

Progressive Iso-surfaces on the Web

Klaus Engel, Roberto Grosso, Thomas Ertl

Computer Graphics Group
Universität Erlangen-Nürnberg, Germany *

Abstract

Visualization of volume data on the WWW using the Virtual Reality Modeling Language (VRML) and the Java programming language offers new perspectives for distributed and platform independent applications. A naive approach would either transfer the volume data to the client side for local processing or it would compute the iso-surface on the server side by a marching cubes algorithm and transfer the resulting polygons to the client. In both cases the network bandwidth would be the limiting factor for large data sets. In this paper we try to overcome this restriction by using a progressive iso-surface algorithm, which allows the generation of surface hierarchies from the volume data with a level-of-detail control (LOD). The surface hierarchy is generated on the server and the levels of resolution are transmitted to the client progressively, i.e. only the difference between two successive levels is sent across the network. In order to be platform independent on the client side, a Java applet running in a web-browser is used to reconstruct the hierarchy progressively. The visualization of one level of resolution is done by a VRML web-browser plug-in or the Java3D API, which allows the use of 3D graphics hardware support. In order to avoid the transmission of a large number of polygons, the user can reconstruct the surface up to a coarse level of resolution and refine it if required. The surface hierarchy can be visualized at any level of detail up to the finest level available, while the applet continuously reconstructs the hierarchy progressively.

1 Introduction

Due to the exponentially increasing size of volume datasets, volume data base servers are nowadays often used to store and maintain such datasets. These servers can be accessed by a large variety of client systems, ranging from low end PCs to high end graphics workstations. The visualization of volume data requires distributed applications, which allow the inspections of the volume data interactively by balancing rendering quality and real time performance. This issue is very important for distributed applications on the World Wide Web (WWW), which still suffers from network bandwidth restrictions. In this paper we focus on distributed volume data visualization using iso-surfaces.

We developed a new web-based iso-surface visualization tool which uses the concept of progressive iso-surfaces introduced in previous work [3]. The application is located at <http://www9.informatik.uni-erlangen.de/eng/research/prolet/> and is based on the client-server paradigm. The server generates the multi-resolution surface from the volume data and sends it progressively to the client which is a Java applet running in the web-browser. The user can select a data set from different application areas such as medicine or CFD. A coarse level of resolution can be selected for inspecting the volume data at

* Lehrstuhl für Graphische Datenverarbeitung (IMMD9), Universität Erlangen-Nürnberg, Am Weichselgarten 9, 91058 Erlangen, Germany, Email: engel@informatik.uni-erlangen.de, URL: <http://www9.informatik.uni-erlangen.de>

different iso-values, which requires low network throughput and therefore enables interactive rates. If needed, the surface can be refined up to the finest level of resolution, which will require a longer network transmission. In that case all levels of resolution can be selectively visualized by the user. This can be useful in the case that the client side does not have 3D graphics hardware support for polygon rendering.

On the client side, our web application is completely written in Java, which evolved to become the ideal programming language for web-based distributed applications during the past years. However, Java still suffers from performance limitations, especially when rendering complex 3D scenes. The Virtual Reality Modeling Language (VRML) [5, 1] offers the possibility to visualize 3D data on the internet using the full 3D acceleration hardware of a given client. An external Java applet is able to get access to a VRML plug-in, embedded into a web page, using the External Authoring Interface (EAI) [7]. This can be used to create geometry dynamically. We also implemented a version, which uses the new Java3D API [8] to visualize the iso-surfaces.

In the following section we will discuss related work in the field of distributed web-based volume visualization applications and progressive iso-surface generation algorithms. Section 3 discusses implementation details. The transmission and the visualization of the iso-surfaces is controlled by a java applet, which will be explained in Section 4. In Section 5 we present some results of our performance tests on a ISDN and Ethernet connection of the Java3D and the VRML implementation. We will conclude the paper with some remarks on future activities.

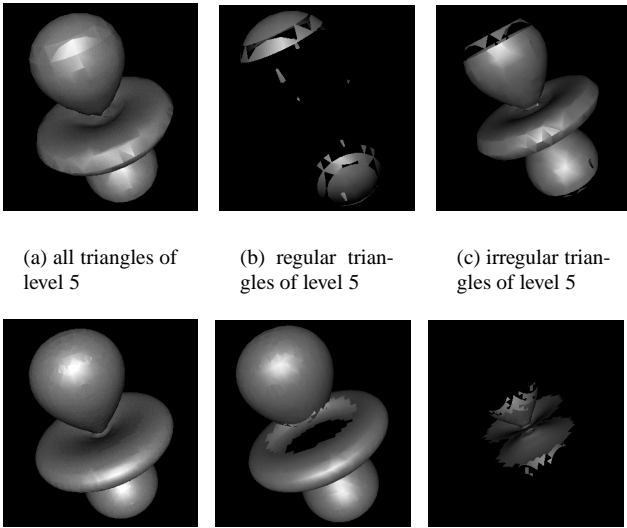
2 Related Work

In the past years several approaches for scientific visualization on the web were investigated. Trapp and Pagendarm presented a prototype of a WWW-based flow visualization service [4]. They allow arbitrary user data to be sent to a visualization server by a copy-and-paste operation from a data producer application into a HTML form. The server generates a VRML model of the visualization of the data, which is requested by a Java applet and visualized by the browser VRML plug-in. The data transfer between client and server is done using plain ASCII text, which requires a very high network throughput and processing time. The major drawback, however, is that the geometric data produced by the server has to be transmitted to the client completely, before it can be visualized, thus limiting the possibilities of the user to control the data visualization interactively.

Another interesting approach is the VisWiz [2] platform independent visualization tool for iso-surfaces, cutting planes and elevation plots for 2D and 3D datasets.

One of the first progressive applications for volume visualization was presented in [6]. Their system is based on the local reconstruction of wavelet-compressed data, but since it works in the Fourier domain it is restricted to X-ray like images.

In this paper we present a system based on iso-surfaces which makes a better and more balanced use of network and computing resources. Our system transmits as few data as possible and exploits



(a) all triangles of level 5 (b) regular triangles of level 5 (c) irregular triangles of level 5

(d) all triangles of level 6 (e) regular triangles of level 6 (f) irregular triangles of level 6

Figure 1: Isosurface of a spherical harmonic function

native code and hardware support for the processing steps where it is required and available.

3 Implementation Issues

The WWW visualization application is based on three main parts: a server program, which is written in C++, a network communication protocol and the client program which is written in Java. The client has to be able to build a VRML or Java3D scene graph which allows to switch between levels of resolution for an iso-surface. A major technical contribution of this paper are the methods for building the VRML scene. In this section we briefly describe the network communication protocol and the methods for creating and manipulating the VRML or Java3D scene graph.

3.1 Network Communication

If the client requests an iso-surface for a given iso-value and resolution level, the server computes the iso-surface and sends the resulting triangles to the client. The server sends further information in order to mark the triangles for building the hierarchy of surface levels. The client-server communication is based on Berkeley sockets.

A dataset for one level of the iso-surface consists of an array of vertices for triangles, which share no vertices and an array of indices. The indices mark triangles, that will be replaced in higher levels of the iso-surface (irregular triangles). The vertex and index data have 16 bit precision. A triangle consumes 18 bytes and an index 2 bytes.

3.2 Visualization

Each level of the iso-surface contains two types of triangles: *regular* and *irregular* ones. Regular triangles will not change in the higher levels of the surface. Regular triangles of level m are reused in all levels $n > m$. Irregular triangles of level m are only used at level m . Figs. 1(b,e) show the *regular* triangles for two different levels of an iso-surface while Figs. 1(c,f) show the corresponding irregular triangles. The complete iso-surfaces can be seen in Figs. 1(a,d).

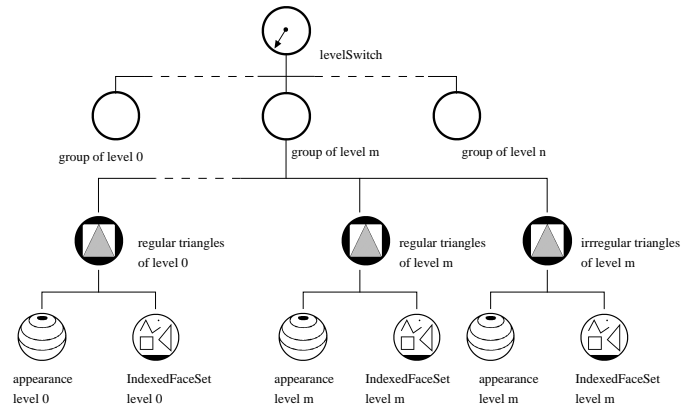


Figure 2: VRML Scene graph for the iso-surface hierarchy

3.2.1 VRML

The VRML scene graph stores all the different levels of detail of the iso-surface (Fig. 2). In order to switch between the different levels a **Switch** node is used as the root node for the scene graph. Each child of this switch node is a **Group** node, that represents one level of the iso-surface. The children of a **Group** node are **Shape** nodes, which consist of an **Appearance** and a **Geometry** node. We use **IndexedFaceSet** nodes as **Geometry** nodes.

In order to build the VRML scene graph using data events, the triangles are stored in an array. The indices, which are created on the client, are collected in a second array. The **IndexedFaceSets** are created immediately after the VRML plug-in is loaded at startup. They build a *base scene* which is actually empty. This scene is filled up with triangles using VRML events.

We use the **eventIn** field "points" of the sub-node **Coordinate** of a **IndexedFaceSet** node to set the vertices of the data. This field is mapped to an instance of the class **EventInMFInt32**.

```
public class EventInMFInt32 extends EventIn {
    public void setValue(int[] value)
        throws IllegalArgumentException;
}
```

The **eventIn** field "set_coordIndex" of the **IndexedFaceSet** node is used to set the indices of the **IndexedFaceSet** node. This field is mapped to an instance of the **EventInMFVec3f** class.

This approach of creating VRML geometry dynamically turned out to be the most efficient, because no string operations and no VRML parser is involved. The vertices and the index data can be handed directly to the geometry nodes of the VRML scene graph.

3.2.2 Java3D

Java3D is a high level scenegraph API for creating and manipulating 3D geometry and for creating the structures used in rendering that geometry [8]. The current Java3D implementations are based on top of OpenGL, thus Java3D rendering is accelerated across the same wide range of systems that are supported by this lower-level API. We used the Java3D Alpha3 Release which runs with JDK1.2 Beta3.

The structure of the Java3D scene graph (Fig. 3) is very similar to the earlier described VRML scene graph. It stores all the different levels of detail of the iso-surface. Each **BranchGroup** child node of a **Switch** node represents one level of the iso-surface. If a set of regular and irregular triangles is received for one level of the iso-surface, two **Shape** nodes are instantiated. The first stores the

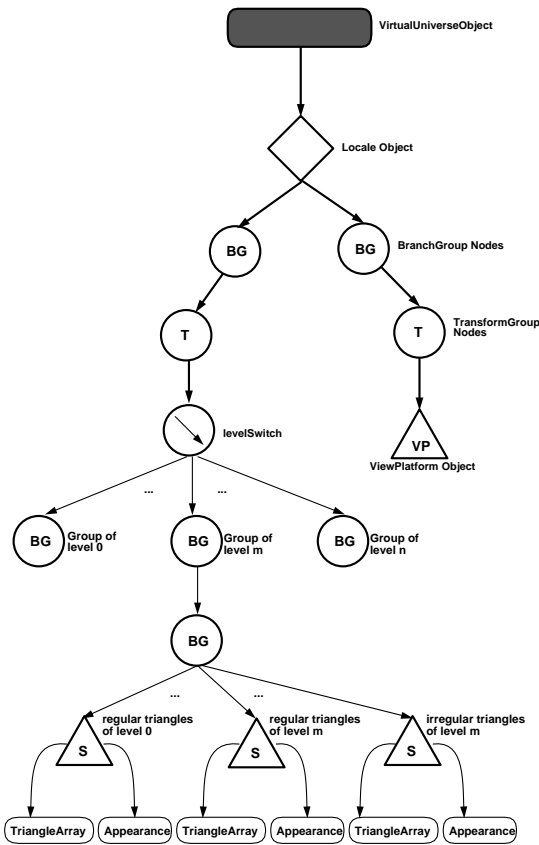


Figure 3: Java3D Scene graph for the iso-surface hierarchy

irregular triangles in a **TriangleArray** object. This **Shape** node is added only to the **BranchGroup** node of the corresponding level of the iso-surface. The second **Shape** node stores the regular triangles in a **TriangleArray** object. It is added to the **BranchGroup** node of the corresponding level and to the **BranchGroup** nodes of all higher levels of the iso-surface.

4 The Java User Interface

The Java user interface is attached to the bottom of the VRML plug-in and provides many parameters for the user to choose from.

A typical working session proceeds as follows. First select the machine running the server process. Due to Java security restrictions, an applet is only able to connect to the server, where it was down-loaded from. If the server differs from the HTTP server, the

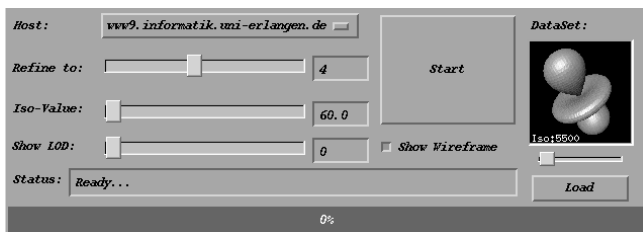
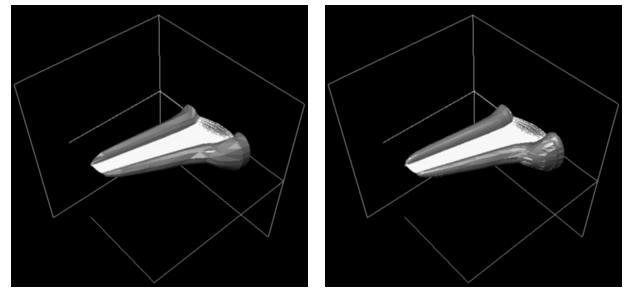


Figure 4: The user interface of the Java applet



(a) level 4 - 1500 triangles
(iso-value 1.1)

(b) level 6 - 16421 triangles
(iso-value 1.1)

Figure 5: Pressure iso-surface around a plane wing with different LOD

access must be explicitly granted and the user will be prompted with a security dialog.

A pool of representative data sets containing pre-processed 3D scalar volumes in the form of adaptively refined unstructured meshes that can be visualized with the progressive iso-surface application has been made available. A data set can be selected by moving a scroll bar and choosing one of the icons obtained from the visualization of the data. The data set is loaded on the server by pressing the **load** button. The maximum level of resolution available on the server side is transferred to the client and all the sliders are adjusted accordingly.

Now, an iso-value and a level of detail can be selected with the **Refine-Level** and the **Iso-value** sliders. Pressing the **Start** button initiates the computation of the surface levels at the server side and the progressive transmission to the client. If a new dataset is selected or if the iso-value is changed, the iso-surface will be recomputed and transferred from level 0 up to the selected end level to the client. On the other hand, if only the level of resolution is changed, the iso-surface will be refined, i.e. the corresponding surface levels will be computed on the server side and only the differences will be transmitted to the client.

The Java applet will automatically switch to a higher level of detail, as soon as a new level is available. The **Show level** slider will be adjusted accordingly. This slider can be used to switch back to any previous level of detail, which allows to visualize the complete surface hierarchy. Because the VRML plug-in and the Java applet run asynchronously, the iso-surface can be visualized, i.e. rotated, translated and zoomed in and out, while higher levels of the surface are being transmitted. A status line at the bottom of the user interface prompts status and error messages.

5 Results

In this section we compare the performance of the transfer and visualization of the surface for the VRML and Java3D implementation.

The client machine was a PC with a 200 MHz Pentium processor, 80MB RAM and a Diamond Fire GL 1000 Pro graphics adapter. We used Internet Explorer 4.01 and CosmoPlayer 2.1 for the tests with our VRML implementation and the JDK 1.2 Beta3 and the Java3D Alpha3 release for the Java3D tests. The server was a SGI O2 with a R5000 processor and 128MB RAM. A 64 KBit ISDN and an 10 Mbit Ethernet connection were used.

Our measurements were based on a numerical simulation of the fluid motion around a wing (Figure 5). The data is discretized on a structured grid of dimension $100 \times 78 \times 34$. The data set was

Transmission and Construction times

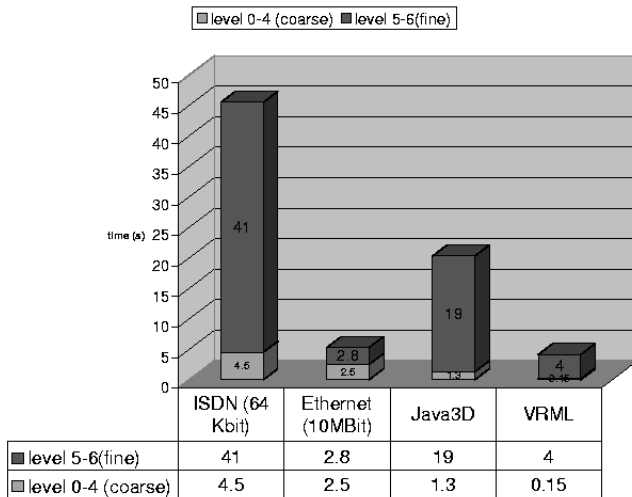


Figure 6: Transmission and construction times for the plane-wing

level	Δ triangles ^a	indices ^b	triangles ^c	data ^d	sum ^e
0	2	2	2	40	40
1	6	6	6	120	160
2	78	78	78	1560	1720
3	354	354	354	7080	8800
4	1500	1076	1500	29152	37952
5	8317	1171	8741	151836	189788
6	8745	1881	16421	157410	347198

^atriangles transmitted in this level

^bindices transmitted in this level

^ctriangles in this level

^ddata transmitted in this level (bytes)

^etotal data transmitted up to this level (bytes)

Table 1: Data transmission for the plane wing iso-surface (Iso: 1.1)

pre-processed and stored in compressed format as described in [3]. The size of the compressed data set is 3.2 MBytes.

The benefits of our approach are obvious. The times given include the network transmission and the construction of the scene graph. The levels 0 to 4 of the iso-surfaces were visualized in less than 5 second (Figure 6). The user is able to get a quick impression of the data and find interesting iso-values.

The performance for the refinement of the iso-surface up to the highest level of detail is obviously much lower (Figure 6 column 2), but still high enough to maintain an acceptable rate of interactivity.

The performance of Java3D for the construction of the scene-graph was much slower as the VRML implementation (Figure 6). This can be explained with the early state of the Java3D API (Alpha 3) and JDK1.2 (Beta3), which are not optimized for performance.

Table 1 shows the number of triangles and indices transmitted for each level of the iso-surface, the number of triangles in the according level, the number of bytes transmitted from one level to the next and the total number of bytes transmitted. Each triangle consists of an array of 18 bytes (3 vertices with 16 bits precision). Each index marks an irregular triangle and consumes 2 bytes.

The number of data transmitted for lower levels of the iso-surface is very small compared to the transmission for higher levels. The

transmission overhead caused by our progressive approach is very limited. The number of bytes transmitted up to the highest level of the plane wing iso-surface is 347 198 bytes for the progressive algorithm and 295 578 (16421 triangles * 18) bytes for a non progressive algorithm.

6 Conclusions

We have presented a distributed application which is suitable for the visualization of large volume datasets on the WWW. At the core of the system is a progressive iso-surface algorithm, which allows the user to interactively adjust the iso-value and the resolution of extracted iso-surfaces. If an interesting iso-value is found, the user can refine the iso-surface depending on the performance of the client machine, hardware support and network load. The overheads produced by our progressive transmission is very low. Because the VRML plug-in and the network transmission run asynchronously, the user can visualize the data, i.e. rotate, translate and zoom in or out, while the surface is being refined.

The use of Java and VRML gives us the advantage of being platform independent while exploiting graphics hardware support if available. However, special attention has to be paid to the efficient communication between both components.

With the techniques introduced in [3] it is possible to progressively transmit not only surfaces, but also entire volumes across the network. At the client side a Java applet could reconstruct the hierarchy of the 3D meshes and extract iso-surfaces at different levels of resolution. The development of such a Java applet which satisfies the performance requirements of interactive visualization is one topic of future research.

References

- [1] Rikk Carey and Gavin Bell. *The Annotated VRML 2.0 Reference Manual*. Addison-Wesley Developer Press, 1997.
- [2] Michael Bailey Cheryl Michaels. VizWiz: A Java Applet for Interactive 3D Scientific Visualization on the Web. In *Proceedings IEEE Visualization '97*, pages 261–267, 1997.
- [3] R. Grosso and Th. Ertl. Progressive Iso-Surface Extraction from Hierarchical 3D Meshes. In *Proceedings EUROGRAPHICS '98*, 1998.
- [4] Hans-Georg Pagendarm Jens Trapp. A Prototype for a WWW-based Visualization Service. In *Proceedings Eurographics '97*, pages 23–30, 1997.
- [5] Rodget Lea, Kouichi Matsuda, and Ken Miyashita. *JAVA for 3D and VRML Worlds*. New Riders Publishing, 1996.
- [6] L. Lippert, M.H. Gross, and C. Kurmann. Compression domain volume rendering for distributed environments. In *Proceedings Eurographics '97*, pages C95–C107, 1997.
- [7] Chris Marrin. Proposal for a VRML 2.0 Information Annex. <http://cosmosoftware.com/developer/moving-worlds/spec/ExternalInterface.html>, 1997.
- [8] SUN Microsystems. Java3D API Specification. <http://www.javasoft.com/products/java-media/3D/forDevelopers/j3dguide/j3dTOC.doc.html>, 1998.