

# Accessing Multi-user Virtual Worlds over IP

R. Bernardini and G. M. Cortelazzo

Dipartimento di Elettronica ed Informatica  
Università di Padova  
Via Gradenigo 6A, 35100 Padova  
e-mail: {bernardi, corte}@dei.unipd.it

## Abstract

The construction of photo-realistic virtual worlds is at reach of current computer graphics. Unfortunately, the philosophy currently adopted for the diffusion of virtual worlds over the Internet, which calls for downloading at client side the 3D virtual world description, does not allow several user to share the same virtual world. Recently, in order to solve this issue, the split-browser approach was proposed. This work suggests a compression scheme for the stream of images generated in the split-browser approach. The proposed scheme decomposes the virtual world as a union of objects which are in turn approximated as polyhedrons. The image of each face of the polyhedron is compressed with a MPEG-like approach by predicting it by means of a projective transformation.

## 1 Introduction

Three-dimensional virtual environments are often described in VRML, a language tailored for the Internet exchange of 3D virtual worlds. Unfortunately, the current way of accessing VRML described virtual worlds (i.e. downloading the VRML file to the user's computer, then render it) is plagued by several problems such as long downloading times (a good quality VRML file can be as large as 100 MB), poor interaction (due to the huge complexity of VRML files which would require very powerful computers in order to have a smooth navigation) and loss of copyright control (producing a good quality VRML file can require several months of work, but if the user can download it, the author cannot control its diffusion).

In order to solve such problems, in [1] the alternative *split-browser* approach for virtual worlds browsing was proposed. The split-browser ap-

proach is based on the idea of performing the rendering at the server's side and sending the resulting images to the client, transforming the interaction with the VRML file in a problem of image transmission.

It is clear that a major problem that must be solved in order to make the split-browser approach effective, is the compression of the generated images. Although such a problem could be solved by means of off-the-shelf solutions such as JPEG or MPEG, the images transmitted with the split-browser approach have the peculiarity of being artificially generated by the server and this suggests that a specifically tailored compression technique could result in a greater efficiency.

This paper presents a compression scheme that we are currently developing in order to allow efficient access to virtual worlds over the Internet. In the proposed compression scheme the sequence of rendered views is compressed by splitting it in smaller images, which in turn are compressed by means of an MPEG-like approach, but using "generalized motion vectors" representing projective transformations.

In the literature it is possible to find several approaches similar to the one proposed in this work. For example, in [2, 3] the server sends a stream of images to the client which makes its requests by means of CORBA. Tsoi and Gröller [4] propose an approach which dynamically partitions the computational duties between client and server, according to network load and client capabilities. An approach which allows remote access to radiological data to multiple users is described in [5]. In [5] sections of the radiological data are converted into VRML models which are sent to the client which visualizes them by means of a VRML renderer. A similar approach is presented in [6]. Finally, the maybe most similar approach to the tech-

nique presented in this paper is given in [7]. The main difference is in the compression technique used in [7]. Finally, it is worth remembering also that there are also some commercial products based on the idea of moving the rendering phase to the server (e.g., the SGI's OpenGL Vizserver).

This paper has 5 sections. Section 2 briefly recalls the split-browser approach proposed in [1], Section 3 describes the compression scheme that we are developing, Section 4 briefly describes other issues related to the efficient access to remote 3D virtual environments. Section 5 gives the conclusions.

## 2 A Split-browser Structure

The recent development of 3D imaging technologies, e.g. the availability of range cameras and semiautomatic 3D modeling tools, makes feasible the construction of 3D models of real objects and suggests their use for interactive applications such as virtual visits.

The only currently available way to view a 3D model, is by locally downloading a file (e.g. in VRML format) and having the client rendering it. This approach, however, has several drawbacks

- Descriptions of 3D models can be quite large. As an example, the full, uncompressed model of "Madonna con bambino" by Giovanni Pisano (Fig. 1) is approximately 100 Mbytes.
- Rendering with good quality a VRML file requires a lot of memory and computational power, often not available to home users. Interacting with virtual objects and environments can be quite frustrating if the rendering is too slow.
- 3D model construction is a labor-intensive task and it is reasonable to assume that one may want to keep the copyright of the 3D model. If the electronic description of the model can be downloaded via Internet, copyright control becomes very challenging.

In order to solve these problems, in [1] the split-browser approach was proposed. In order to make this paper self-contained, we will briefly recall the main ideas behind the split-browser concept. As a first step, let us analyze how a VRML browser works. In Fig. 2a one can see the internal structure of a generic VRML browser: the end user interacts with a Graphical User Interface (GUI) which, by means of events, controls a virtual world and a



Figure 1: 3D View of the complete model of "Madonna con Bambino" by Giovanni Pisano (Arena Chapel, Padova).

graphical engine (GE) whose task is to produce 2D views of the virtual world.

The solution proposed in [1] splits the browser in two: the part with the GE is moved to the server and the part with the GUI remains at the client (see Fig. 2b). The internal events generated by the GUI in Fig. 2a are now transmitted to the server along the network. At the server's side the GE responds by sending back to the client the updated views as images. This approach, which essentially turns the interactive inspection of a 3D model into an image transmission task, has the following advantages

- The amount of data transmitted from the server to the client is much smaller than sending the full 3D model.
- Visualization at the client side does not require strong computational capabilities anymore.
- The copyright control of the 3D data is preserved, since the end user does not receive the whole model, but only 2D views of it.

Observe also that the split-browser idea allows several users to interact with the same virtual environment, making possible applications like network video game and sharing of scientific data. This situation, with a virtual world which is accessed by several users, is shown in Fig. 3. Corresponding to each user, there is a "local" application running at the user's computer and a "remote" application running at the server. Each user interacts with a local GUI which generates some events that are con-

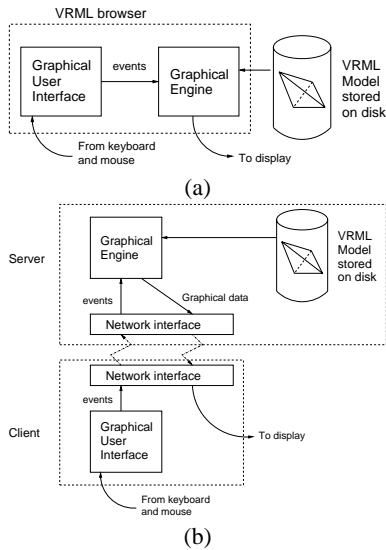


Figure 2: The Split-browser approach: (a) The internal structure of a generic VRML browser. (b) In the split-browser approach the graphical engine remotely runs at the server and sends the computed views to the client.

verted into packets and transmitted to the server via the network.

Although the proposed solution is conceptually very simple, several questions must be answered in order to make it suitable for applications. A fundamental issue is about compression schemes suited to make more efficient the downloading of the updated views. Minor (but important) issues concern the transport protocol to be used for sending the compressed images and how to avoid problems due to packet loss when sending the packets over IP. In the following sections we are going to address such questions.

It is worth to make clear that the “events” sent along the network in Fig. 2b are *not* necessarily the same events generated by the GUI in Fig. 2a. More precisely, while events in Fig. 2a are usually of “primitive” type such as “Mouse moved in...”, “Left button released” and so on, the events transmitted in Fig. 2b are “high level” events such as “Move the viewpoint to...”, “Move the pawn from b2 to c3” (in a chess game).

Sending higher level events allows both to save bandwidth and making the server independent from

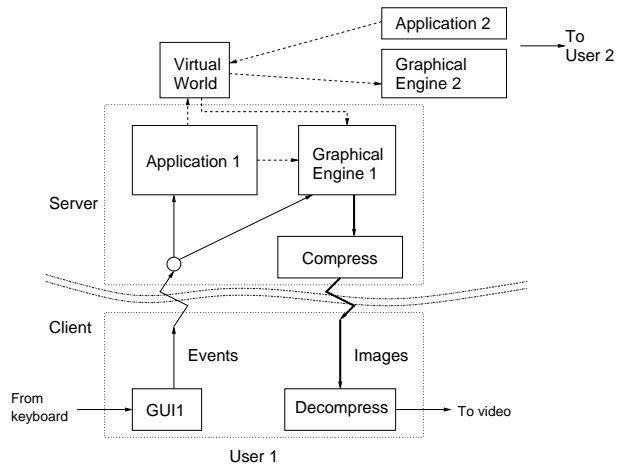


Figure 3: Example of an application with a virtual world accessed by several users.

the GUI which can be chosen by the user according to his/her own taste. For example, in a chess game the user can move the pieces by means of an alphanumeric interface, by using the mouse or maybe by using a data-glove and a stereo display; to the server this does not matter as long as it receives commands like “Move piece ... in ...”.

As a final remark, it is worth observing that the main drawback of the split-browser approach is that most of the computational load is moved to the server which must be a very powerful computer in order to serve several users at time. One could wonder if it can be more economical to buy several last-generation PCs with good graphical card instead of a specialized computer.

The answer actually depends on the specific application. If one wants to use the split-browser approach in a local network, maybe the “several PCs” solution is more convenient, but there are some applications where the “several PCs” solution is not applicable. For example, a video game producer could make a multi-user action game where each user connects to the server by means of a PC. It is clear that the producer has no control on the performance of the user’s PC. The only way to be granted for suitable performances is to declare that the game needs a PC with suitable characteristics (3D graphic card, minimum amount of memory and so on). Unfortunately, this would reduce the number of possible customers.

In this case the split-browser approach has several advantages. A first one is that the software house does not need to produce a different version for every architecture/operating system, since the only part which is architecture dependent is the client (making the video game in Java is not suitable, due to the high performances required to action video games). Another important advantage of a split-browser game is that the users do not buy *copies* of the software (which can be duplicated and given to friends), but *accesses* which cannot be duplicated, since it is straightforward for the server to check that a given user is not connected more than once. The economical advantage due to the loss of piracy could make convenient for the video game producer the acquisition of a powerful graphical server.

### 3 A Compression Scheme

Although off-the-shelf solutions (e.g., JPEG or MPEG) could be used in order to compress the rendered view, it is well worth developing a compression scheme tailored to this particular application in order to exploit the fact that the images sent to the client are artificially generated.

#### 3.1 Images of Planar and Quasi-planar Objects

Consider the situation depicted in Fig. 4 where the virtual world contains just one planar object  $O(\mathbf{x})$  (a picture, for example),  $V_1$  is the position of the user's "virtual camera" and  $L_1(\mathbf{x}) : \mathbb{R}^2 \rightarrow \mathbb{R}$  is the image shown on the user's screen. Suppose now the user moves the view point to  $V_2$  and let  $L_2(\mathbf{x})$  be the corresponding view. Our goal is to compress  $L_2(\mathbf{x})$  by (eventually) exploiting the fact that  $L_1$  and  $L_2$  are different views of the same object.

It is well-known that if  $O$  is planar, there is a simple relationship between  $L_1$  and  $L_2$ , more precisely,

$$L_2(\mathbf{x}) = L_1 \left( \frac{\mathbf{A}\mathbf{x} + \mathbf{b}}{\mathbf{c}^t\mathbf{x} + d} \right) \quad (1)$$

where  $\mathbf{A}$  is a  $2 \times 2$  real matrix,  $\mathbf{b}, \mathbf{c} \in \mathbb{R}^2$  and  $d \in \mathbb{R}$ . Transformations of type (1) are called *projective transformations*. Parameters  $\mathbf{A}, \mathbf{b}, \mathbf{c}$  and  $d$  can be easily be computed by knowing the positions of the planar object  $O$  and the user's virtual camera.

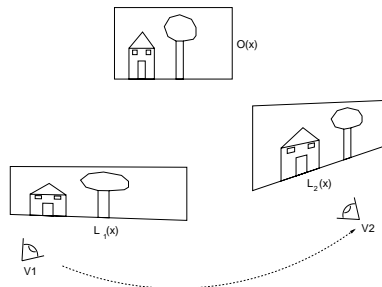


Figure 4: Two views  $L_1$  and  $L_2$  of the same planar object are related by a projective transformation.

If the object is not planar, but it can be fairly well approximated by a planar object, (i.e., it is a *quasi-planar* object)<sup>1</sup>, it is reasonable to expect that (1) "almost" holds, in the sense that the power of error image

$$E(\mathbf{x}) \triangleq L_2(\mathbf{x}) - L_1 \left( \frac{\mathbf{A}\mathbf{x} + \mathbf{b}}{\mathbf{c}^t\mathbf{x} + d} \right) \quad (2)$$

is small. This suggests to transmit  $L_2$  by sending the four parameters  $\mathbf{A}, \mathbf{b}, \mathbf{c}$  and  $d$  and the error image  $E$  compressed by means of a lossless or lossy technique. This approach resembles the approach used in MPEG, but with motion vectors replaced by the "generalized motion vector" represented by  $\mathbf{A}, \mathbf{b}, \mathbf{c}$  and  $d$ .

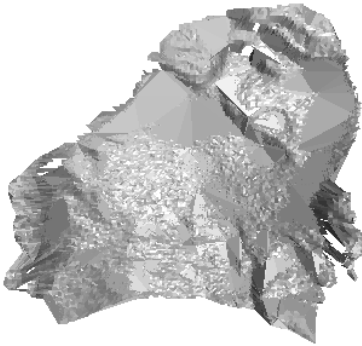
#### 3.2 Images of general objects

It is clear that the hypothesis of planar (or quasi-planar) object is a very strong one since most objects cannot be considered quasi-planar. In such cases, the object is first roughly approximated by means of a polyhedra whose faces correspond to quasi-planar zones of the original object. Successively, the region relative to each face of the polyhedron is compressed as a quasi-planar object. The rough approximation of the object can be easily obtained by simplifying the original model with well-known algorithms.

More precisely, the compression algorithm we suggest is the following

1. Determine the set  $\mathcal{F}$  of the faces visible in both the old and the new view

<sup>1</sup>Note that the condition of being quasi-planar it depends both on the object and on the viewing distance



(a)



(b)

Figure 5: (a) Close-up of the 3D model of *Madonna con Bambino*. (b) Example of a rough approximation of (a) by means of a planar faces.

2. For each face  $F \in \mathcal{F}$  found at the previous step
  - (a) Determine the projection  $P_F$  of  $F$  on the old image
  - (b) Determine the parameters  $\mathbf{A}_F$ ,  $\mathbf{b}_F$ ,  $\mathbf{c}_F$ ,  $d_F$  of the corresponding projective transformation
  - (c) Compute the distorted version

$$M_F(\mathbf{x}) = L_1 \left( \frac{\mathbf{A}_F \mathbf{x} + \mathbf{b}_F}{\mathbf{c}_F^t \mathbf{x} + d_F} \right) \times \chi_{P_F} \left( \frac{\mathbf{A}_F \mathbf{x} + \mathbf{b}_F}{\mathbf{c}_F^t \mathbf{x} + d_F} \right) \quad (3)$$

where  $\chi_{P_F}(\mathbf{x}) = 1$  if  $\mathbf{x} \in P_F$  and  $\chi_{P_F}(\mathbf{x}) = 0$  otherwise.

3. Compute the predicted image

$$\hat{L}_2(\mathbf{x}) = \sum_{F \in \mathcal{F}} M_F(\mathbf{x}) \quad (4)$$

4. Compute the prediction error

$$E(\mathbf{x}) = L_2(\mathbf{x}) - \hat{L}_2(\mathbf{x}) \quad (5)$$

5. For each face  $F \in \mathcal{F}$  transmit
  - The parameters  $\mathbf{A}_F$ ,  $\mathbf{b}_F$ ,  $\mathbf{c}_F$ ,  $d_F$
  - The vertices of  $P_F$
6. Transmit the prediction error  $E(\mathbf{x})$ , suitably compressed.

Observe that the overall scheme is lossy or lossless, depending on the compression method used for  $E$ . A schematic picture of the proposed compression scheme can be found in Fig. 6.

It is worth noting in Fig. 6 the presence of a cache. The motivation for its introduction is that while browsing a virtual world is very common to return in an already visited positions. It is clear that in such a case the rendered image will be equal to the image already sent to the client. In order to exploit this, it is convenient to keep a cache where the more recently generated views are saved.

Note from Fig. 6 that the server uses the cached images also as reference images to be used in the compression scheme. This allows the server to use as reference any previously generated image and not necessarily the current one.

### 3.3 The Sprite Model

Until now we made the hypothesis that the virtual world contains only one object. Although this

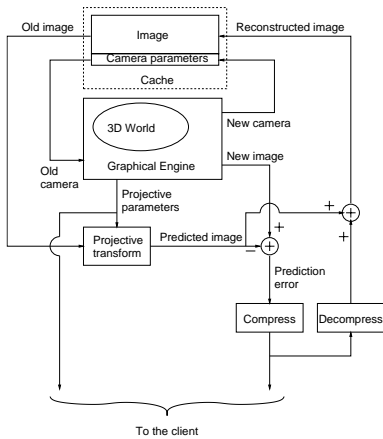


Figure 6: Proposed compression scheme.

could be true in some applications (for example in a close-up of a statue during a virtual visit or in an e-commerce context), it is clear that several other applications (e.g. video games) will have virtual worlds populated by several objects.

The compression scheme presented in the previous sections can be nevertheless still used by exploiting the *sprite model* introduced in [1]. Within the *sprite model*, each image sent to the client is considered as a set of several independent layered images (with non-rectangular support) called *sprites*. Each *sprite* is the rendered view of a single object or a part of an object.

A *sprite* is completely described by its support and by the RGB values associated to points of its support. The support can be described as a polygon (for simple supports, like the support of the *sprite* relative to a check-board) or by means of a transparency component, also known as *alpha-channel*.

It is worth observing that the task of composing several *sprites* into a single image is well within the capabilities of current personal computers.

## 4 Other issues

### 4.1 Progressive transmission

The prediction error can be compressed both in a lossy or lossless way. If a lossy scheme is chosen, it can be convenient to use a wavelet-based one such as SPIHT [8] or EZW [9] which allow for progressive transmission (i.e., a coarse approximation

of the compressed image can be reconstructed from any initial part of the resulting bit-stream). In order to understand why this can be convenient, consider a virtual world with a moving object. Each time the object moves the GE must generate a new view and send it to the client. If the object moves very fast, it could happen that the GE generates a new view before the complete description of the previous one is sent to the client. In applications like network games this would bring to an unacceptable loss of synchronization between the virtual world and the view shown to the users. However, since the view is changing rapidly, it is reasonable to assume that it should be possible to lower the quality of the compressed image since the user will not notice it. If the bit-stream allows for progressive transmission, this can be achieved by just stopping the transmission of the old image and beginning to transmit the new one.

### 4.2 Choosing the Transport Protocol and Resilience to Packet Loss

At the moment, in our experimental implementation, we are using TCP as a transport protocol because of its simplicity. However, the retransmission mechanism used in TCP could introduce too high an overhead in the interaction between the user and the virtual world. Because of this, it could be convenient to switch to an UDP-based protocol, such as RTP. This choice, however, introduces the problem of dealing with packet losses. Since the bit-stream created by the split-browser approach contains more precious data (projective transform parameters and low-resolution components) and less precious ones (high-resolution error components), we plan to use priority encoded transmission [10], [11].

## 5 Conclusions

The construction of articulated and photo-realistic 3D virtual worlds is at reach of current computer graphics and 3D imaging technology. The philosophy currently adopted for the diffusion of 3D virtual worlds over the Internet via the VRML and/or current related extensions has a fundamental drawback: it calls for downloading at client side the 3D virtual world description. The split-browser approach solves this issue by giving an alternative solution

which transforms the problem of interacting with virtual worlds in an image transmission task.

This work addresses the crucial issue of compressing the image stream in order to allow smooth interaction between the user and the virtual world. The proposed compression scheme first decomposes the virtual world as a union of objects, successively it approximates each object as a polyhedron, and, finally, it compresses the image of each face of the polyhedron by predicting it with a projective transformation and sending the parameters of the projective transformation together with the prediction error suitably compressed. In order to allow automatic bandwidth/image quality tradeoff, the prediction error should be compressed with a multiresolution scheme which allows for progressive transmission (e.g. SPIHT or EZW).

Future research will address the issue of the transmission protocol to be used for the split-browser structure and (eventually) how to deal with packet loss.

## References

- [1] R. Bernardini and G. M. Cortelazzo, "An efficient network protocol for virtual worlds browsing," in *The 12th Tyrrhenian International Workshop on Digital Communications "Software Radio Technologies and Services"*, (Portoferraio, Italy), CNIT, Springer-Verlag, Berlin, Sept. 2000.
- [2] K. Engel, O. Sommer, and T. Ertl, "A framework for interactive hardware accelerated remote 3d-visualization," in *IEEE TCVG Symposium on Visualization*, IEEE, 2000.
- [3] K. Engel, O. Sommer, C. Ernst, and T. Ertl, "Remote 3d visualization using image-streaming techniques," in *Proc. of the International Symposium on Intelligent Multimedia and Distance Education*, 1999.
- [4] K. Tsoi and E. Grvller, "Adaptive visualization over the internet."
- [5] K. Engel and T. Ertl, "Texture-based volume visualization for multiple users on the world wide web," in *Virtual Environments '99. Proceedings of the Eurographics Workshop in Vienna, Austria* (M. Gervaut, D. Schmalstieg, and A. Hildebrand, eds.), pp. 115–124, 1999.
- [6] M. Bailey and C. Michaels, "VizWiz: a Java applet for interactive 3d scientific visualization on the web," in *Proceedings IEEE Visualization '97*, pp. 261–267, 1997.
- [7] I. Yoon and U. Neumann, "Web-based remote rendering with IBRAC (image-based acceleration an compression)," in *Proc. of EUROGRAPHICS 2000* (M. Gross and F. HopGood, eds.), vol. 19, Blackell Publishers, Oxford, UK, 2000.
- [8] A. Said and W. A. Pearlman, "A new fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 243–250, 1996.
- [9] J. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Transaction on Signal Processing*, vol. 41, pp. 3445–3462, Dec. 1993.
- [10] J. Albanese, Blomer, J. Edmonds, M. Luby, and M. Sudan, "Priority encoded transmission," *IEEE Transaction on Information Theory*, vol. 42, pp. 1737–1744, Nov. 1996.
- [11] A. Mohr, E. Riskin, and R. Ladner, "Unequal loss protection: Graceful degradation of image quality over packet erasure channels through forward error correction," in *Proc. of DCC*, 1999.