and other information. A number of technologies have been developed that provide limited support for sophisticated visualizations, such as SMIL, SVG, Java3D, VRML, and X3D. These technologies can be used to create traditional visualizations and mapping of data onto graphical icons, but are, by themselves, unable to provide the kinds of interactions and user interfaces common in stand-alone applications. The question we address in this paper is, can these technologies be effectively used for *interactive visualizations* in a web environment.

There are a number of issues that differentiate web-based interactive visualization from conventional applications. One is the nature of the visualizations themselves. Some visualizations can only be expressed using modern computer graphics techniques that may not be possible in the web technologies that are available (e.g., 2 dimensions vs. 3 dimensions). Another issue is interaction. The standard web browser and the standards it supports provide a very limited form of control and a very simple form of interaction. A number of technologies have been introduced to provide client-side interaction, such as Javascript, Java Applets, and various plug-in support, but whether these can be effectively married with the visualization technologies needs to be addressed. A third issue is the networked nature of web applications. This raises questions about performance, both throughput and latency, that may affect the effectiveness of the application. Finally, there is the issue of the maturity and accessibility of the technologies. Most require some form of third-party plug-in to the web browser. Whether there is a mature-enough product for a user's preferred browser will also impact an application's effectiveness.

In this paper, we address the issues raised above and describe a generalized framework for determining how to choose the right set of technologies for a web-based interactive visualization. We begin by providing an overview of visualization, with a view to determining the requirements of applications. We then present an evaluation model that captures the issues discussed above and allows the different choices to be evaluated. In section 4, we discuss a number of available technologies in the context of the evaluation model. Section 5 presents our proposed visualization framework, and then section 6 shows an application we have developed using the framework. We then evaluate our results in section 7, and present our conclusions.

2 What is Visualization

Tufte defines visualization as a *science* instead of simply the art of creating comprehensible pictures (Tufte 1983). He describes the necessity of not only creating graphical representations that best display the information in question but that also maintain 'graphical integrity'.

2.1 The Types of Visualization

There are three primary types of visualization; information, scientific and software. Information visualization is defined as 'the use of computer-supported, interactive, and visual representations of abstract data to amplify cognition' (Wakita & Matsumoto 2003). This can be used in almost any field to help users deal with the problems specific to their field, whether this be anything from chemistry to organizational development. More recently these techniques are moving from the lab into real applications used by the general populous (Plaisant 2004) and so the importance of the field is ever growing. Informational visualization appears to be the first to be suitable for publishing on the web, perhaps because of its wider potential audience.

The other two forms of visualization, scientific and software follow similar principles. Scientific visualization is described by (Aref, Charles & Elvins 1994) as 'when computer graphics is applied to scientific data for purposes

of gaining insight, testing hypothesis, and general education' while software visualization is used to understand complex software systems and their lifecycles.

The original intention of this work is to focus on informational visualization and the publication of such on the web however the principles can easily be applied to the other forms discussed here.

2.2 The Visualization Process

Figure 1, adapted from (Wuensche & Lobb 2001) shows the basic process of visualization including the creation and subsequent dissemination by humans. We can only control the visualization stage with the hope that the users interpretation will be as successful as possible and thus we concern ourselves with the method of mapping data to certain structures and then displaying these structures in the best way possible.

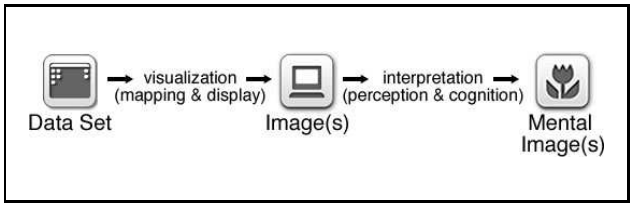

Figure 1: The visualization process

We now must also take into account issues resulting from using the web as a medium. For example, we are now dealing with both a client and a server and as such our discussion must take this into account. Later we use a specific system architecture for this but for the purposes of discussion this simple distinction is sufficient. A second issue is that whatever structure is chosen for the mapping must be sent to the user's computer and so both size and overhead have to be taken into account. An example is that if terrain data is represented as a series of polygons, it will have more overhead than representing the data just as a set of height values, however sending only height values is only an option of the technology supports its use (such as an 'elevationgrid' structure being used to display said data). While the choice of data representation can impact the effectiveness of the application's implementation, it is the choice of technology that is the main focus of this paper.

3 The Evaluation Framework

In an attempt to objectively compare the different technologies under review we have developed an evaluation framework. It consists of 15 different measures in 4 categories. While some of these measures are subjective they cover a wide variety of important aspects for the creation of web-based visualization and facilitate our discussion. A general approach is taken wherever possible although a section on the application specific features of each technology is included to ensure that these are accounted for.

In the following discussion an 'element' is a particular object within the overall visualization user interface. An example is a particular instance of a plug-in object that itself shows an X3D file. Thus inter-element communication deals with that between such stand alone objects and likewise intra-element events deal with how events are handled within the particular object or technology.

3.1 Technical Capabilities

The first section in this evaluation framework involves testing the technical capabilities of the various technologies and allows for an objective view of what the technology can do. We use 7 measures to describe this.

Network and Communications is the first measure and revolves around whether the technology natively supports communication with other servers, specifically for the downloading of new content or dynamic updating of the visualization. This feature makes it possible to split computational work between the client and the server.

The second measure is *2D/3D* and indicates whether the technology contains 2D or 3D graphical primitives to render natively. In this case the term 'natively rendered' refers to how viewers are intended to treat or store objects during the rendering process and what objects are made available. An example of this would be that mapping 3D volume data onto a 2D graphical icon is inappropriate.

Compression and Encryption asks if there is support for either compressed or encrypted communication, both with respect to the original content download and for any dynamic updates that may occur later in the visualization. As some scenes involve a lot of data, this is important as otherwise network latency becomes a limiting factor. While initially the concern for security was extended to enquiring if particular technologies supported the creation of user profiles it was felt that this was more aptly served by other web technologies as will be discussed later.

Native support for *animation* and/or tweening is also important as if these are included then processes or temporal changes can also be visualized. More important is the question of how easy it is to incorporate these animations

Finally in this section we attempt to measure how easy it is to create visualizations or models. This is really a subjective measure that indicates if programming and/or 3D modeling experience are needed to use the technology. Related to this is also the prevalence and quality of editing tools and the subjective measure is based on our experience using these tools.

3.2 Interactivity

The second section of our evaluation framework contains metrics for describing how visualizations can respond to user input. We have also included measures regarding the ability of a technology to integrate new content 'on the fly' and to communicate with markup languages to coordinate any other media that may be involved in the visualization.

The first measure is the *event model* and level of interactivity that is possible. This measures what can be received as input and what types of events can be used between parts of the scene contained within the visualization.

Script support, or what languages can be used for client side interaction, is also important. Related to this is also the script's security model and any reliance on third-party conformance. This is important as, for example, a script's ability to manipulate files on the local file system may dissuade many from using the technology as it represents a possible threat.

Dynamic integration of new content into the active scene is the third measure in this section. This involves the creation of models on the fly and the ability to insert them in whatever scene graph is used by the technology.

The last measure in this section is whether *communication with the containing markup language* or other elements is possible, such as DHTML or SMIL, and the level of control that this provides in both directions. This is important as it allows various elements on the page to be coordinated, allows supplementary information to be incorporated and increases the level of interaction possible. It is also important to note whether this capability is restricted to a particular browser such as Internet Explorer (IE) or whether it is cross compatible.

3.3 Community Support

This group of measures includes everything from how easy it is to find viewers for the technology and whether it requires special software installed to the likelihood that it will become or remain a dominant player in the online visualization field. The first measure is the *ubiquity and standardization of plugins or viewers*. This measure quantifies the availability and quality of these programs and indicates the maturity of the technology. Wherever possible mention is made of any special features that specific viewers have and the level of congruence between different plugins. As will be discussed later, this measure is extremely important when evaluating the suitability of a technology. The maturity and future prospects of the technology are also discussed.

The next measure considers whether the technology is *standards based*, whether this is from a respected organization such as ISO or if there is a particular group or company driving it. (Polys 2003) discusses the importance of this, especially in regards to the likely uptake of a particular visualization technique or technology if it adheres to a standard.

The *cost and availability of development tools* is also important and is related to the above measure. A discussion on the tools found and used during the evaluation process is included to ensure that an accurate representation of the process is made. Translation support and any tools that are available are also discussed as these can overcome the limitations of native development tools. (Plaisant 2004) explains that easy-to-use tools are extremely important as many users struggle already with simple business graphics. *Browser or operating system specificity* is the third measure as developing for the web should be open to as many possible end-users as possible.

Finally we review the *extensibility and ability to tailor* the technology to a specific application. This includes how flexible the rendering environment is and what degree of control we have over creating new objects and structures onto which the data can be mapped when creating a visualization. This dimension enables us to measure the power of the reviewed technologies and to balance the 'ease of creation' that often is opposed to this.

3.4 Application Specific

The final category is designed to capture any features of the technology that have been designed to make it appropriate for a specific type of visualization. There are only two measures in this section. The first is whether there are *native structures* for specific visualization icons or graphical entities such as height fields or volume visualizations. The second is if there have been *examples of the technology's use* in published visualizations. This measure was motivated by the fact that if the technology choice has been made by others already then there is likely to be a rational basis for using that technology. Also, existing applications often form a suitable starting point for developing new applications.

4 The Technologies

This section evaluates and categorizes web-based visualization technologies using the framework introduced above, the conclusions of which are summarized in Table 1. We found that the primary categories are 2D and 3D visualizations and we will use this distinction in this section. Later we also discuss different technologies for the presentation and integration of other media types and to the overall interface and interactivity.

The 3D technologies under review included the standard VRML97, its newer XML based successor X3D and the Java3D API. If a two-dimensional visualization is more appropriate, we have looked at SVG and Dynamic HTML as two possible options.

This is by no means an exhaustive list of the technologies that could be used to create online visualizations. These were originally chosen for review based on the apparent level of support that each has enjoyed and the expected possibilities. A review of other technologies such as Shockwave/Flash has been undertaken but has been omitted here due to space constraints.

SVG and Dynamic HTML can also be used for the basic user interface, including the placement of elements within the page and for providing interactivity. As we will discuss one could also use the more traditional Java interface through the use of an applet if that would be better. The final category for which technologies were reviewed was the integration of other forms of media, whether that be video, audio, or image based, that enhance the effectiveness of the visualization.

Before continuing our analysis of visualization technologies we will introduce two important tools mentioned in the subsequent discussion namely: Web Services and scene graphs. A Web Service is a fully functional program that can accept requests in a manner similar to a standard HTTP GET request and returns its response in standard XML using the SOAP protocol (Gerimenko & Chen 2005). This means that it can, for example, query a database based on user information, tailor the results based on user permissions and return complex data structures as required by the querying application.

The second concept that needs to be discussed is that of the scene graph. This can be thought of as a hierarchy of nodes describing transformations and objects that will dictate how a particular scene, either in 2 or 3 dimensions, will be rendered. It is normally a directed acyclic graph. The reason many technologies use this is that it results in the lower level details of managing graphics to be removed from the actual development of the scene. This idea is common in XML based representations of scenes and is held in contrast to a polygon based model whereby vertices are simply pushed through a rendering pipeline in the order they arrive and the graphics card then handles their organization.

4.1 Technologies for 2D Visualization

SVG, or Scalable Vector Graphics, is a 2D XML based vector graphics file type that was first formally made into a W3C Recommendation in 2001. The initial standard (1.0) incorporated basic 2D shapes, arbitrary polygons, text, and images with the ability to generate events and modify the document dynamically through ECMAScript. Further iterations such as SVG 1.1 and SVG Tiny have focused on providing further support for behaviours and events through the use of Java and an emphasis on ensuring that SVG can be rendered on almost any type of device. The next iteration of the SVG standard (1.2) will include support for animation, video and sound media, synchronization of elements, and support for streaming media (Jackson & Northway 2005).

Through the use of the Java.net package generic network communication becomes possible and even within ECMAScript some functions are provided that make this possible. One specific example of these functions that is worth mentioning is that through the Adobe SVGViewer plugin, which as is described later, it is possible to make asynchronous calls to Web Services (with getURL()) and then parse the resulting document into a DOM document fragment for easy inclusion in the SVG scene (using parseXML()). This means that SVG not only has communication support but also can include dynamically generated content easily. While these scripting options are sufficient for most calculations, our experience is that they are still rather inefficient.

	SVG	DHTML	VRML97	X3D	Java3D
Technical Capabilities					
Communications	Limited	Very Limited	Limited	Limited	Yes
2D	Yes	Yes	Yes	Yes	Yes
3D	No	No	Yes	Yes	Yes
Compression	Yes	No	Yes	Yes - encrypt	Yes
Animation	Very Good	Limited	Interpolation	Interpolation	Programmable
Ease of Creation	Easy	Easy	Moderate	Moderate	Difficult
Interactivity					
Intra-element events	Event Model	Event Model	Route Model	Route Model	Composite
Scripting Support	Java/JavaScript	JavaScript	Java/JavaScript	Java/JavaScript	Java
Dynamic Update	Yes (w/ Adobe)	Yes-limits	Plug-in dpndnt	Plug-in dpndnt	Yes
Inter-element communication	Limited	Full	IE Only	IE Only	Full
Support					
Plug-in ubiquity	Very Good	n/a	OK	Poor	OK
Standards Based	Yes(2001)	Yes	Yes(1997)	Yes(2003)	Yes
Cross-Platform	Yes	Yes	Yes	Not Yet	Yes
Application Specific					
Native Structures	Simple	No	Good	Good	Programmable
Previous Use	Many	Many	Prevalent	Emerging	Emerging

Table 1: An analysis of web-based technologies using our evaluation framework

In terms of encryption and compression SVG does support a compressed format of itself for faster downloads that is then decompressed into the normal DOM tree structure on the client. Encryption, however, is not considered part of the standard. Likewise user based rights or profiles could be included but are not supported natively.

SVG is a widely accepted format with a standard plugin (Adobe SVGViewer) being used almost ubiquitously. It should also be mentioned that the next generation of web browsers such as Opera 8 and Konqueror natively support SVG with other browsers soon following suit. This means that there should soon be no need for any additional downloads for users viewing visualizations made using SVG.

The second technology that we discuss for the visualization of 2D data is simple HTML with JavaScript support. This has recently gained popularity primarily due to certain applications to mapping such as Google Maps (Google 2005).

The advantage of this approach is that it requires no extra plug-in downloads and is compatible with all major browsers. Of course using the same language for the visualization as the containing markup means that we avoid the problems associated with inter-element communication but many of the features that are specifically included for visualization technologies are lost. As such compression, encryption and animation are not normally supported.

Dynamic updating through the use of XMLHTTP request objects allow for web applications to communicate and retrieve XML documents from Web Services or other sources in most browsers. It should be noted, however, that due to security restrictions communications are restricted to the originating server. This feature is particularly important for large datasets through which a user is likely to navigate predictably or selectively.

Using JavaScript as the language for programmability means that the level and type of interactivity obtained is similar to other technologies although it may be harder to obtain information on where in a particular element an event has occurred.

Recently there have been several high profile applications that use this technology such as Google Maps (Google 2005). This web-application allows users to view map and satellite imagery data over the entire world at varying resolutions. It also provides the ability to search for particular locations and place 'pins' at relevant locations, in essence dynamically changing the visualization to fit user requirements.

4.2 Technologies for 3D Visualization

Arguably the most commonly used technology for presenting 3D models on the web is currently VRML and more specifically the specification released in 1997 by ISO/IEC (Walsh & Bourges-Sévenier 2001) known as VRML97. This specification defines a file format using blocks to represent a scene graph and describe 3D scenes.

While the acronym VRML stands for Virtual Reality Modeling Language it is not confined to virtual reality. It has the bare minimum of geometric primitives and need not be immersive. Users can also define 'prototypes' to combine other node or primitive types that can then be used as if they were a built in type. These can both be defined within the same file or be treated as external resources, allowing for distributed scenes. It does provide a level of interactivity however through the use of 'sensors', and 'routes' through which messages and changes can be passed to influence other elements in the scene. This message passing model can also be enhanced through the use of either ECMAScript or Java scripting. Likewise animation is possible through the use of 'interpolators' that allow for an element's attributes to change according to time. Finally, this format also allows for data to be compressed when sending the scene to a client program

The event model is not particularly flexible or intuitive and lacks the control that other technologies offer. This view is reinforced by (Manoharan, Taylor & Gardiner 2002) who decided not to use VRML with JavaScript due to its lack of simplicity and control.

As this technology has been around for some time there are many different plug-ins and viewers available that adhere completely to the standard and can view any correct scene. These viewers often support some form of inter-element communication however this is largely dependent on the browser used. Some browsers impose restrictions by adhering only to the JavaScript standard and thus being unable to call methods on non-standard nodes (such as plug-ins).

Another implication of the length of time VRML has been around is that there are many examples of its use for visualizations. The first example that will be discussed

here, presented by (Neo, Lin & Gay 2004),is the example of using VRML to allow authors to publish 3D models in conjunction with their paper for viewing online. The idea being that small animations can reinforce and visualize the concepts presented in a particular paper. This work found that VRML was sufficient for providing interactive worlds with animations. However the authors also say that it is believed this idea has not been taken up due to the lack of tools for parsing a scene definition and creating the VRML file. As will be discussed this is not a limitation found in other technologies.

The second example of VRML in active use for visualization is that presented by (Gill, Caris & Smith 2004). In this paper the authors use block models in VRML for isosurface extraction in the field of mining and the tracking of ore. This visualization also uses dynamic updating by only loading the initial controls and then including content at runtime. This shows both the extensibility of visualizations using VRML and its novel application to a particular problem. The VRML scene also communicates with a Java Applet for control and to provide a user interface. The authors do, however, concede that it still requires experts to create the required visualizations. This paper also specifies, however, that all future work on the application will be done using X3D, the next technology under discussion here.

X3D is the successor to the aforementioned VRML97 standard. Introduced in 2003 it has now reached the level of a Final Draft International Standard. The largest difference between X3D and its predecessors is that it is XML based meaning that the whole range of associated tools can be used to more powerfully and easily manipulate scenes (Walsh & Bourges-Sévenier 2001).

Unfortunately it does not have the support base of some of the other technologies since it is a relatively new standard and there have not been many large scale uses of this technology. This results in what one author referred to as a 'browser minefield' (Niccolucci 2002) where there is no one standard implementation and no implementation that adheres completely to the standard. That said, more comprehensive and stable plug-ins have become available recently.

Some of the features supported by X3D include different formats allowing for the compression and encryption of scenes as they travel between the client and server, dramatically reducing load while increasing security. Different profiles also allow for different levels of adherence to the complete standard each of which support different node sets and requirements of the browser.

The fact that X3D is actually a form of XML document means that several advantages are obtained. The first of these is that we can easily be assured that a particular scene adheres to the specification by checking it against the publicly available DTD. However, perhaps of more use is the use of XSLT transformations which, as (Polys 2003) states, means that external data can easily be incorporated into an X3D scene. Likewise the scene can be traversed to add nodes where necessary and the problems of parsing a scene definition found in VRML are mitigated.

Examples of the use of X3D to provide visualizations on the web can be found in (McIntosh, Hamilton & van Schyndel 2005), (Polys 2003) and (Niccolucci 2002) where X3D is used for the visualization of UML diagrams, chemistry curricula, and archaeological information respectively.

The final option for 3D visualizations that will be discussed here is the use of Java3D. Java3D is quite different to the previous two technologies. It is a 3D graphics API originally introduced in 1997 that sits atop either OpenGL or Direct3D (Walsh & Bourges-Sévenier 2001). Unlike VRML and X3D, this means that, after programming is complete, the scene must be compiled. This does make it far harder to develop for and, while the emergence of visual development tools is predicted, this has not yet occurred to our knowledge. The Java3D addition to the Java API allows for the rendering of scene graphs similar to VRML and X3D and in fact several examples exist of Java3D being used to render VRML. The Web3D consortium has a working group set up specifically for this (Walsh & Bourges-Sévenier 2001) that makes the technology particularly flexible and allows it to, in part, inherit the advantages of VRML and X3D.

Because this technology is based on Java and can exist as a subset of an applet it has all the power of a traditional applet including user interface components and communications with the server. It is these communications that make it appropriate for the distributed viewing of the same dataset as it evolves over time.

An example of Java3D being used for an interactive visualization can be found in (Salisbury, Farr & Moore 1999) where distributed views of a military simulation environment were able to be formed where each viewer has a synchronized view of the simulation. This shows the considerably increased control and networking support that this technology offers over other options.

One major disadvantage of all three of these technologies is the necessity to download additional software. In the case of Java3D this involves not only the JRE but also the implementation of the Java3D library. That said it is the only one of the three that has support on some operating systems such as Mac OS X 10.4, which comes with the Java3D library already installed.

4.3 Interactivity Technologies

There are several technologies important in the context of this discussion that are not directly related to presenting the visualization icons. As such we do not apply the evaluation framework but instead discuss them in an informal manner with reference to their effect on the overall visualization, or system as it will be referred to.

The user interface and basic interactivity of our system is also of concern and is something for which we have several options. The first of these is to use the traditional web based tools such as CSS and HTML to create the interface and organize the various components of our visualization. The benefit of using these technologies is that they are widely supported by newer browsers and are well understood. Using certain tags it is possible to place elements absolutely within the page meaning that almost any interface design can be accommodated.

The second option for providing a user interface is to use SVG exclusively. This also allows the positioning of elements in an absolute way while additionally giving greater control over animation, repositioning etc. However the limitation of native support that has been discussed above is also a problem here meaning that this option is perhaps only viable if SVG was intended to be used elsewhere in the visualization.

Finally, if already using Java technology there is no reason that the interface types that can be created within a normal applet should not be possible here also. This allows for a more traditional stand alone application style interface but does mean that one loses some flexibility in the arrangement of elements.

4.4 Media Integration Technologies

In order to create a framework for developing powerful web-based visualization applications it is necessary to review two technologies that allow for the integration and synchronization of more traditional forms of media. This can be used to enhance the effectiveness of the visualization in question by providing supplementary materials that can, for example, demonstrate to users how they might use

the system or provide explanatory information regarding the contents of the visualization.

The Synchronized Multimedia Integration Language, or SMIL, is a markup standard organized by the W3 organization for creating synchronized presentations containing different multimedia content. Because of its XML basis it becomes possible to dynamically manipulate the presentation before presenting it as part of the visualization. This means that extra information pertinent to what the user appears to be doing can be incorporated. Likewise SMIL can cater for a variety of user preferences, such as language, by providing alternative content depending on regional settings.

One of the largest drawbacks of SMIL, however, is that many browsers do not yet natively support it. Instead standalone applications such as RealNetworks Real Player are used.

The second form of media integration we considered is the more traditional use of QuickTime or other common media players with the use of JavaScript to control the operation of the player. This constitutes a tried and true method that should work on most systems due to the prevalence of media players already installed. Unfortunately most browsers do not support the ability to call methods on media objects normally. Hence the level of control that can be afforded natively is quite limited.

5 The Visualization Framework

In this section we describe a framework of software and logical system architecture that offers flexibility for creating web-based visualizations. The proposed system architecture should meet the computational requirements of most visualizations by allowing for the flexibility of either immediate server side creation or an offline database of pre-created views.

The system architecture described in Figure 2 is what we believe delivers the most appropriate balance. Here Web Services can be employed for the actual creation and caching of visualizations while the actual interface is provided through a traditional web server. As such it is not necessary that all of the server side elements reside on the same machine allowing for a more flexible deployment.

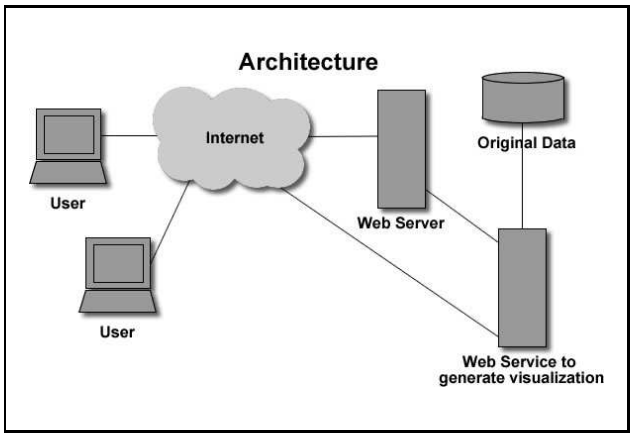

Figure 2: The proposed system architecture

On the web server it is possible to use several technologies to enhance the visualization. The first of these is server-side scripting and database support. Server-side scripting means that some processing based on user input can be done on the server. Likewise pages can be customized to specific datasets or users at request time. While we chose PHP, competing technologies could be used as well.

The use of databases to restrict access based on user profiles and customize experiences based on user preferences is also essential. This is done both to ensure a consistent experience across different uses of the application and enable only authorized users to view certain features or parts of the dataset. The reason for this is two-fold as there exists not only a need to restrict some users but also to avoid unnecessary confusion the full feature set may entail.

The technologies used for the actual page should revolve primarily around the flexibility and interactivity that scripting languages such as JavaScript provide. This enables inter-element communication and control, which is especially important between the embedded visualization and the rest of the interface. The creation of other windows and changes to the DOM of the web page itself are also possible. This is all done on the client with no need for communication with the server, overcoming the issue of latency and throughput that are inherent with web-based applications. Obviously there are restrictions on what can be achieved, however it is still preferable to only using server-side computation.

JavaScript can, however, be used for communication with a secondary web service as described below. This means that content can be dynamically selected, downloaded and integrated into the visualization. This includes the retrieval of supporting data that can reinforce or enhance the visualization.

Another common web technology that is important for providing a consistent and visually appealing user interface is CSS. The use of this enables greater control over the absolute positioning of HTML elements along with the ability to hide elements not in use. As a result, flexible interfaces, far more like those common to stand alone applications, can be created.

The use of both CSS and to a large extent JavaScript are dependent on using normal HTML as the encompassing markup language. While this does not offer the synchronization support of SMIL or the graphical interfaces SVG applications may have, it is extremely common, well understood, and easily developed. HTML does, however, have the capacity for embedding media elements such as audio and video through widely available plug-ins such as QuickTime. Some simple control over their operation is also possible.

Finally it must also be said that, as mentioned in section 4, Web Services are of importance for our system. In the context of our framework these are used to provide back-end computational services and create the actual visualizations themselves on the fly and ready for delivery to the client. A discussion on the applicability of web services for visualizations using SVG can be found in (Aulenback & Williamson 2002) although the concepts extend easily to the other technologies under discussion.

6 A Sample Application

To demonstrate the applicability of our framework to a real application we applied it to a commercial project that uses height fields to visualize survey data. The data used for these visualizations has several dimensions and as such a ‘cut’ based on chosen criteria can be used to view a particular dimension of the data. The requirements were that the application be able to generate the required models on the fly based on user selections, include not only the required 3D model but also the ability to query the underlying selected data, generate statistics, and synchronize with various types of media. We were provided with an existing Java program that generated VRML representations of the data based on a hard coded subset of cuts.

The framework described in the previous section was used as the basis for our final solution. PHP and an SQL database were used to ensure that user profiles and settings could be maintained, giving both added security and cus-

tomization. For our particular application further use of server side scripting was not required.

Creating the actual user interface and environment for the visualization involved the use of standard DHTML with a heavy emphasis on JavaScript as described. This was chosen as the greater control and animation support of SVG was not needed while the ease of embedding other plug-ins was. Likewise the advanced synchronization capabilities of SMIL and its derivatives were not necessary, especially in light of the lack of native browser support. As such Quicktime movies and audio were used to supplement the visualization where appropriate.

The original program was also extended to include a web service interface and to allow the selection of a particular visualization. This program was also modified to create X3D files as these were far more easily manipulated.

As such X3D constitutes the chosen technology for displaying the 3D visualization and integrates with other elements of the visualization. The reason this technology was selected was that while the program given generated VRML this is more difficult to manipulate. By converting this to X3D it become possible to easily incorporate the necessary level of interactivity and expected actions associated with user input by manipulating the XML structure of the document.

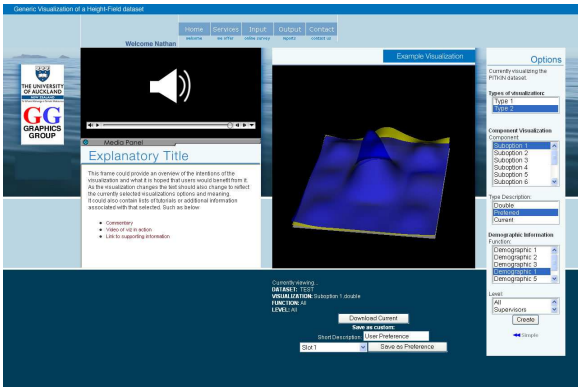

Figure 3: An example screenshot of our application at work.

We now discuss some of the features that our actual implementation offered and involved. An example of the web application at work demonstrating the arrangement of various elements and the overall design can be found in Figure 3. The main visualization itself consisted of an X3D visualization of a height field using the native 'ElevationGrid' structure (Walsh & Bourges-Sévenier 2001).

For viewing the X3D models we chose to use Bit Management's BS Contact (Bit Management GmbH 2005). This plug-in still has some rendering issues, such as the blending of semi-transparent surfaces, but seemed to support the greatest number of required features. Consequently we currently are reviewing several newer options. Apart from simply displaying the data in the form of a height field, the scene also included interactivity that allowed users to select either points or portions of the surface and, through a second query to the web service, be shown the underlying data that made up that section. This was important for our specific application as the data came from a variety of demographic sources and it was important to be able to identify an underlying trend in an unexpected aberration.

The second instance in which this facility was important was due to the fact that the visualization was designed to be used and understood by a wide variety of users. The ability to provide not only a visual representation but also hard data means that both sensate and creative people are catered for. Likewise the generation of information on the statistical integrity of the visualization based on the cut of the dataset shown was also important for this.

Different users were intended to have different levels of access to the data and different abilities in selecting which cuts of the dataset to create. By including different panels that either show a generic set of cuts or an interface for selecting any cut by any demographic or dataset this was possible. Linked to this separation of user access was the ability for a user to save a particular cut for later dissemination. This was also to be persistent across accesses.

Some of the features that were also included were the ability to have supplementary information included in a panel beside the main visualization. This was intended to enhance understanding by explaining in a generic fashion what the visualization was currently showing. However, because we also included the ability to have generic 'cuts' of the dataset as decided by the creator of the data, extra explanatory information including a detailed analysis and audio/video media created specifically for the dissemination of each cut was also possible through this panel. As has been mentioned QuickTime was chosen for this over SMIL due to the lack of browser support and the fact the visualization would not have been significantly enhanced through synchronization.

7 Discussion and Related Work

The presented work consists of two major parts. The first is the framework for comparison between different visualization technologies and the second is a generalized framework for developing powerful web-based visualization applications.

In reviewing previous works that have attempted to compare different technologies we found that few seek to compare technologies. Most related works seem to fall into two categories. The first seek to extol the virtues of a particular technology or set of technologies to the field of visualization and the second to apply a technology to a particular problem. An example of the first would be (Gerimenko & Chen 2005). This recently published work includes a series of articles on SVG and X3D and their application to the field of Information Visualization. A discussion of the use of these technologies for providing user interfaces to Web Services is presented as is other related applications such as Interactive TV publishing. The newness of this work illustrates how these are emerging technologies that are only now gaining acceptance. A second example of this form of work that was published less recently is (Walsh & Bourges-Sévenier 2001). Here a discussion of the various technologies that are of interest to the Web3D Consortium are reviewed and while visualization is not the main emphasis of the work the concepts discussed are directly relevant.

There are also those who have applied a technology to a particular problem and discuss the merits of the chosen technology. These have been mentioned where relevant throughout the text but specific examples include the use by (Gill et al. 2004) of VRML as discussed in section 4.2. (Polys 2003) discuss the use of Web3D and specifically X3D for a chemistry curricula. Stylesheet transformations are cited as the deciding factor illustrating that in some cases alternatives are not relevant due to one particular feature. As a final example (Duignan, Biddle & Tempero 2003) evaluate extensively the applicability of SVG to the problem of software visualization.

The second part of this work is the framework presented for showing visualizations with the ideal being that whichever technology best fits the needs of the creator can be substituted as required. As this constitutes a user in-

terface for information visualization it can be evaluated as such. However because we are attempting to evaluate a framework we thought it more appropriate to evaluate a particular implementation of the framework as discussed in section 6. (Plaisant 2004) discusses the various types and troubles that are associated with evaluating information visualizations and defines four different evaluation practices. One of the methods described is the use of case studies of people using the tools in a realistic setting. At present we are in the process of applying this technique and will hopefully have data in the near future.

8 Conclusion

As the prevalence of web-applications continues to increase it is inevitable that this form of application will be extended to the process of visualization. Many works have already been presented that seek to use a particular technology to visualize a particular type of data or discuss the merits of a particular technology. We have developed and successfully applied in practice an evaluation framework to the most prevalent technologies available today and the creation of a framework within which the chosen technology might be employed.

This evaluation framework was used to categorize the 3D technologies of X3D, VRML and Java3D and the technologies for 2D visualization SVG and DHTML. While many of the measures in this framework are somewhat subjective we believe it provides a good basis for choosing between the different options dependent on a user's requirements. This follows the general consensus that there is no one tool or technology that is best suited for all visualizations.

Of course the choice of technology or creation of a particular visualization is only half the problem when creating an interactive web-based application as those under discussion here. As such we have created a framework of elements and technologies within which a particular choice can be placed with all the support needed for an effective visualization. An example application was also presented that uses this framework and solves a current problem, illustrating the appropriateness of this framework.

References

Aref, H., Charles, R. & Elvins, T. (1994), Scientific visualization of fluid flow, *in* C. Pickover & S. Tewksbury, eds, 'Frontiers of Scientific Visualization', Wiley Interscience.

Aulenback, S. & Williamson, R. (2002), 'SVG as the visual interface layer to the emerging web services market', *SVG Open / Carto.net Developers Conference* .

Bit Management GmbH (2005), 'BS Contact VRML/X3D Release 6.2', Last seen 18/08/2005. URL: `http://www.bitmanagement.com/`.

Duignan, M., Biddle, R. & Tempero, E. (2003), Evaluating scalable vector graphics for use in software visualisation, *in* 'CRPITS '24: Proceedings of the Australian symposium on Information visualisation', Australian Computer Society, Inc., Darlinghurst, Australia, Australia, pp. 127–136.

Gerimenko, V. & Chen, C. (2005), *Visualizing Information Using SVG and X3D: XML-based Technologies for the XML-based Web*, Springer, London, UK.

Gill, T., Caris, C. & Smith, G. L. (2004), Interactive web-based visualisation of block model data, *in* 'Web3D '04: Proceedings of the ninth international conference on 3D Web technology', ACM Press, New York, NY, USA, pp. 23–28.

Google (2005), 'Google Maps', Last seen 18/08/2005. URL: `http://maps.google.com/`.

Jackson, D. & Northway, C. (2005), 'Scalable vector graphics (SVG) full 1.2 specification', W3C Recommendation. URL: `http://www.w3.org/TR/SVG12/`.

Manoharan, T., Taylor, H. & Gardiner, P. (2002), A collaborative analysis tool for visualisation and interaction with spatial data, *in* 'Web3D '02: Proceeding of the seventh international conference on 3D Web technology', ACM Press, New York, NY, USA, pp. 75–83.

McIntosh, P., Hamilton, M. & van Schyndel, R. (2005), X3D-UML: enabling advanced UML visualisation through X3D, *in* 'Web3D '05: Proceedings of the tenth international conference on 3D Web technology', ACM Press, New York, NY, USA, pp. 135–142.

Neo, K. S., Lin, Q. & Gay, R. K. L. (2004), A web-based system for interactive visualization of scientific concepts, *in* 'VRCAI '04: Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry', ACM Press, New York, NY, USA, pp. 155–158.

Niccolucci, F. (2002), 'XML and the future of humanities computing', *SIGAPP Appl. Comput. Rev.* **10**(1), 43–47.

Plaisant, C. (2004), The challenge of information visualization evaluation, *in* 'AVI '04: Proceedings of the working conference on Advanced Visual Interfaces', ACM Press, New York, NY, USA, pp. 109–116.

Polys, N. F. (2003), Stylesheet transformations for interactive visualization: towards a Web3D chemistry curricula, *in* 'Web3D '03: Proceeding of the eighth international conference on 3D Web technology', ACM Press, New York, NY, USA, pp. 85–ff.

Salisbury, C. F., Farr, S. D. & Moore, J. A. (1999), Web-based simulation visualization using Java3D, *in* 'WSC '99: Proceedings of the 31st conference on Winter simulation', ACM Press, New York, NY, USA, pp. 1425–1429.

Tufte, E. R. (1983), *The Visual Display of Quantitative Information*, Graphics Press, Cheshire, Connecticut, U.S.A.

Wakita, A. & Matsumoto, F. (2003), 'Information visualization with Web3D: spatial visualization of human activity area and its condition', *SIGGRAPH Comput. Graph.* **37**(3), 29–33.

Walsh, A. E. & Bourges-Sévenier, M. (2001), *Core Web3D*, Prentice Hall PTR, Upper Saddle River, NJ, USA.

Wuensche, B. & Lobb, R. (2001), A scientific visualization schema incorporating perceptual concepts, *in* 'Proceedings of ICVNZ'01', pp. 31–36.