Adaptive Visualization of Distributed 3D Documents Using Image Streaming Techniques

Jobst Löffler[1] and Dieter W. Fellner[2]

[1] Fraunhofer Institute for Media Communication, Schloss Birlinghoven,
D-53754 Sankt Augustin, Germany
jobst.loeffler@imk.fhg.de
[2] Institute of Computer Graphics, Technical University of Braunschweig,
Rebenring 18, D-38106 Braunschweig, Germany
d.fellner@tu-bs.de

Abstract. With the emergence of open information spaces such as digital libraries, advanced techniques for interactive visualization of complex and protected 3D documents are needed. This paper presents a new visualization concept which deals with the problems of complexity and security of digital 3D documents in open information spaces. A dynamic combination of remote and local 3D rendering is used to allow scaling of the information quantity on the client side. The software architecture SCA3D (3D Scalable Scenes Architecture), which provides functionality for adaptive visualization and protection of intellectual property rights, is presented.

1 Introduction

Distributed work involving 3D scenes as part of multimedia documents within networked information systems needs technical support for retrieval, interactive visualization and communication. A networked information system is called an open information space if access to documents and services is not restricted to a predetermined area such as to groups of previously known users and information servers. Homogeneous conditions for work with digital objects and interfaces to other information spaces are characteristics of an open information space which comprises several information servers, document databases, multiple clients and telecommunication networks between them. Figure 1 shows an open information space built up of two information servers (S) with document databases (DB), a number of users (C), who can also use additional local databases, and an interface between the information servers.

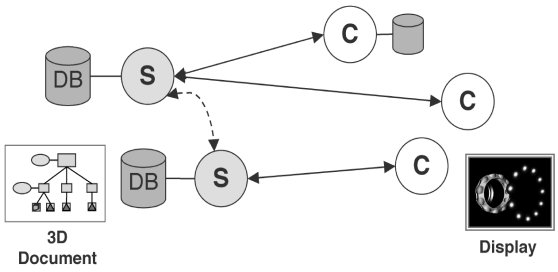

Fig. 1. 3D Visualization in open information spaces.

Documents of a digital library used by a group of students and a tutor working on a research project which involves 3D models (for example an architectural construction project) is a practical example. A digital library allows users to explore already existing 3D documents, generate and add new documents as a result of project work and discuss results. A digital library system should also allow visual cut-and-paste, indexing and extending 3D documents. Protection of the original document and management of document copies are important issues which need to be considered during system development. In order to exploit the full potential of the open information space, both work at the central facility on a LAN using specialized equipment and work at home using standard PCs and low-bandwidth connections should be equally possible. In this paper a visualization architecture is introduced which deals with these issues of complexity and security by using a combined approach of remote and local 3D rendering and an object-oriented document model for 3D scenes. A document-request-broker approach is presented which allows access to objects of distributed 3D documents via proxy objects with a varying level-of-information called object representations.

The remainder of the paper is organized as follows. In section 2 background information is discussed. In section 3 the software architecture SCA3D is presented, while in section 4 its adaptive behavior is evaluated. Finally, in section 5 conclusions are drawn.

2 Visualization of 3D Documents using Hybrid 2D/3D Scenes on Client Side

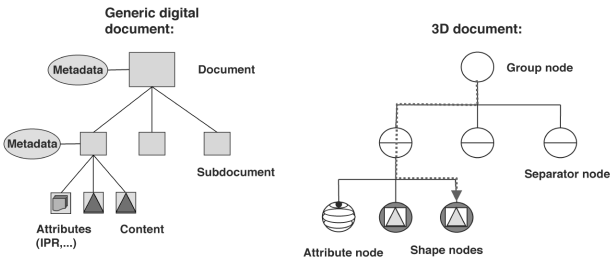

Fig. 2. Generic digital document and 3D document.

Considering 3D scenes as ordinary documents emphasizes that 3D scenes can be handled in the same way and can have similar features, such as cut-and-paste, support of retrieval, indexing or extensibility. They are subject to the same constraints with regard to security or complexity as generalized digital documents in distributed information spaces (see also [6]). 3D scenes can be seen as digital documents because both have a common composition, which comprises a structure plus the actual content (see also [10]). In Figure 2 the assembly of a generic digital document and a 3D document with path information is shown. 3D documents include structure information, which can be mapped onto an internal document model commonly known as the scene graph, and contain geometric objects as content, which are represented as shape nodes in the scene graph. Today's rendering systems, like OpenInventor [14] and Java3D, use this scene graph concept to enable interactive visualization.

The scene graph concept corresponds to an object-oriented document model, as described for example by the W3C DOM (Document Object Model) [2]. This model defines a document and its components in an object-oriented way: objects are distinguishable, they are arranged following a certain hierarchy (scene graph structure), object data and state are encapsulated and only accessible via a well-defined interface. Using the OpenInventor scene graph model, interactive visualization of 3D scenes is performed by applying actions, such as *render* or *pick*, to the scene graph and thereby changing the state of the objects. Applying a render action generates image data showing the results of a lighting simulation. Other methods of the object interface are used to construct a 3D scene by generating new objects, copying object data or removing objects from the scene graph. Object manipulation can be done using methods which change the internal state of the object, for example by calling the method *setValue(float r)* of the radius field of a sphere. Meta-data like node names or information concerning security issues, typically access rights or copyright data, can be integrated into a 3D document by filling the data fields of attribute nodes.

2.1 Combining Remote and Local 3D Visualization

Visualization of distributed 3D documents can be done by copying the data to the client's computer and using a local rendering pipeline or alternatively by rendering the data on a server machine and distributing the rendered image data. Local 3D visualization, where the entire visualization pipeline is performed on the same machine, takes advantage of available hardware acceleration for geometry processing and display. On the other hand in open information spaces there are important reasons not to distribute all geometry information without control, because security, bandwidth and performance requirements must be taken into account. Examples using local 3D visualization with data replication are presented in [1], [5] and [8]. Many of these systems are used to support cooperation with distributed 3D visualization. Alternatively, remote 3D visualization as a service can be used to let users interactively visualize 3D scenes. On the client side of a network connection a visualization application reads the images transmitted from the server and displays them using an interactive image viewer. A user interacts with a 3D scene by sending back the performed actions over a reliable connection to the server where they are applied. The advantages of such a system are that it can deal with complexity and security requirements because the original data can remain on the server. But using remote visualization also brings some disadvantages: image generation on the server side and the transmission of images with high resolution cause a delay in the interaction loop which can disturb the interactive impression. The approach of remote 3D visualization and control of distributed 3D worlds over image streams is discussed in [7], [3] and [9]. This work indicates that interactive frame rates for remote visualization are hard to reach with today's systems.

A solution that combines the advantages of both approaches and minimizes the disadvantages would combine remote and local 3D visualization and synchronize both processes in a uniting software architecture. There are still relatively few references that discuss a combination of remote and local 3D visualization One

example [4] introduces a concept and system for distributed 3D visualization of volume data for medical applications. A solution that minimizes delay in remotely controlled 3D environments by selective pixel transmission and mapping onto a basis of local geometry is presented in [11].

2.2 Object Representations with Varying Level-of-Information

Geometry-based content is commonly described either as an exact representation through parametric curves and surfaces or as polygonal 3D meshes such as triangle meshes. But working with distributed and complex 3D scenes does not imply that a user needs or wants the full geometric information of 3D objects from the beginning.

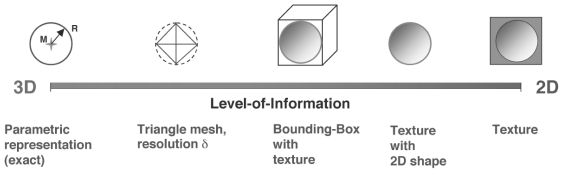

Fig. 3. Object with increasing level-of-information.

A client who just wants to get an overview of a virtual world or to navigate in it can work with representations that have a low level of complexity, like images or texture plus bounding box information. A full 3D representation needs to be transmitted in 3D mesh or parametric model form only when a request for high-level local interaction with objects is made. Figure 3 illustrates the idea of representing 3D objects with increasing level-of-information. The Level-of-Information approach presented here is based on dynamic control of the distribution process of object representations which best fit the connection and local performance conditions of a client. The following object representations for 3D objects are supported by the SCA3D architecture: *parametric models*, *polygonal mesh models with adaptable resolution*, *bounding box plus texture*, *texture plus 2D shape* and *plane texture*. The level-of-information is connected to the accessible object information of a representation.

In Figure 4, object representations are plotted into a bandwidth-vs.-performance diagram to illustrate how the configuration of a client scene can be adjusted to the connection bandwidth and the local CPU-Power of a client. Four different scenarios are classified with regard to bandwidth and power of a client: **(A)** a personal computer in a local area network (PC in a LAN), **(B)** a notebook computer with ISDN connection, **(C)** a personal digital assistant connected to a LAN (PDA in a LAN) and **(D)** a personal digital assistant with mobile connection

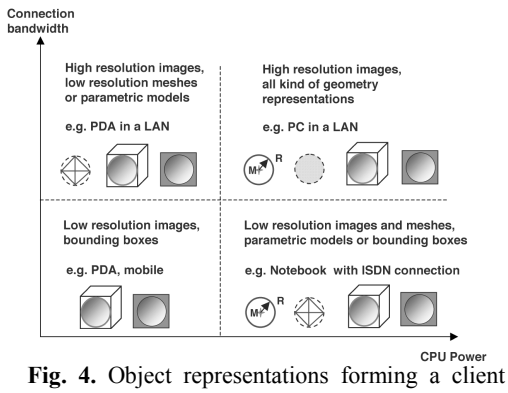

Fig. 4. Object representations forming a client scene under different conditions.

(PDA mobile). In case **(A)** representing a client with high bandwidth and very good CPU-Power all object representations can be used to build a client scene. For instance, such a client can use remote rendering in connection with high resolution images for navigation in a document collection and can be allowed the necessary permission to obtain the original data and then download all objects as full resolution geometry. A client who is represented by case **(D)** cannot download large amounts of geometry data over a small bandwidth connection and would not be able to process the geometry data locally. For such a client working on a 3D document, object representations like low resolution images and bounding boxes would be a good choice. In case **(B)** and **(C)** other representations such as meshes with varying resolution are suitable to fit the described conditions.

3 A Distributed Software Architecture for Adaptive 3D Visualization

The Scalable Scenes concept combines remote and local 3D visualization to maximize the advantages and minimize the disadvantages of the two approaches. Integrating both approaches into a common concept allows the realization of a system that enables interactive frame rates on the client side for local objects and security and complexity management for remote objects. Synchronization of the visualization pipeline on the server side with the client's visualization pipeline allows the user to navigate in remote 3D documents and to interact with objects of the same document located as copies on his computer. The concept distinguishes a remote navigation space on the server side and a local interaction space on the client side. A level-of-information slider provided by the client application controls which object representations can be transmitted to a client on a selection event. The positions of the LoI-slider range between 2D image objects and 3D original objects. Certain positions can be locked to prevent a user from transmitting protected objects. Locking is indicated by a hatched area of the slider of client 1 in Figure 5 which shows the concept of Scalable Scenes in an overview. Three states of an object are distinguished. An *inactive server-side object* is only present in the navigation space and can

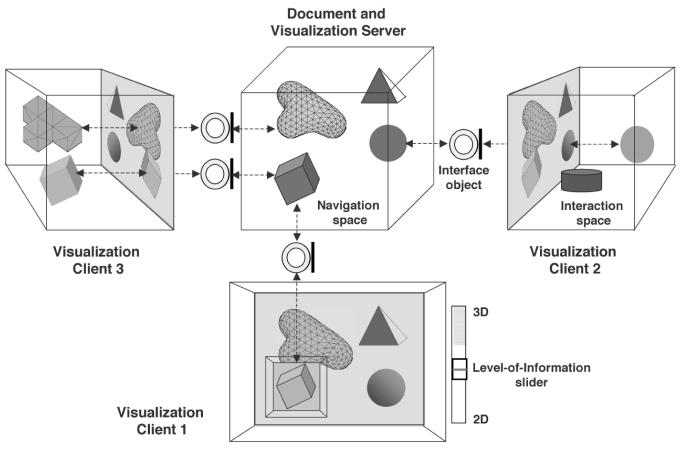

Fig. 5. The combined concept of Scalable Scenes.

therefore be visualized remotely. An object which is selected by a user is in the state of an *active object*. An active object is present in the navigation space and represented on the client side by a suitable object representation which is connected to the server-side object via an interface object. The interface object provides a method of interface to the user which is accessible over the local object representation. Finally, an object can be in the state of an *inactive client-side object* indicating that this object was transmitted as original from server to client.

The SCA3D architecture is built to support multi-client collaboration. Every client with a connection to the SCA3D server is provided with a uniquely registered ID, so that its remotely callable methods are individually addressed. The collaboration model of SCA3D uses unique client IDs to maintain individual object references for every client. A multi-client image server is used to distribute image streams over socket connections to any number of clients. The software architecture comprises four layers (see Figure 6): a WWW layer, a visualization layer using the OpenInventor API, a document control layer using a CORBA implementation, and finally, an image layer used for image coding, transmission and decoding. The document control layer and the image layer are discussed in more detail in the following sections.

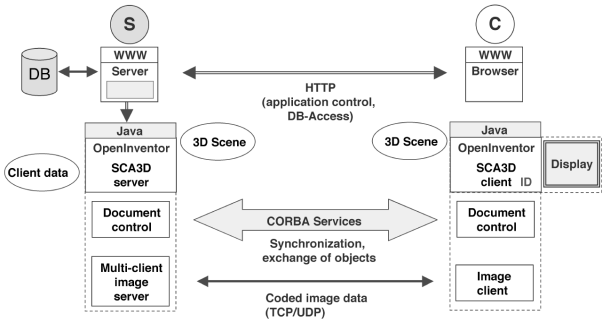

Fig. 6. Layer model of the SCA3D architecture.

3.1 A Document-Request-Broker Approach

Access to document objects in the distributed system is managed using a technique which is similar to an object-oriented middleware concept based on remote-procedure-calls (RPC) [13]. Generally, the standard architecture CORBA is used to develop distributed applications where remote objects are accessed via local proxy objects. On the server, method implementations provided by so called skeleton objects are used to apply the RPC. A similar approach, which we term *document-request-broker* (DRB), is integrated into the Scalable Scenes concept. The document-request-broker uses interface objects on the server side and local object representations on the client side

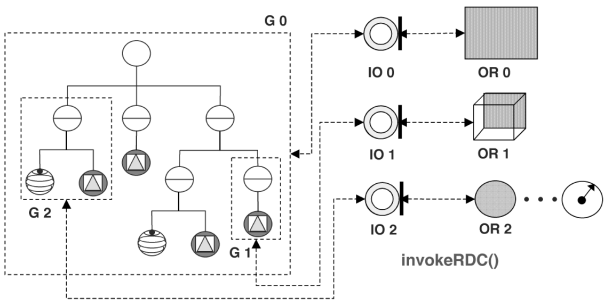

Fig. 7. Access to documents using proxy objects.

to support visual access to remote 3D documents. The object-oriented document model of a scene graph supports this approach very well.

The DRB-concept is shown in Figure 7. The document itself is equivalent to the group *G0* of the document-request-broker. It is represented on the client side by the object representation *OR0*, which is the remotely rendered image of the 3D scene. The original document and the client's object representation are connected by an interface object *IO0*. The client can now apply actions on the document by invoking methods provided by the object representation, such as *pickObject* or *getCopyrightInformation*. These *remote-document-calls* (RDC) are applied by the document-request-broker to the original document using the interface object's methods. In the next steps, further objects can be activated and interface objects together with object representations for subgroups of the document are generated and connected. All object representations mentioned above can be used to work with server-side document objects and therefore the information quantity can be scaled dynamically. For these purposes transformations between graphical representations of objects are performed on server side, e.g. by surface sampling of parametric surfaces and tessellation using an additional complexity node in the scene graph or by bounding box calculation. The position of the client-side level-of-information slider is used to control these transformation processes.

3.2 Organization of the Image Streaming Layer

Images are distributed in the software architecture to support object representations which use remotely rendered images. The basic configuration of the client scene requires that an image representation of the whole 3D document is generated on the server side with a certain server frame rate (**SFR**), transmitted to the client and mapped onto the client's projection plane with a certain texture mapping rate (**TMR**). To adjust the rate of an image stream to a client's condition, a mechanism for rate control is implemented in the image layer. The rate control technique applied here is also used in image streaming environments to prevent network congestion (see [12]).

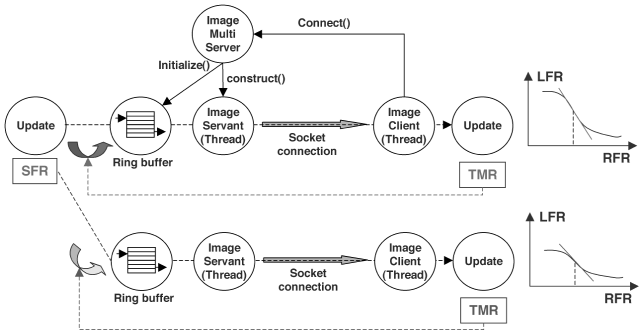

Fig. 8. The SCA3D image layer for multi-client support.

The SCA3D image layer uses server-side image buffering with a feedback from receiver and sender to control the filling rate of the image ring buffer. The rate with which the ring buffer is filled depends on the ratio of the fixed server frame rate and

the individual texture mapping rate of each client. An image servant thread on the server side reads the image data from the ring buffer and writes them to a connected image client thread over a socket connection previously provided by an image multi server. On the client side, the images are mapped onto the projection plane with the adjusted texture mapping rate. The setup of the image layer with its Java classes is shown in Figure 8.

To find the appropriate ratio of server frame rate and texture mapping rate for each client, a performance measurement step must be performed. For this purpose the local frame rate (**LFR**) and remote frame rate (**RFR**) are measured and their dependency is evaluated. In Figure 8 the characteristic LFR-vs.-RFR diagram for both clients connected to the server is shown on the right side. To produce the dependency curve, the remote image frame rate was stepwise increased from a minimum value to a maximum value by changing server frame rate and texture mapping rate accordingly.

4 Adaptive Behavior by Control of Frame Rates and Local Geometry Load

The software architecture SCA3D implements an adaptive behavior with regard to local and remote frame rates. When a client connects to a server, the optimal ratio between local and remote frame rates will be adjusted in order to minimize delays for local and remote actions using a performance evaluation process which is performed at fixed intervals. The remote frame rate is influenced by three factors: the delay caused by image generation, by data transmission over the network and by texture mapping on the client side. The reciprocal value of the overall delay is used here to approximate the RFR value (see Equation 1(left)). The local frame rate is estimated as the reciprocal of the local delay of an action (see Equation 1(right)). It depends mainly on the amount of geometry information to be rendered, on the adjusted texture mapping rate and on the resolution of texture images.

$$RFR \cong \frac{1}{Tgen + Tnet + Tmap}, \qquad LFR \cong \frac{1}{Tlocal} \tag{1}$$

In order to support work on a distributed 3D document it is necessary to find the optimal ratio between local and remote frame rates. An optimal ratio of local and remote frame rates enables fluent local work on 3D objects and a remote visual feedback with a small delay reflecting the state of the server-side document. In the upper right corner of Figure 9 (a) this is demonstrated by showing a 3D object that is moved locally together with the remote visual feedback reflecting the state of the original document. The difference of local and remote delay results in an impression that the remote object is connected to the local object representation with an elastic band.

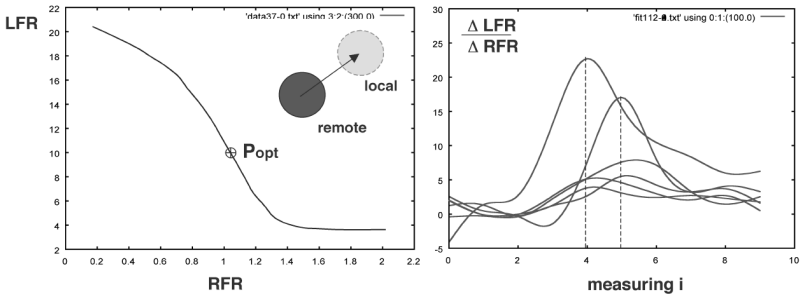

Fig. 9. Estimation of the optimal ratio: (a) LFR vs. RFR (left), (b) slope of LFR-vs.-RFR curve for varying geometry (right).

Figure 9 (a) shows the experimental results of a performance test for one client as an LFR-vs.-RFR curve with the optimal ratio *Popt*. This point is estimated by evaluating the slope of the curve. The characteristic curve of a client application, which is busy with texture mapping and with local rendering, shows a point of maximum slope indicating the optimal ratio of local and remote frame rates. This point is used to adjust the local texture mapping rate and the server-side push rate of the image ring buffer. Diagram 9(b) shows the changing slopes resulting from performance evaluation of the same client for a varying geometry load. The client's behavior changes under varying conditions and performance evaluation must be repeated from time to time.

To minimize the overall delay, the optimal ratio of local and remote frame rates of every client has to be adjusted individually using the rate control mechanism described above. The server frame rate, which is proportional to the maximum texture mapping rate demanded in the environment, and the texture mapping rate of a client are used to control the push rate of the image buffer. The delay factor for filling the image buffer is estimated as the reciprocal of the ratio $\omega i = SFR/TMRi$ between the server frame rate and the individual texture mapping rate. By multiplying the server frame rate with the delay factor, image streams at optimal frame rate can be delivered to every client.

In order to compare clients, a performance metric is needed which reflects the differences in processor power and bandwidth. The metric is then used to control the level-of-information slider and to provide a client with object representations best fitting the individual local conditions. Figure 10 shows the comparison of two clients with help of the LFR-vs.-RFR diagram. The performance of client 1 is better than that of client 2, which is expressed by the difference of the length *L* of the

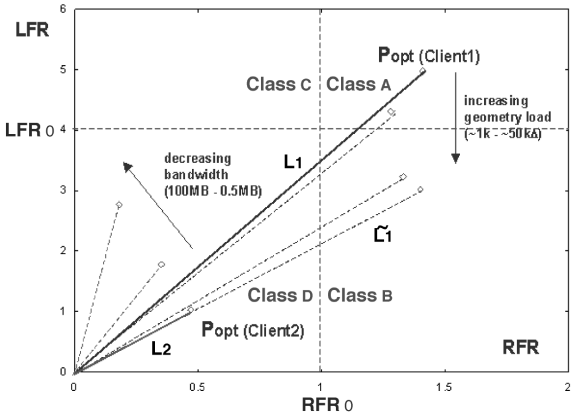

Fig. 10. Performance comparison of two clients.

vector from the origin to *Popt*. Evaluation of the vector length of both clients in LFR-RFR space as $L = \sqrt{LFR^2 + RFR^2}$ shows that the performance of client 1 is approximately five times the performance of client 2. Both clients were connected to the server over a LAN (*10 Mbit/s*). Client 1 was a personal computer with a powerful processor (*L1=5.2*) whereas client 2 was an older workstation with less CPU-power (*L2=1.1*).

To show how this metric reflects the performance changes under varying conditions, the geometry load was increased for client 1. The results are presented in Figure 10. The remote frame rate remained nearly the same under the new conditions whereas the local frame rate was decreased in indirectly proportion to the number of triangles in the client scene. The performance value changed from *L1=5.2* to $\widetilde{L1}=3.4$ as a response to a change of the triangle number from *1000* triangles to *50000* triangles. The distance ΔL between the clients changed from $\Delta L=4.1$ to $\widetilde{\Delta L}=2.3$. Also the behavior of client 2 under changing bandwidth conditions between server and client is shown in Figure 10. The bandwidth was decreased stepwise during the tests from *100 Mbit/s* to *0.5 Mbit/s* and the optimal ratio of the frame rates was re-evaluated after each step.

Using this performance metric, clients represented by the classification scheme shown in Figure 4 can now work together on a 3D document because the system adjusts the level-of-information on the client side. To achieve this goal performance, classes are defined using threshold values (*LFR0* and *RFR0*) for local and remote frame rates as shown in Figure 10. Correlation of performance classes with the geometry load caused by accessible object representations allows control of the level-of-information on the client side.

5 Conclusions

The starting point of this paper was distributed work on 3D documents in open information spaces. The software architecture SCA3D presented here is a solution for interactive visualization of distributed 3D documents with integrated techniques for complexity and security management. The basic concept of the architecture combines local with remote 3D rendering and uses hybrid 2D/3D client scenes. An object-oriented document model together with a document-request-broker approach for remote-document-calls is integrated into the concept to enable scaling of the information quantity on the client side. The adaptive behavior of the architecture was described by introducing a technique for rate control known from image streaming systems and a performance metric which allows a comparison of clients working together on a 3D document.

Generalization of the concept to support distributed work on documents which also contain image, video, audio and text data will be explored in future stages. The combination of standards like XML, W3C DOM (document object model) and MPEG-7 together with a document-request-broker approach is seen as very promising to support retrieval and visualization of generic documents in open information spaces.

References

1. Carlsson, C., Hagsand, O.: DIVE- A Platform for Multi-User Virtual Environments. Computer & Graphics, Vol. 7(6) (1993) 663-669
2. DOM (W3C): Document Object Model (DOM) Core Specification. W3C (2000)
3. Engel, K., Sommer, O. and Ertl, T.: A Framework for Interactive Hardware Accelerated Remote 3D-Visualization. EG/IEEE VisSym '00 (2000), 167-177
4. Engel, K., Hastreiter, P., Tomandl, B., Eberhardt, K. and Ertl, T.: Combining Local and Remote Visualization Techniques for Interactive Volume Rendering in Medical Applications. IEEE Visualization '00 (2000), 6-10
5. Fellner, D.W., Hopp, A.: MRT-VR Multi-User Virtual Environment. AAA'97 (1997)
6. Fellner, D.W., Havemann, S., Müller, G.: Modeling of and Navigation in complex 3D Documents. Computer & Graphics, Vol.22(6) (1998), 647-653
7. Haulsen, I., Jung, T., Tuchtenhagen, D: Remote Control of Virtual Environments Using Image Streams. Second IMA Conference on Image Processing (1998)
8. Hesina, G. et al.: Distributed Open Inventor: A Practical Approach to Distributed 3D Graphics, ACM Virtual Reality Software & Technology '99 (1999), 74-81
9. Löffler, J.: Object-based Image Coding for Cooperative 3D Visualization. International Conference on Computer Graphics WSCG (2000), 197-203
10. Löffler, J.: Content-based Retrieval of 3D Models in Distributed Web Databases by Visual Shape Information. IEEE Information Visualization IV'00 (2000), 82-87
11. Mann, Y., Cohen-Or, D.: Selective Pixel Transmission for Navigating in Remote Virtual Environments. Eurographics Conference EG'97, Vol.16(3) (1997)
12. Riley, M., Richardson, I.: Digital video communications. Artech House, ISBN 0-89006-890-9 (1997)
13. Siegel, J.: CORBA: fundamentals and programming. Wiley, ISBN 0-471-12148-7 (1996)
14. Wernecke, J.: The Inventor Mentor - Programming Object-Oriented 3D Graphics with Open Inventor. Addisson Wesley, ISBN 0-201-62495-8 (1998)