

An enhanced web-service for scientific data visualization

ASHRAF S. HUSSEIN, MOHAMMED S. ABDELWAHAB, MOHAMMED F. TOLBA
Scientific Computing Department,
Faculty of Computers and Information Sciences, Ain Shams University
Abbassia, 11566, Cairo
EGYPT

Abstract:- This paper summarizes the progress achieved through the build of a scientific data visualization web service; a client-server system that is designed to acquire visualization requests from users connected to a web browser, perform the visualization process and provide them with the visualization results. Interacting with the visualization results is another issue that has to be taken into account. Providing results using VRML gives the opportunity to navigate the visualization and makes the visualization as an analysis tool rather than just a tool for the presentation of results. The different trade offs and optimization parameters associated with the VRML output are discussed through the presentation of an optimized framework aiming to provide interactive, fast, portable and easy to use services. The framework is tested against different visualization techniques to ensure its adaptability to each of them. An average reduction factor of 88 % is achieved to the final VRML resultant file and an average speed up percentage of 10 % is achieved to the execution time of the visualization process.

Key-Words: - Scientific visualization, VRML, internet services and knowledge discovery

1 Introduction

Total Cost of Ownership (TCO), a new variable has been added recently to the complicated system of equations governing the design of any commercial application. Low cost or even no-cost scientific services will significantly lower the entry barrier of the use of scientific applications.

Scientific data visualization is considered one of the earliest scientific fields which takes the advantage of this philosophy. The WWW environment offers an excellent opportunity of providing data visualization techniques to the general public. The final goal is to have the visualization technology be available on every desk in a transparent manner.

Recently, many authors [1] discussed the applicability of the scientific data visualization especially over the WWW environment. Pioneering work on the Web-based visualization was done by Ang et al. [2] they exploited the MIME*-typing concept to allow visualization data to be sent over the Web and be processed by a Web browser. They linked their medical visualization system (VIS) to

the Mosaic browser via its CGI[†] interface: when data of MIME-type was retrieved by the browser, it was passed to the VIS as helper application.

After this contribution, a lot of trials for the efficient use of the WWW platform to support scientific data visualization have been introduced specially for large volumetric data. All these trials can be classified into three main approaches; the first approach [3] makes minimal use of the Web, relying on it simply as a data communication medium and exploiting the MIME-typing concept to divert data to a particular application. Users continue to use their own software on their own platform.

The second approach is the Java applet technique which was designed for the cases where the data is closely associated with the software, and indeed located on the same server [4]. The server hosts an instruction file specifying the data to be visualized, and the initial pipeline to be created. On receiving this file, the browser invokes an application that interprets the file and executes the commands. Although that this approach allows scientific visualization applications to be platform independent [5], however, this is inappropriate as

* Multipurpose Internet Mail Extensions

† Common Gateway Interface

the data is usually associated with the user rather than the software provider.

The third approach is a very general approach taken by Wood et al [6], rather than transfer visualization data or software across the Web, a proposed system was designed in which the visualization was executed on a remote server, and the resulting graphics returned over the Web for viewing in a browser. Several systems have been proposed supporting this branch for example; the authors of the visualization toolkit (VTK) [7] explain how to build a VTK server-based visualization service. The form-based interface of these systems is simple but inflexible.

All these trials [[2],[6], and [7]] were focused mainly on discussing how the data and the visualization results should be manipulated and transferred over the WWW-based environment without taking into consideration one of the most important factors which is interactivity as a key facility to improve the process of getting insight in large data sets. So, the need to a new emerging technology like virtual reality for the visual representations was driven by the increasing importance of the intuitive real world interaction, realistic representation [[8],[9]] and the usage of the web as a rich distributed computing environment specially after the new integration between the internet computing and virtual reality through the VRML[‡]. VRML is not only promising to support interactivity which is a missing part in most of WWW-based scientific visualization applications but also it provides a missing level of portability for visualization systems [10].

Technically speaking, VRML is neither a virtual reality nor a modeling language. At its core, VRML is simply a three-dimensional interchange format [11], which can be considered as three-dimensional analog to HTML. This means that VRML serves as a simple, multiplatform language that supports the publishing of three-dimensional worlds over the WWW platform.

With respect to introducing VRML for scientific data visualization, current research is categorized into two main threads. The first thread seeks for the exploit of primitive VRML ISO standard in order to support the needed functionalities. The second

[‡] Virtual reality modeling language

thread tries to extend the current VRML standard for specialized usage in the field of scientific data visualization [[12], [13]].

There is no doubt that the new special adding to the VRML standard to support scientific data visualization applications will result in better performance, higher flexibility and ease of use [12]. On the other hand, using such new standard requires special purpose VRML browsers, which mean lower level of portability. The approach followed in this work is to optimize the usage of the current VRML ISO-standard to provide the user with the highest quality of service and the fastest service-time.

2 The New Framework

Dealing with the customer requirements for scientific services is quite different from dealing with the other regular customers. The huge amount of data, the required accuracy level, the limited computational resources, in addition to the low communication speed make providing such services a very difficult goal. Providing a scientific data visualization service adds another obstacle which is interactivity. So, it is clear that a scientific data visualization web application should provide their customers with an accurate, interactive, portable service with an acceptable service-time taking into consideration the environment constrains [14].

2.1 Interactivity

From the discussed review of the previous work, it can be concluded that the use of VRML as a three-dimensional graphical file-format to represent the output of the visualization process enables the framework to be portable and the user to easily interact with the resultant world especially after the huge addition of the action scripts to the VRML 2.0 standard.

Although this ability had facilitated the interaction of the user with the visualization results, there are still a lot of limitations on the user interaction especially in some visualization techniques which need an interaction between the user and the underlying data used to generate the VRML resultant world [15].

As shown in Fig. 1 the implemented framework will adopt the third branch discussed before [6], by adding some changes to it which enables the server to store the user data file for a certain period of time.

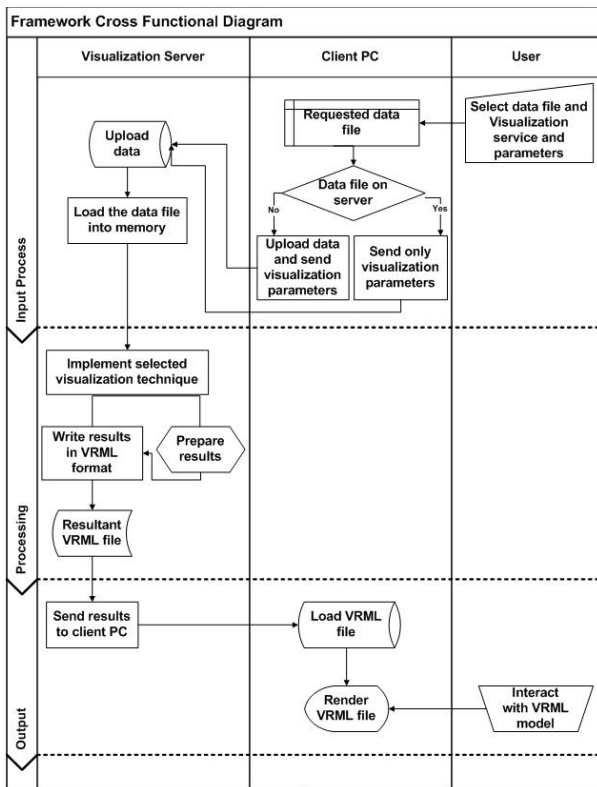


Fig 1. Schematic drawing for the interaction between clients and server

Although this addition may increase the needed storage required by the server, it will enable the user to interact with the underlying data without the need to upload it each time a visualization service is requested or visualization parameters are changed on the current data. This addition in the architecture will decrease the service time, the communication overhead and the bandwidth required from the server. In addition, the output VRML world will contain action scripts which enable the user to change the light source properties such as color and position and interact with the whole world using a keyboard based simulation to a trackball. This enables the framework to decrease the interactivity limitations and provide the user with customized

interactive controls independent on the VRML viewer capabilities.

2.2 Service Time

The service time is the most important factor affecting the performance of a web application providing a scientific data visualization services. As shown in Fig. 2 the service time consists of three components. The first component is the upload time of the data file (to be visualized) to the server, and this component is not controllable because it highly depends on the data file size and the network speed. The second and the third components are the processing time and the download time of the resultant VRML file to the client. The third component is a direct function of the file size and consequently in the second component. The more reduction in the file size the more processing time required to generate the VRML file.

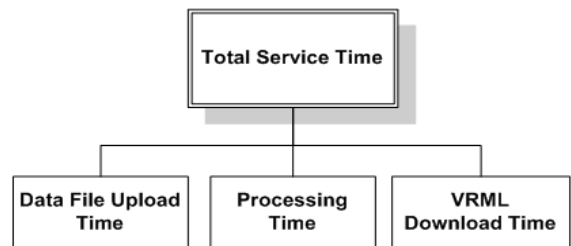


Fig 2. The parameters affecting the final service time

In order to achieve best performance, the implemented framework presents optimized solutions for the reduction of the service time by reducing the processing time needed by the visualization process and by the efficient reduction of the resultant VRML file size without affecting the overall processing time.

2.3 VRML File Size

The resultant VRML file size is considered one of the most important leading factors in implementing any VRML based scientific data visualization service especially over low communication speed networks such as internet. VRML is a UTF-8 file format that has a lot of tricks which enable the designer to decrease the required file size to

represent the resultant world. These features are used to develop a sophisticated model for file size optimization. In order to decrease the resultant VRML file size, the application will use a lot of methods some of them are usually used and the others considered one of the main contributions of the present work.

The new developed model to reduce the file size depends mainly on one of the most important features of VRML which is the ability to combine more than one polygon in the same VRML node. This feature will enable the framework not only to remove unneeded nodes headers but also enable it by using some additional computation to remove the duplicated point position and substitute it with its index. Due to the sharing of vertices between different polygons, this reduces the file size with an average percentage of 87.63323%, where the reduction factor of file size is calculated according to equation (1).

$$\frac{\text{file size before optimization} - \text{file size after optimization}}{\text{file size before optimization}}$$

(1)

After many investigations, it is found that the percentage reduction in the file size is affected by the number of duplicated vertices indices and the number of polygons per node which reflect directly on the number of digits needed to uniquely identify the polygons within the current node. This means that the reduction in the file size is dependent only on the characteristics of the data file and cannot be predicted in advance, generally the most optimum file size is close to the case of which all polygons are written in the same node.

In the developed solution, eliminating the duplicated points using sequential search for a vertex represented by a regular XYZ coordinate-representation will consume a lot of time even if the array of points is sorted. So, there was a need for an algorithm that avoids both searching in the three-dimensional space and the radix sort overhead specially in case of the huge amount of data points. To solve this problem, the framework uses a very simple technique which is based on finding whether a particular point was previously used or not. The application uses the point XYZ coordinate to generate a string key used to index a hash table. By

this way, the points in the hash table do not need to be sorted and the search time will be significantly reduced.

2.4 Processing Time

In order to decrease the overall processing time, the server side of the framework is implemented as a highly optimized multithreaded producer-consumer like dynamic link library. The process of generating VRML file contains mainly two separate processes; the first process is the preparation of the nodes data which will be written to the VRML file and the second process is writing the VRML file to the storage device. The two processes can be separated and implemented as producer-consumer architecture [16], where the first process represents the producer and the second represents the consumer.

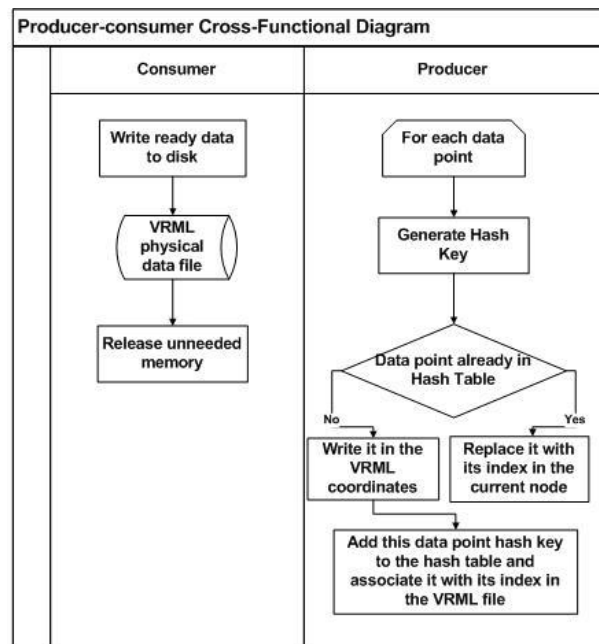


Fig 3. Schematic drawing illustrate the produced-consumer architecture usage

Using this architecture enables the environment to begin the file writing process without waiting for the whole preparation process to complete. Due to that both processes use different computational resources (the processor for the first process and the

storage device for the second), the process of eliminating the duplicated points which previously illustrated will not delay the process of file generation as it is known that the dealing with the storage device is much slower and takes more time.

3. Implementation of Visualization Techniques

The present framework enables the implementation of data visualization techniques. Each technique is based on the same concept of multithreading producer consumer architecture with some slight modifications.

For all the implemented visualization techniques, two producer threads are running concurrently. The first one is the process of generating the visualization output implemented on the input data. The second process is the generation of the wire frame model of the object. The two threads are sharing the processor resource while only one consumer is running for writing the result data to the storage device. This is done in order to maintain an organized representation to the VRML file structure.

The modification implemented in the algorithms of the visualization techniques concerns the way of constructing the hash key used to index the hash table in the process of removing the duplicated points. In the implementation of both line contour and isosurface techniques- as a sample of the scalar visualization techniques- Fig. 4 and 5, the string key representing the new calculated contour point on each polygon edge is constructed using the two end vertices of this edge. Since no edge can contain more than one point in the same contour level, using the two end vertices of that edge can uniquely define the resulting contour point.

```

For each polygon in the data set do
If the used shape number of vertices is higher than 3 then
  Triangulate the used shape
For each triangle edge do
  Construct the hash key string for this edge using the
  two edge end points
If the hash key exists in the hash table then
  Place its associated index in the current VRML node
Else
  Construct the hash key string by reversing the two
  edge end points
If the hash key exists in the hash table then
  Place its associated index in the current VRML node
Else
  Calculate the line contour point of the current
  contour value,
  Add the point coordinate to an array of points,
  Add point index to the hash table using the
  constructed key string,
  Add point index to the current VRML node.

```

Fig 4. PSEUDO-CODE for line contour producer

```

For each polygon in the data set do
If the used shape number of vertices equal 8 then
  Calculate the binary code associated with this polygon
  vertices
For each polygon edge do
  Construct the hash key string for this edge using
  the two edge end points
If the hash key exists in the hash table then
  Place its associated index in the current VRML node
Else
  Construct the hash key string by reversing the
  two edge end points
If the hash key exists in the hash table then
  Place its associated index in the current VRML node
Else
  Calculate the line contour point of the current
  contour value,
  Add the point coordinate to an array of points,
  Add point index to the hash table using the
  constructed key string,

```

Fig 5. PSEUDO-CODE for isosurface producer

The algorithm of flooded contour depends in the generation of each colored filled region on the idea of constructing a continuous closed intervals located between each two consecutive contour levels and then filling these regions with the appropriate color [17]. In this situation, the generated contour points on each edge is uniquely defined by using the two end vertices of this edge and the current contour level, since it is possible for more than one level to pass across the same polygon edge.

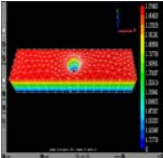
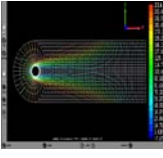
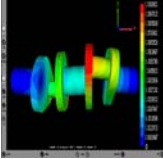
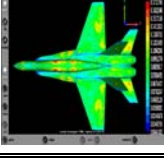
For each polygon in the data set **do**
If the used shape number of vertices is higher than 3 **then**
Triangulate the used shape
For each triangle edge **do**
Initialize array of indices,
Construct the hash key string for this edge using
the two edge end points and the current contour
value
If the hash key exists in the hash table **then**
If contour point index does not exist in the
array of indices **then**
Add contour point index to array of indices
Else
Construct the hash key string by reversing the
two edge end points and the current contour
value
If the hash key exists in the hash table **then**
If contour point index does not exist in the
array of indices **then**
Add contour point index to array of
Indices
Else
Calculate the line contour point of the
current contour value
If contour point index does not exist in the
array of indices **then**
Add contour point index to array of
indices,
Add point coordinates to array of points,
Add point index to the hash table using the
constructed key string
For each vertex in the triangle **do**
If the current contour value \leq vertex value \leq
next contour value **then**
Add the point to the array
If the number of polygon vertices added to the array =3 **then**
Color the whole polygon with the current contour color
Else
Calculate the next contour color points using the
same procedure above,
Add the result indices to the array,
Arrange the array of indices according to their
associated points in the array of points,
Write the arranged indices to the current VRML
node.

Fig 6. PSEUDO-CODE for flooded contour procedure

4. Analysis and Discussion

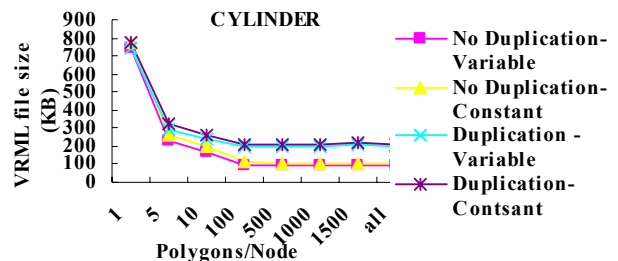
In this section, the two major parameters affecting the service time which are the execution time and the VRML file size are evaluated using sample test cases presented in Table 1.

Table 1. The test cases used to validate the new framework

Title	Number of Polygons	No of Polygon Vertices	Computational Grid Type	Sample Output
OUTSURF DATA (FEBRICK DATA)	1000	8	Unstructured	
CYLINDER	1760	4	Structured	
FE-VOLUME KURBEL3F BRICK DATA, SFB393	3424	4	Unstructured	
3D FE DATASET	7366	4	Unstructured	

4.1 Analysis of VRML File Size

Fig. 7 shows, for each test case, the relation between the VRML file size and the number of polygons written per VRML node according to the two main variables affecting the final file size; the number of duplicated vertices indices and the maximum number of digits used to identify the polygons.



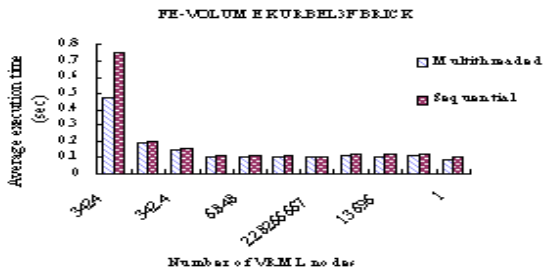


Fig 8. Execution time performance over dual processor

As shown in the figure, the main factor affecting the execution time is the process of writing the VRML nodes to the storage device (the consumer thread). The execution time reaches its highest value in case of writing each polygon in a separate VRML node and this value decreases rapidly with the increase of number of polygons written per node until this decreasing trend becomes approximately flat at a certain point. This point appears when change in number of written nodes is small; the value of the execution time is approximately the same -oscillating within small interval around a threshold value- and this small difference depends on the effect of eliminating the duplicated points in the process of generating the VRML nodes (the producer thread).

Table 2 illustrates the speed up percentage of the execution time, resulting from running both the multithreaded and the conventional sequential algorithms over a dual processor environment.

Table 2. Speed Up percentage of execution time

Average Min Speed Up	1.63536 %
Average Max speed Up	26.0157 %
Average Speed Up	9.2793 %

The speed up factor is calculated using equation (2),

$$\frac{\text{Sequential processing time} - \text{multithreaded processing time}}{\text{Sequential processing time}} \quad (2)$$

5. Conclusion

Using VRML in scientific data visualization especially over the WWW-platform enables it to be more interactive and portable. Different factors affecting the implementation of such commercial scientific web-services can be optimized to increase the quality of scientific service produced. The main leading factor in such environment is the communication bandwidth and speed, which represent a great obstacle. Reducing the final resultant VRML file size without highly affect the final execution time will affect highly on the final service time and the needed communication bandwidth and speed. Different factors affecting the final VRML file size such as the number of polygons written per VRML node are optimized, in order to best adopt the VRML file architecture which leads to a great decrease in the VRML file size.

References:

- [1] K. Brodlie, S. Lovegrove and J. Wood, UK team surveys web-based visualization, in *ACM SIGGRAPH Computer Graphics*, ACM Press New York, NY, USA, February 2000, pp. 10-12, (See also Webvis website: <http://www.scs.leeds.ac.uk/vis/webvis.html>).
- [2] C. S. Ang, D. C. Martin and M. D. Doyle, Integrated control of distributed volume visualization through the World Wide Web, in *Proceedings of IEEE Visualization94*, R. D. Bergeron and A. E. Kaufman, ed., 1994, pp. 13-20.
- [3] Vis5D website. See: <http://www.ssec.wisc.edu/~billhT/vis5d.html>, 1999.
- [4] NPAC. The NPAC Visible Human Viewer website, <http://www.npac.syr.edu/projects/vishuman/VisibleHuman.html>, 1999.
- [5] C. Michaels and M. Bailey, VizWiz: a Java applet for interactive 3D scientific visualization on the web, in *Proceedings of the 8th conference on IEEE Visualization97*, IEEE Computer Society Press Los Alamitos, CA, USA, 1997, pp. 261-ff.

- [6] J. Wood, K. W. Brodli and H. Wright, Visualization over the WWW and its application to environmental data, in *Proceedings of the 7th conference on IEEE Visualization96*, R. Yagel, G. M. Nielson, ed., IEEE Computer Society Press Los Alamitos, CA, USA, 1996, pp. 81-ff.
- [7] W. Shroeder, K. Martin and W. E. Lorensen, *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*, Prentice-Hall, 1996. (See also vtk website: <http://www.kitware.com/vtk.html>).
- [8] S. Bryson, Virtual reality in scientific visualization, in *Communications of the ACM*, ACM Press New York, NY, USA, May 1996, pp. 62-71.
- [9] H. Haase, M. Gobel, P. Astheimer, K. Karlsson, F. Schroder, Th. Fruhauf and R. Ziegler, How Scientific Visualization can benefit from Virtual Environments, *CWI Quarterly*, The Netherlands, 1994, pp. 159-174.
- [10] J. L. Wesson and P. R. Warren, Interactive visualization of large multivariate datasets on the world-wide web, in *ACM International Conference Proceeding Series*, Australian Computer Society, Inc. Darlinghurst, Australia, 2001, pp. 151-157.
- [11] ISO/IEC 14772-1, the virtual reality modeling language, 1997. <http://www.vrml.org/Specifications/VRML97>
- [12] R. Ginis and D. Nadeau, Creating VRML extension to support scientific visualization, in *Proceedings of the first symposium on Virtual reality modeling language*, ACM Press New York, NY, USA, 1995, pp. 13-20.
- [13] J. Behr, M. Alexa, Volume Visualization in VRML, in *Proceedings of the sixth international conference on 3D Web technology*, ACM Press New York, NY, USA, 2001, pp. 23-27.
- [14] W. Lefer and J. M. Pierson, Visualization Components on the Internet, in *proceedings of the 3rd International Conference on Visual Computing (ICVC)*, Mexico City, Mexico, September 18-22, 2000, pp. 104-112.
- [15] M. Jern, Information drill-down using Web Tools, In *Proceedings of IEEE Visualization97*, IEEE Computer Society Press Los Alamitos, CA, USA, 1997.
- [16] W. Stallings, *Operating systems, Internals and design principles*, Upper Saddle River, NJ: Prentice Hall, 1998.
- [17] M. F. Tolba., A. S. Hussien, A. H. Aziz and H. E. Ibrahim, A fast Algorithm for colored filled-regions contouring representation, in *Proceeding of first International Conference on Intelligent Computing and Information Systems (ICICIS)*, Cairo, Egypt, June 24-26, 2002, pp. 404-409.