Sharing and immersing applications in a 3D virtual inhabited world

José Marques Soares, Patrick Horain, André Bideau

GET / INT / EPH - Intermedia
9 rue Charles Fourier, 91011 Evry CEDEX France
{Jose_Marques.Soares, Patrick.Horain, Andre.Bideau}@int-evry.fr

Keywords: Collaborative Work, Inhabited Virtual Worlds

ABSTRACT

In a remote computer-mediated collaboration, co-workers have a limited perception of each other. Virtual humans can be used to increase interaction by immersion in 3D collaborative virtual environment. We discuss solutions to immerse shared 2D applications in a 3D world and to represent user actions on the application as avatar actions in the 3D virtual immersive inhabited world.

1. Introduction

In a remote computer-mediated collaboration, co-workers have a limited perception of each other: they loose the sense of immersion. To solve this problem, some collaborative environments such as NetMeeting [1] take advantage of videoconferencing. Leung [2] claims that these systems lack the perception of immersion because each user sees other users in separate windows and he doesn't know who is acting.

In order to improve the perception of the group, some virtual-reality environments allow immersing 2D applications in a 3D world while representing users with avatars. In these environments, the uncomfortable lack of physical contact among distant users can be minimised by classical communication channels such as text, voice or video. However, gestures are part of human communication and may help understanding. This kind of communication can be enabled in a virtual collaborative environment with articulated humanoid avatars [3], which brings immersion in a 3D inhabited space.

VReng [4] allows immersing a 2D interface in a 3D virtual world. It uses VNC [5] to provide a virtual board on which 2D application is mapped. Unfortunately, in VReng, users actions on the 2D-interface application are not associated with avatars actions.

NetICE [6] allows sharing a whiteboard in a 3D world. User avatars exhibit body and facial animation, but these are triggered from a menu and are dissociated from user actions on the whiteboard.

A problem when 2D applications are immersed in a 3D virtual environment is that interaction devices on 3D spaces are sometimes the same used for 2D applications (mouse, keyboard). In this case, it is necessary to use a mechanism to switch between these two modes of interaction 2D/3D. Besides, 2D interfaces are usually poorly viewed and manipulated in the 3D worlds. To work around these limitations, NetICE allows switching temporarily from the 3D interface to the 2D application, but this breaks the ability to see other users avatars.

We propose to retain a permanent view of the 2D-application interface, and to augment it with a 3D inhabited virtual world where user actions are reproduced by avatars.

2. A 2D+3D-hybrid interface

Our interface has two separate areas: an *application space* and an *immersive collaborative space*. The *application space* is a high quality view of the 2D interface of the shared application on which users can directly interact. The *immersive collaborative space* is a virtual multi-user 3D world with a board on which the *application space* is mapped. Each user is represented with a humanoid avatar, defined according to the H-ANIM standard [7] that reproduces user actions on the *application space.*

Events in the *application space* trigger animations either predefined or generated on the fly.

The predefined animations of avatars can be:

- **Entering the collaborative space.** When a user connects into the system, his avatar is positioned at either side of the virtual board. In the same way, the avatar leaves the environment when the user disconnects.

- **Moving in front of the virtual board.** Only one participant can be active when sharing a single-user application. When a user requests to become active and the position gets available, his avatar is moved in front of the board. The active avatar returning to the original position indicates the liberation of the board by the active participant.
- **Raising the arm.** When sharing a single-user application, this posture indicates that the user is willing to interact.
- **Other communication gestures.** The user can choose to play some predefined animations available from a list.

The hand of the active avatar shows the positions of the events triggered by users actions in the *applicative space*. This allows a direct visualisation of user actions.

2.1. Prototypes and Architectures

Two prototypes have been developed to experiment this interface. One is a multi-user whiteboard application (Figure 1). The other allows sharing and immersing any standard single-user application (Figure 3).
The two prototypes rely on Java based client-server architectures. A Java server controls the general state of the system, manages client connections and broadcasts data to clients such as actualisation of the virtual board and avatar animation data. Clients consist of a Java program that receives data and applies them to the virtual world.
Avatars can play predefined animations in MPEG–4/BAPs format. This event is activated in both prototypes when a user chooses the animation in a list on the 2D-interface.
In real time animations the avatar arm movements are calculated by inverse kinematics with the IKAN library [8, 9]. For this, clients get the joint positions of avatar arms and the goal is the point of interaction on the 2D-application interface. This is the point where the avatar hand will touch the virtual board. Since IKAN is written in C++, we have created a dynamic library (DLL) that is called (using Java Native Interface) to calculate the avatar arm joint angles.
The particular characteristics of each prototype are presented thereafter.

2.2. Whiteboard

In the whiteboard prototype the avatar touches the vertices of the object being drawn to simulate the actions of the active participant on the 2D-interface (Figure 1). For this, the drawing data are extracted from the 2D-application window and sent to the server with the co-ordinates of the active avatar. Then the server computes inverse kinematics and broadcasts the drawn object with the active avatar animation to clients.

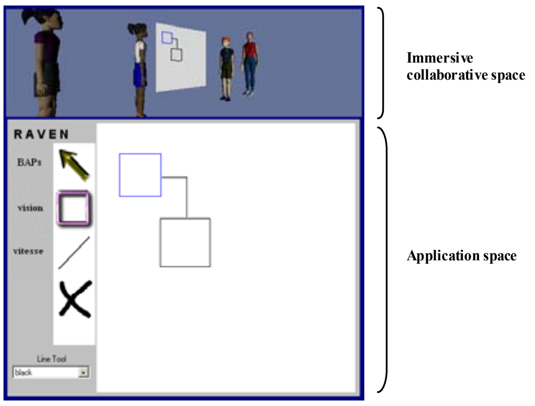

Figure 1 – Immersion of a Whiteboard

Following the structure of other networked virtual environments such as DeepMatrix [10] and Vnet [11], we have developed this first prototype on a Web client server architecture. A VRML file and a Java applet are stored in an HTTP server and loaded by the client through a Web browser. A VRML browser such as Cortona [12] or Blaxxun Contact [13] allows displaying the virtual world in the Web browser. The EAI (External Authoring Interface) [14] allows the client applet to interact of the VRML browser. Figure 2 shows this architecture.

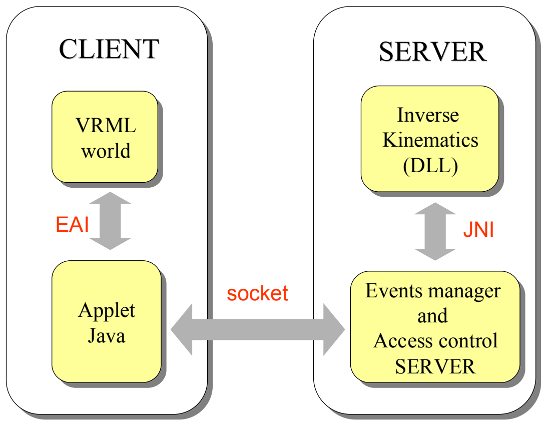

Figure 2 – Architecture 1: whiteboard

2.3. VNC client

The second prototype immerses a VNC [5] client in the 3D world. VNC (stands for Virtual Network Computing) is a remote display system, which allows viewing a computing desktop running elsewhere on the Internet. To simulate actions of the active participant, the hand of the active avatar follows the mouse in the *application space* (Figure 3 and 5).

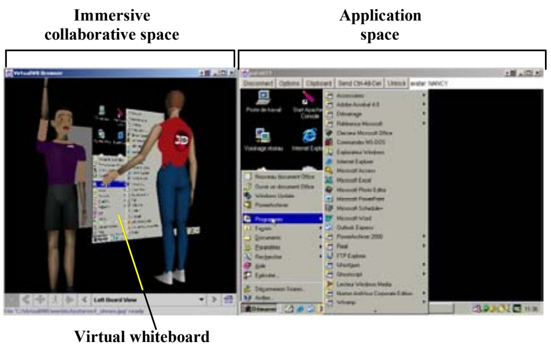

Figure 3 – Immersion of a VNC client

The client server communication in VNC relies on the RFB protocol [15] with a TCP connection. Each time a client interacts with the shared application, the VNC server broadcasts the image of the areas affected by updates on the remote interface. Unfortunately, the identity of the acting client and the event co-ordinates are not sent with the image fragments. This is a problem to represent actions with avatars in the 3D space. An open source VNC client was extended to solve this problem. It sends the co-ordinates of the user action on the 2D-application to our Java server, which in turn resends it to the clients. This communication is made with UDP datagrames to prevent a delay between animations and actualisation of the board in 3D world.

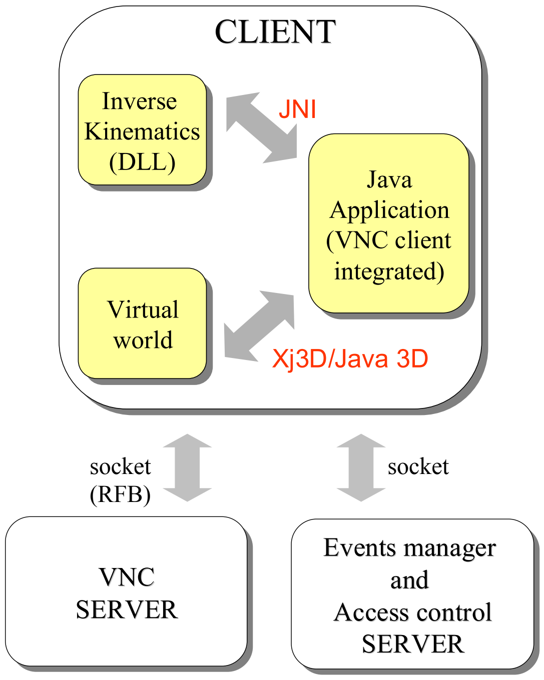

Figure 4 – Architecture 2: VNC Client

The server also controls user access to the application to prevent simultaneous client interactions. When the active user is changed, the server broadcasts this information and each client changes the active avatar for the next.

Because the VNC system has its own server, our clients are Java applications to work around applets security constraints. Figure 4 gives a general view of the architecture 2.

We use Java3D and Xj3D [16] to load the VRML models of the virtual world. Xj3D allows getting co-ordinates of the avatars body as well as to animate them through rotations and translations of their joints.

Figure 5 shows a simple situation where four users share a Windows application (Paint) to draw a picture. We can see a user asking to interact, while another is drawing.

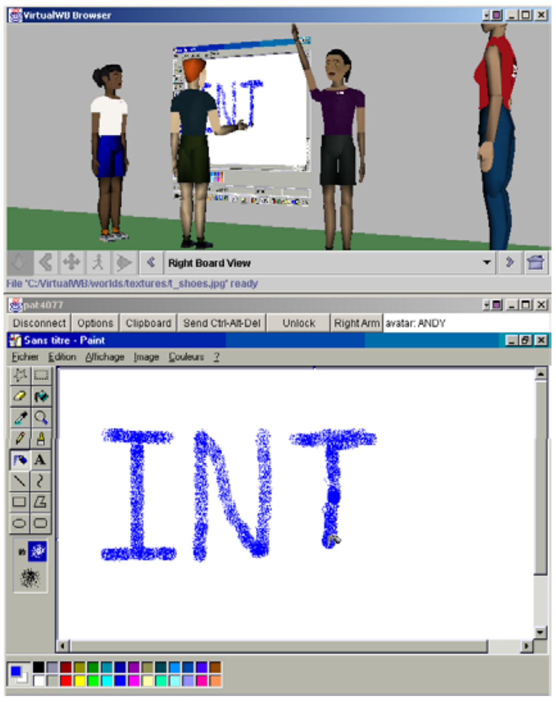

Figure 5 – Four users sharing a simple application

Figure 6 shows an avatar playing a predefined animation.

2.4. Comparison between architectures

The architecture 1 has some technical advantages as a web application: updates to the client code can be quickly and uniformly propagated and it is easy to make the system hardware independent on the client side with the Web browsers and Java applets. However, this approach to our system carries two limitations: One comes from the security constraints of non-signed applets that limits connections by sockets to another host than the HTTP server and does not allows the access to client resources. Sharing the 2D application with VNC system requires a VNC server. This limitation has been bypassed in architecture 2 by using Java applications rather than applets. The Java client

application can also directly call the inverse kinematics dynamic library while the server in architecture 1 computes it. This reduces possible server delays in diffusing avatar animation data.
Another limitation with architecture 1 appears when changing the image mapped onto the virtual board in the 3D space, which is done very frequently. The *PixelTexture* node is the only VRML resource that allows mapping an image stored in the client memory. We observed it is working slowly with the VRML plugins we tried. This was solved in the architecture 2 with the Java3D representation of the VRML board that allows efficient actualisation of the image mapped on it.

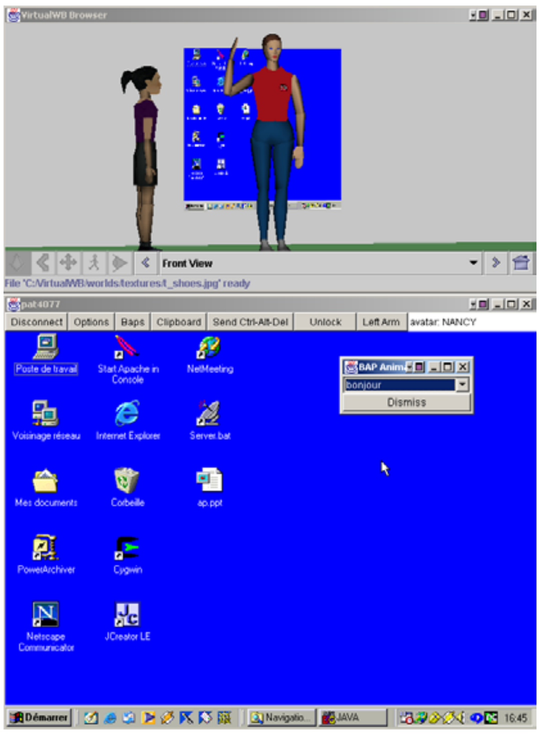

Figure 6 – Avatar playing a predefined animation

3. Conclusion and Perspectives

In order to increase the sense of collaboration among co-workers sharing a 2D-application, we have developed a platform in which user interactions are reproduced in a 3D virtual world.
We propose to augment the usual shared application space with an *immersive collaborative space*. With this new interface, it is possible:

- to see who is present in the collaborative environment;
- to see user entrances and exits;
- to see users actions on the shared application;
- to show users willing to interact.

Video, voice and/or text mechanisms could be used to provide the same set of information. However a 3D inhabited world allows a visual interface more intuitive than, for example, a message of text or an audio signal.
Using two separated areas in a same application interface can represent a new paradigm. A 2D+3D-hybrid solution can improve the sense of immersion in a remote collaborative environment while preserving the 2D-interface ergonomy and allowing comprehensive 3D views of the environment. In this way, users can see other participants acting in the shared workspace.
As a perspective, we plan to integrate a computer vision interface for gesture acquisition. Our method is the 3D human motion capture using a single camera, without markers and without a priori knowledge on gestures. It is based in superposing a 3D articulated model on colour images, respecting biomechanical constraints (for further information, see [17]).
The BAP file played by the avatar in Figure 6 is a result of our computer vision program applied on a real video sequence. This is a predefined way of gesture communication. Our present objective is to increase the non-verbal communication with restitution of distant users gestures in real time.
Our 2D+3D-hybrid interface might be used in collaborative scenarios that integrate a physical whiteboard client in the virtual world. Video projection and a wireless pen such as MIMIO [18] can achieve this. Besides representing the interactions with 2D interfaces, we intend to include users natural gestures realised out of the physical whiteboard.

References

1. Microsoft Corporation, *NetMeeting Home*, www.microsoft.com/windows/netmeeting
2. W. H. Leung & T. Chen, *Creating a Multiuser 3-D Virtual Environment*, IEEE Signal Processing Magazine, 18, May 2001, pp. 9-16.
3. A. Vuilleme-Guye; T. K. Capin; I. Pandzic; N. Thalman; D. Thalman, *Nonverbal Communication Interface for Collaborative Virtual Environments*, Virtual Reality J., vol.4, pp. 49-59, 1999
4. P. Dax, *VREng Virtual Reality Engine*, http://www.infres.enst.fr/net/vreng
5. AT&T Laboratories, *VNC - Virtual Network Computing*, http://www.uk.research.att.com/vnc
6. Advances Multimedia Processing Lab, Carnegie Mellon, *Projet NetICE*. http://amp.ece.cmu.edu/projects/NetICE/

7. Humanoid Animation Working Group, *H–ANIM specification,* http://H-Anim.org

8. Center for Human Modelling and Simulation, University of Pennsylvania, *IKAN: Inverse Kinematics using ANalytical Methods,* http://hms.upenn.edu/software/ik/ik.html

9. D. Tolani; A. Goswami; N. Badler, *Real-time inverse kinematics techniques for anthropomorphic limbs*. Graphical Models, 2000.

10. G. Reitmayr; S. Carroll; A. Reitemeyer - *DeepMatrix – An open technology based virtual envoronment system*, www.geometrek.com/developers/whitepapers

11. S. White, *VNet: Multi-User VRML System,* http://www.csclub.uwaterloo.ca/u/sfwhite/vnet

12. Parallel Graphics, *Cortona vrml client,* www.parallelgraphics.com/products/cortona/

13. Blaxxun Interactive, *Blaxxun contact - Multimedia communications client,* http://developer.blaxxun.com/download

14. *The Virtual Reality Modeling Language (External Authoring Interface) ISO/IEC FDIS 14772-2:2002,* http://www.web3d.org/technicalinfo/specifications/eai

15. T. Richardson ; K. R. Wood, *The RFB Protocol, Version 3.3*, 1998, http://www.uk.research.att.com/vnc/protocol

16. *Xj3D Open Source VRML/X3D Toolkit,* www.web3d.org/TaskGroups/source/xj3d

17. P. Horain, M. Bomb, *3D Model Based Gesture Acquisition Using a Single Camera*, IEEE Workshop on Applications of Computer Vision, Orlando, Florida, December, 3-4, 2002, pp. 158-162. http://www-eph.int-evry.fr/~horain

18. Virtual Ink Corporation, *MIMIO products*, http://www.mimio.com