Journal of Universal Computer Science, vol. 14, no. 19 (2008), 3071-3084
submitted: 23/7/08, accepted: 29/10/08, appeared: 1/11/08 © J.UCS

A Model of Interaction for CVEs Based on the Model of Human Communication

Diego Martínez, Arturo S. García
Jonatan Martínez, José P. Molina
Pascual Gonzalez
(LoUISE research group, University of Castilla-La Mancha, Albacete, Spain
{diegomp1982, arturo}@dsi.uclm.es, jonatan.m@gmail.com
{jpmolina, pgonzalez}@dsi.uclm.es)

Abstract: This paper summarizes a model of interaction for CVEs inspired by the process followed in human communication in the real world, detailing both the main elements and the communication process itself. The model proposed copies some properties of the real world communication but also allows the easy integration of Task Analysis to the design of CVEs, helping the developer in the design of the application. Furthermore, some of the benefits that the usage of this model brings to the user are also shown. Finally, some implementation details of a prototype supporting the described model are given. This prototype is used all along the paper to illustrate the explanation of some parts of the model.

Keywords: Virtual Reality, Collaborative Virtual Environments, Human-Computer Interaction
Categories: I.3.7, L.3.1, L.6.2

1 Introduction

Collaborative Virtual Environments (CVEs) have been a research field for many years, but it is in the very last few years when they have grasped more attention. On the one hand, even though some Virtual Reality (VR) devices are still too expensive, graphic cards with a high performance can be found in almost any PC at a more and more affordable price [Leavitt, 01] and, besides, some new devices are approaching VR technology to the general public, as the Wiimote device from Nintendo, including an inertial orientation sensor and an optical tracking system [Chung, 08]. On the other hand, data transfer through the Internet is getting much faster and reliable, making possible not only the communication between two users, but also allowing many users to collaborate and communicate online simultaneously [Johnson, 98].

By now, the most important inhabited Virtual Environments (VEs) in terms of number of users are focused on leisure and entertainment, such as the popular SecondLife [Linden, 08] and World of Warcraft [Blizzard, 08]. So do other recent releases, such as Lively [Google, 08] or Exit Reality [Exit, 08]. But beyond 3D chats and videogames, there is also a growing interest in using this technology in education –i.e. Croquet [Croquet, 08]- and professional areas –i.e. TrueSpace [Calligary, 08]-, and the impact of using new immersive interaction devices can boost their application to other areas.

Even though technology seems to be ready and there is a higher demand for CVEs from the society, developing these systems is still a hard task. Apart from some existing academic solutions, such as DIVE [Frecon, 98] or MASSIVE [Greenhalgh, 00], the developer needs to master a broad set of tools to fulfil her development: visual tools for creating 3D models of the objects, such as Blender [Blender, 08], and file formats to store the 3D models created, such as VRML or X3D [W3D, 08]; APIs for rendering 3D graphics, such as OpenGL [OpenGl, 08], usually combined with libraries that load the 3D models in memory and store them as a scene graph, such as Ogre3D [Ogre3D, 08] or OpenSceneGraph [OpenSceneGraph, 08]; and libraries for managing VR devices, such as VRJuggler [VRJuggler, 08] or OpenTracker [Reitmayr, 01].

Those are just examples of the many tools a developer has to cope with during the development of a CVE. But apart from finding and mastering the appropriate tools to perform her task, one of the main difficulties when building a CVE is the design of the logic of the application. The available tools cited before, allow us to read the input devices used and also to present the VE through the output devices chosen for the user, stimulating her sight, hearing or touch. However, those tools do not describe the dialog between the representation of the user in the system and the rest of the users and objects populating the environment, even though this will be the element guiding the interaction and the mutual perception of the inhabitants of the CVE. In the few platforms found in the literature describing these aspects [Pettifer, 00] [W3D, 08] [Greenhalgh, 00], a division between the scene and the structures representing the logic of the application is made. In these systems, the communication among the objects is described in a logical way, specifying connections and communications in a logical graph that does not belong to the 3D space that the objects are sharing.

The interaction model described in this paper tackles the problem of defining the logic of a CVE, focussing mainly on the communication among the objects. Communication is considered as the main vehicle to structure interaction and mutual perception and, thus, the key element to define the logic of the application. Also, the model puts together the 3D space and the logic of the application, using the shared scene graph as the transmitter of the communication among the inhabitants of the CVE. The scene graph does not only describe the space by means of the objects it contains, but it represents the medium shared by the inhabitants of the CVE and, thus, the key element that supports their communication and allows messages to be sent, transmitted and received. This work is an extension to the model proposed in [Martinez, 07]. Some elements remain as proposed, such as the ideas of *Action*, *Interaction* or some concepts of the scene graph, while others have been modified and some new concepts have been included. Also, to support the ideas presented in this paper, an early prototype has been developed to show the utility of the model and to illustrate some of the concepts it proposes.

2 The prototype: A Lego-like building game

In order to illustrate the behaviour of the model and to help in the explanation of some of the elements proposed, a sample environment was implemented. This environment consists of a virtual room where the user can pick pieces and join them to build models in a similar way she could do playing with the popular Lego game.

The communication model makes no assumptions about the kind of input or output devices used in the application. This allowed the prototype to be easily adapted to several different configurations: a common desktop configuration –using a keyboard and a mouse-, an immersive configuration –using a HMD, data gloves, haptic feedback and trackers- and a configuration using the novel Wiimote device.

The prototype was implemented in C++, using the *pthread* library for synchronization and thread creation, and *std* to define the required data structures. This prototype follows a plug-in schema that allows the use of the different input and output devices without having to modify the program itself. For the implementation of these plug-ins some other libraries were required: OGRE3D [Ogre, 08] for the rendering, and OIS [OIS, 08], wiiuse [wiiuse, 08] and VRJuggler [VRJuggler, 08] to gather information from the input devices.

3 The Communication Model proposed

In this section the key ideas and the main elements that converge on the communication model proposed are described. In order to achieve a complete definition of the communication model, human communication and its main elements –sender, receiver, channel, context, etc (see figure 1) - have been studied and adapted to the context of CVEs.

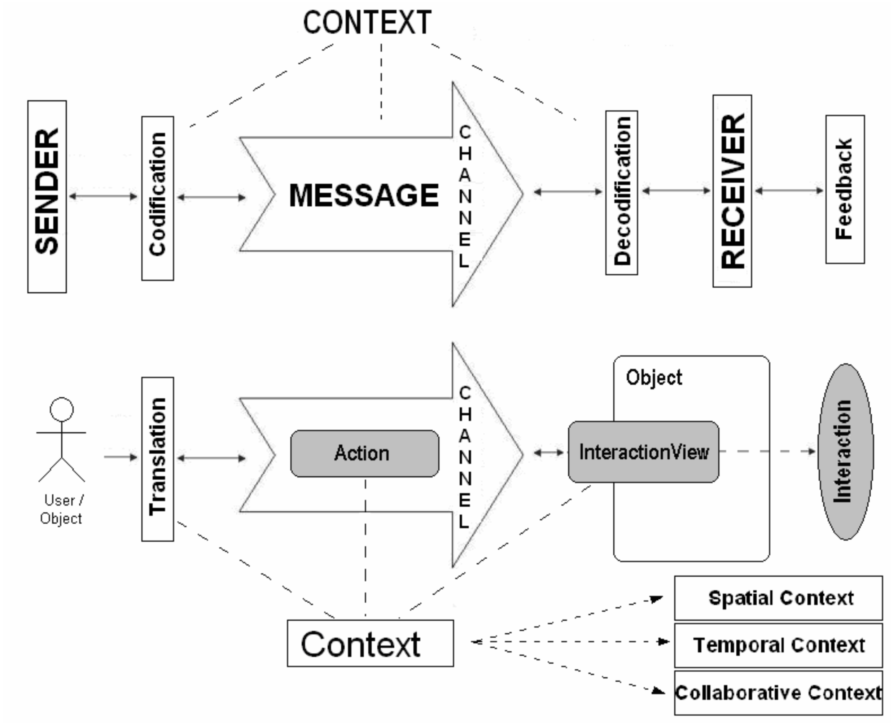

Figure 1: The process of human communication and its correspondence to the model.

The process of human communication can be understood as “every social interaction process by means of symbols or any messages system. It includes any process in which the behaviour of a human stimulates the behaviour of another human" [Pichon, 85]. This definition describes communication as an element that allows information transfer but also that triggers behaviours at the receiver. Thus, communication can be seen as the vehicle that allows a sender to interact with the receiver of her message.

However, this definition does not highlight some aspects of the communication that, even though they may seem obvious in the real world, have to be taken into account when modelling communication in a CVE. Humans can communicate because they share a common space –the real world-, and that space allows the transmission of several messages. We can see each other because the light is reflected from the observed person and arrives to the eyes of the beholder; we can talk to other people because the sound of our voice travels through the air and reaches the ears of our listeners. Also, if we consider the real world as the only element that the sender and the receiver share, it is obvious that messages must be elements of the real world, so that sender and receiver can share them.

In a CVE, the counterpart of the real world –understood as a shared space allowing communication- could be the shared scene graph. This model uses the shared scene graph as the key point for communication, but it adds some elements observed from the real world, such as the addition of some channels to the scene graph –so that the messages can be delivered-, or considering the messages exchanged as elements of the shared scene. The adaptation of the basic elements of communication to the particular case of CVEs will be explained in the following sections.

3.1 Channels

As it has already been said, the world is not simply a dark 3D space in which humans coexist, receiving no information about the rest of the elements of the world. It is a medium that allows the transmission of several stimuli –light, sound, heat, etc- and, thus, a medium that permits communication. Human beings have senses –sight, hearing, touch, smell and taste- that allow them to receive those stimuli. Like senses act as receivers of the stimuli in a channel of the real world, objects in the proposed model define receivers for each of the channels of the CVE.

The main problem rises when trying to specify the channels that are defined by the virtual scene. It is easy to accept that users' senses determine the available channels through which they can receive information from the system. These 'human channels', however, are not well suited to manage the information sent from the user to the system. Even though it would be possible to use cameras or microphones, those are not the best input for computers as they are not able to get information from images or sound in the same way humans do. But, more important, they do not need to receive that kind of information. If attention is focused on the kind of information an object needs to receive, it turns out that it just needs to perceive the user actions required to fulfil the tasks it is associated to.

Thus, we propose the use of a *Hierarchical Task Decomposition* (HTD) [Bowman, 99] describing the tasks a user will perform in the CVE and the actions she will perform on each object to fulfil those tasks. The scene would only need to include one channel for each of the basic actions in the HTD –so that the objects can receive the messages they need, each of them corresponding to a basic action in the

HTD-. Additionally, some other channels would be defined to allow the user to perceive the CVE. Each of these channels would be associated to one of the senses of the user.

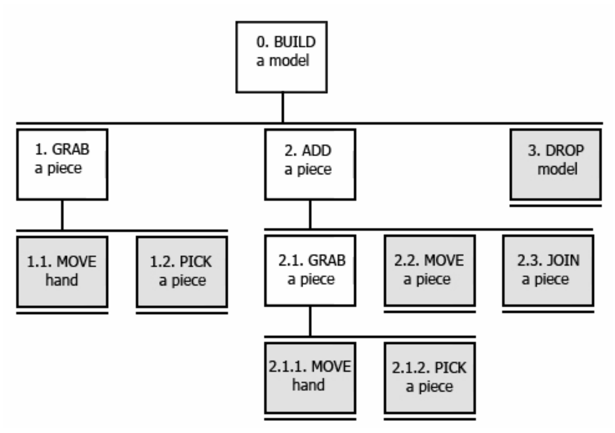

Figure 2: The basic actions in the Hierarchical Task Decomposition done for the prototype allow us to identify the channels required in the CVE.

The HTD of the prototype implemented used four basic actions –PICK, DROP, JOIN and MOVE-. Having a channel for each of those four actions, together with the VISUAL channel necessary for the user to visually perceive the CVE, suggested that the scene graph would require up to five channels –PICK, DROP, JOIN, MOVE and VISUAL-.

3.2 Context

The context is defined as the information known by both the sender and the receiver that is not held in the message being transmitted, but that influences its meaning. When speaking about the human communication, the context is quite complex, as it covers the complex human understanding of the world, including psychological, social and physical aspects. On the other hand, when speaking about CVEs the receiver of the messages will be, in many cases, a computer. Thus, the kind of context to use is usually much simpler.

The model assumes that, given the nature of a CVE, every element of the environment will have the following properties:

- **Spatial Context:** Each element of the scene will occupy a volume in the space.
- **Temporal Context**: Each element of the scene will exist during a given period of time.
- **Collaborative Context:** Each element of the scene will exist for a part of the members of the CVE.

These properties will be considered as implicit for any element of the scene and they receive the name of *Context.*

3.3 Messages

Messages represent the information transmitted during a communication process. They are elements of the scene produced by the objects when they start a communication process, and they contain: the producer of the message, the channel through which it is propagated –identifying an action of the HTD-, the context describing where, when and to whom the message is produced and any other additional information required. Given that they are usually associated to a basic action in the HTD, messages in the CVE are usually referred to as *Actions*.

3.4 Sender and receiver

The definition of communication studied assumes that humans will be the senders and the receivers of the messages. In the context of CVEs, any element of the environment can take part in a communication process. This fact results in the definition of four kinds of communication: human-object or object-human (human-computer interaction), human-human (social interaction), object-object (multiagent systems).

The sender will be any object in the environment starting a communication process, whereas the receiver will be any object who gets aware of a message, interprets it and reacts to its meaning.

The way objects receive messages is also inspired in human communication. Just like humans rely on their senses as receivers of the messages propagated through the real world, the objects in the scene have *interaction views*. In order to receive messages from other object of the CVE, these receivers will also belong to the scene, and thus, they will have an associated Context (spatial, temporal and collaborative). They will receive the messages existing in the scene graph, filter them according to some factors, and communicate them to the object for their processing.

The filtering made by the *interaction views* defines the interests of the object or its perception capabilities. Just like human eye does not transmit every electromagnetic stimulus to the brain, but only those between infrared and ultraviolet light, *interaction views* filter the messages sent according to the properties the *Action* and *interaction view* have in the CVE. While the conditions defined in the real world can refer to any of the many properties that real objects may have –such as the wavelength in the example, weight, chemical composition-, the properties of the objects of the proposed model are much more restricted (spatial, temporal and collaborative context) and, thus, the conditions the *interaction views* will be able to define are much simpler. However, these few properties allow the definition of rich and general enough conditions, such as defining where or how the message must be located from the receiver, when it can be received, its appropriate senders, etc.

4 Communication process

Once the basic elements that participate in the communication process have been described, the process itself will be detailed. It is summarized in figure 3, which shows an example of how a user, controlling her avatar, sends *Actions* through the scene graph, and how those *Actions* are received by the *Interaction Views* and processed by the *Objects*. To get a better understanding of how this works, each of the phases in the process –codification, transmission and reception- are defined in the following sections.

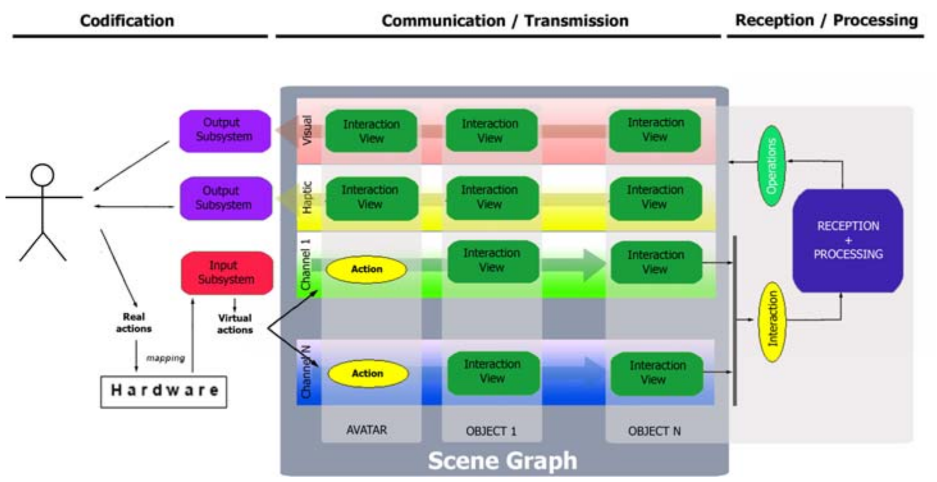

Figure 3: Diagram that summarizes the proposed model.

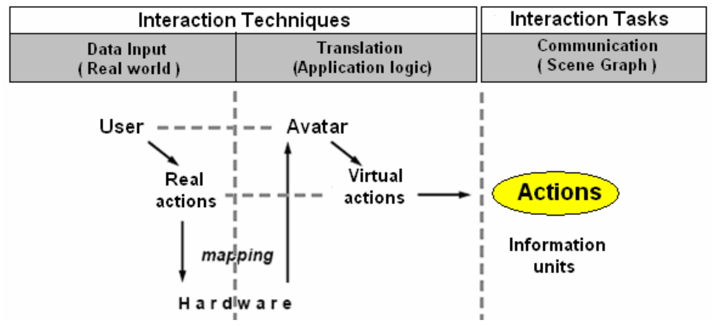

Figure 4: Translation from real world actions into virtual actions.

4.1 Codification

At this stage, the sender chooses the appropriate symbols to transmit its message to the receiver in a way that can be understood by that receiver. In the proposed model, this stage is translated into the creation and transmission of one or more *Actions*. Even though, as it is explained in section 2.4, this can be done by any object, the most interesting case is the one where a user -or any other real entity- acts as the sender. In this case –as it can be seen in figure 4- codification is divided into two stages: data input and translation, while the rest of the objects –which do not communicate with the real world- only execute the second stage to generate the virtual *Actions*.

The first stage deals with the borderline between the real and the digital world, retrieving some parameters that are adapted during the second stage to the appropriate format for the VE. These two stages together describe the input techniques available for the user, linking her actions in the real world to their meaning in the virtual world.

During the first phase –data input stage-, the data from the input devices must be accessed and the input modalities processed, translating the real actions into input parameters for the system. In the developed prototype, the elements that implement this task are the *InputSubsystems*, which encapsulates the communication with the drivers and the data translation to a given set of parameters, in a similar way as in [VRJuggler, 08] or [Reitmayr, 01]. The particular set of parameters to be gathered from the real world depends on the logic of the application. In the prototype, the following parameters were required:

Name	Type	Meaning
headPos	PositionInputSensor	Position of the head
rightHandPos	PositionInputSensor	Position of the right hand
leftHandPos	PositionInputSensor	Position of the left hand
rightPickGesture	DigitalInputSensor	Pick gesture performed with the right hand
rightDropGesture	DigitalInputSensor	Drop gesture performed with the right hand
leftPickGesture	DigitalInputSensor	Pick gesture performed with the left hand
leftDropGesture	DigitalInputSensor	Drop gesture performed with the left hand

Table 1: Parameters defined for the developed prototype.

These parameters can be accessed by any object –usually avatars-. To do so, they use their associated name –“headPos”, “rightHandPos”, etc.-. The information is not directly visible to the objects, but instead they use a proxy –similar to the ones used by [VRJuggler, 08]- called *InputSensor*, which allows them to retrieve data granting mutual exclusion with other objects. Four kinds of InputSensors have been defined so far, thus resulting in four types of possible parameters for any application: *PositionInputSensors* -to encapsulate position and orientation information-, *DigitalInputSensors* –to encapsulate information about two states elements, such as buttons, pinch gestures in certain data gloves, etc-, *AnalogInputSensors* –to describe continuous values, such as temperature, finger flexion, etc- and *StreamInputSensors* –for text strings, files, etc.-

A third element makes the management of these parameters easier: the *InputManager*. This component keeps track of all the *InputSubsystems* used by the system and allows objects to access their *InputSensors* transparently, having no need to know the subsystems used or what subsystem is in charge of a given *InputSensor*.

Three different *InputSubsystems* were built for our prototype –see images in figure 3-. They were meant to control an avatar composed of a body, a head and two hands.

The first *InputSubsystem* used a common keyboard and a mouse, and was implemented using OIS. The second one used the Wiimote device and the free driver *wiiuse*. The third one was designed for an immersive configuration, where a pair of NoDNA data gloves and an Ascension Flock of Birds tracking system were the chosen input devices. For the implementation of this *InputSubsystem*, VRJuggler [VRJuggler, 08] was used.

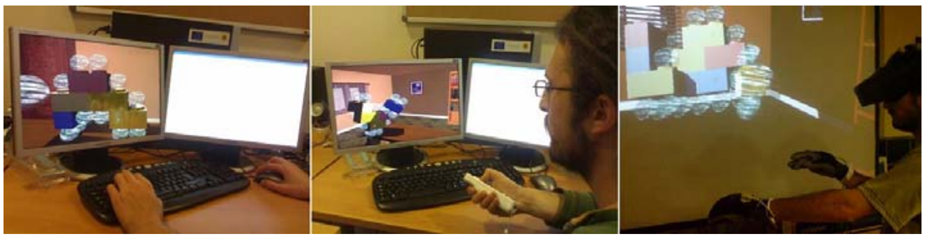

Figure 5: InputSubsystems used in the prototype: using mouse and keyboard; using the Wiimote device, and using data gloves and trackers.

4.2 Transmission

As it is shown in figure 3, second stage, the scene graph is the only communication medium in the system. *Actions* (messages) are transmitted as soon as they are stored in the channels of the scene graph. From that very moment, they can be examined by the *interaction views*. If the *Actions* fulfil the conditions of any of the *interactions view*, an *Interaction* will be generated and processed by the object. The *Action* has just been perceived by the object.

It must be highlighted that the scene graph has to be designed so that it includes the appropriate channels for the kind of messages that will be generated in the CVE. The kind of *Actions* that are likely to be transmitted in the CVE depends on two factors:

- The produced HTD, which shows the channels that the objects in the CVE understand –channels 1 to N in figure 2-.
- The user’s senses to be stimulated –VISUAL and HAPTIC channel in figure 2-

As it was explained in section 3.1, the developed prototype defines five channels, four for the user actions –PICK DROP, JOIN and MOVE channels- and an additional one for the user to perceive the environment. The haptic feedback used the PICK channel instead of a dedicated channel.

4.3 Reception and interpretation

The messages transmitted through the scene graph will be received by the *interaction views* in the objects and filtered according to their conditions. If the *Actions* fulfil the required conditions, they will be translated into an *Interaction* – describing the *Actions* that triggered it and the *interaction view* triggered- and notified to the object. The object will then process the *Interaction* and execute the appropriate response according to the task that this object is associated to.

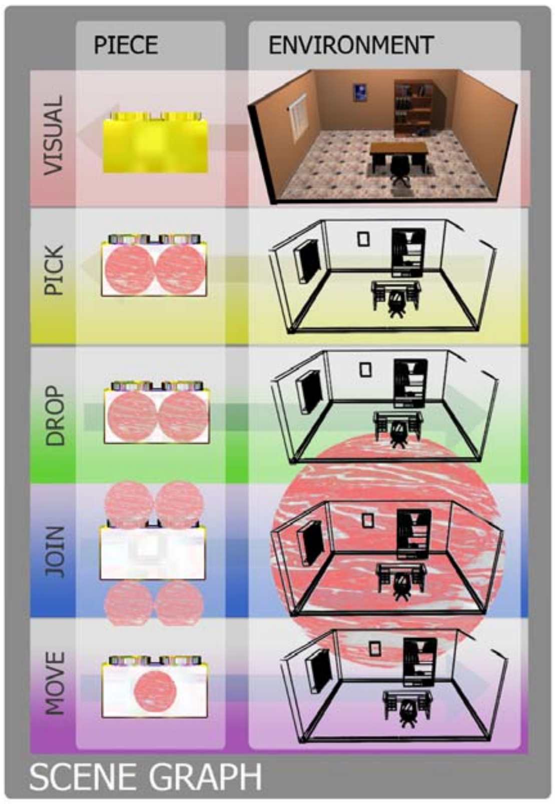

Figure 6: Some of the interaction views defined for the prototype.

Figure 6 shows two of the most interesting objects of the prototype, the environment and the pieces. The environment defines *interaction views* for the VISUAL channel –so that it can be observed- and the JOIN channels –so that users can release and leave pieces floating in the air-. The pieces allow a richer set of interactions. If a PICK *Action* –generated by an avatar’s hand- overlaps the PICK *interaction view of a piece,* then the piece is attached to the producer of that *Action.* The DROP *interaction view* reacts to DROP *Actions, and* generates a JOIN *Action*. Thus, if the user holds a piece in her hand and performs a DROP *Action*, then the piece transforms it into a JOIN *Action* and, as a result, it can be attached to another piece or released in the environment. The JOIN *interaction view* reacts to overlapping JOIN *Actions* –pieces dropped by the user- by attaching the piece that produces the *Action* to the piece that receives it. Finally, the MOVE *interaction view* allows the piece to be moved when held by an avatar's hand.

This model describes a behaviour that is quite similar to the one found in the real world, and can be used to describe the interaction between any two elements of the virtual world –let them be avatars or any other kind of object-. However, applying this model to some channels can end in unsatisfactory results. It happens, for instance, with the visualization of a scene. In order to perceive the VISUAL *interaction views* of the objects in the scene, the user´s avatar should define a VISUAL *interaction view* representing her eyes. On the other hand, light sources should generate LIGHT *Actions*. Some LIGHT *Actions* would reach the avatar's eyes, while others would be reflected on the VISUAL *interaction views* of the objects in the scene, which would transform the colour information held in each of those *Actions*. Actually, light sources should generate billions of such LIGHT *Actions*, and only a small part of them would finally reach the avatar's VISUAL *interaction view* –located at the virtual position of her eyes- and would be presented as pixels to the user. This scheme -which resembles the ray tracing rendering technique- is not efficient at all and it would be difficult to produce the 30 frames per second recommended for this kind of applications -as it happens with ray tracing-. The application of the model becomes worse in the case of the haptic channel, where the high number of samples per second required make time a much more critical factor.

To avoid these problems the implementation includes different *OutputSubsystems*, which simplify the communication model and are in charge of showing a part of the environment –a group of channels- to the user through a given device. These subsystems contain all the information about the *interaction views* – visual, tactile or auditive- that must be presented to the user, together with any other additional information required for the subsystems. This additional information is usually held in the *interaction views* of the objects represented, according to the output channel through which they will be presented –thus, VISUAL *interaction views* will define the meshes to use and tags identifying light sources, whereas HAPTIC *interaction views* will define other properties such as weight-.

In order to keep an independent execution from the main simulation cycle, these subsystems contain the following data:

- **Partial copy of the scene graph**: A local scene graph containing the representations of the *interaction views* of the CVE that must be shown to the user. These representations are stored in a local format that depends on the particular type of *OutputSubsystem* –meshes for a VISUAL subsystem, audio files for an AUDITIVE one, etc.-.
- **OuputSensors**: An *OutputSubsystem* requires knowing what to show –its local scene graph- but, also, where to show it from: A VISUAL *OutputSubsystem* must know where the eyes of the avatar are placed, whereas a HAPTIC subsystem will need to know the position of his hands and fingers. These objects, representing points of the CVE through which the user perceives the virtual world are tagged as *OutputSensors*. Each *OutputSensor* has a unique identifier and matches the location of an avatar's *interaction view* through which the user can receive information from the virtual world.
- **Operation mailbox**: This mailbox receives the updates done in the objects of the scene, so that the local copy of the scene graph evolves in the same way than the main scene graph.

Although *OutputSubsystems* are designed to show a particular channel of the system, they can be used to show other channels, even though the representations for those channels will not be so well suited –i.e. for a visual subsystem, the additional channels would be presented using the geometry of the *interaction view* instead of a mesh; in an auditive subsystem, those volumes would determine where they could be heard, and a pre-defined sound would be played-.

This idea, as it can be seen in figure 7.left, was implemented in our prototype, where the VISUAL and DROP channels where rendered through the HMD. In this case, the user could see the places where she could drop the pieces to join them to the figure she was building. Also, a HAPTIC *OutputSubsystem* was implemented that allowed the user to get vibro-tactile feedback when her fingers touched the PICK *interaction views* of the objects (7.right), getting information about when she could pick any object of the scene. Although no performance evaluation was done, this information was found useful by the users for the completion of their tasks.

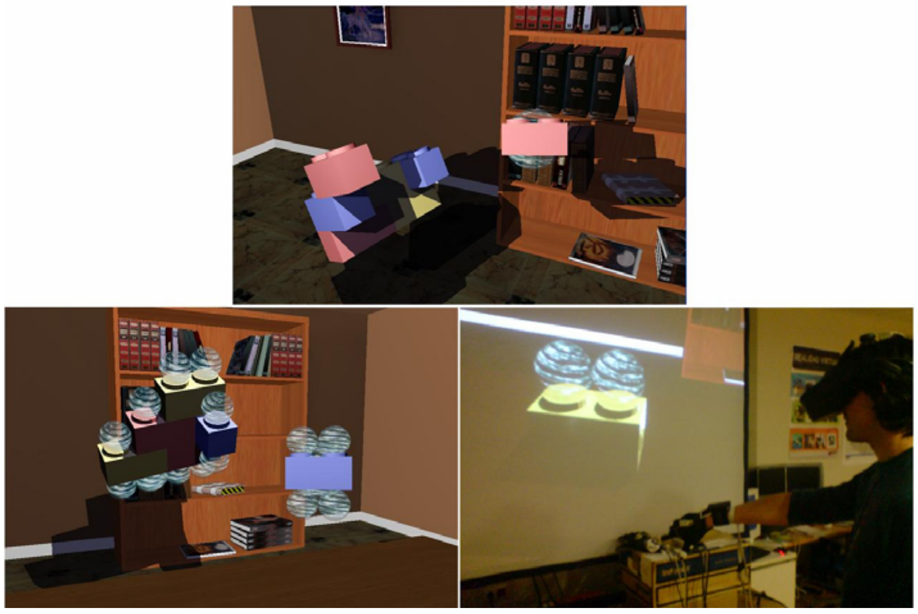

Figure 7: Screenshots of the OutputSubsystems of the prototype: rendering the VISUAL channel (up), rendering the VISUAL and PICK channels together (left) and a vibro-tactile OutputSubsystem for the PICK channel (right).

5 Conclusions

This paper tackles the complex problem of developing CVEs, trying to help in the definition of mechanisms for the communication and interaction among the elements that populate a CVE. For this purpose, a model of human communication has been analyzed and adapted for its usage in the context of a CVE.

The resulting model adds some elements that help the developer. On the one hand, it allows the easy integration of Hierarchical Task Analysis (HTA) in the design of the environment, assigning a channel for each of the basic actions in the HTD and making the design of the logic of the application easier. On the other hand, the use of contexts in both the messages and the receivers allows the easy definition of conditions determining when the communication is possible. The properties checked by these conditions -space, time and collaboration- are considered essential in CVEs.

Furthermore, this paper also describes the adaptations necessary to use this model in the communication with the user, defining the required elements –input and output subsystems- to cope with the frontier between the real and the digital world. The correctness of these elements have been shown in a prototype that defines three *InputSubsystems* to receive data from the user –using a mouse and a keyboard, a WiiRemote device and data gloves and a tracking system-, and two *OuputSubsystems* to show the environment to the user –one for the visual sense using OGRE3D and another one for the tactile sense using a custom-made haptic device-.

Additionally, the possibility of showing several channels of the CVE through the same *OutputSubsystem* was tested. The prototype allowed users to perceive channels associated to the tasks they had to complete in the system This feature brought them an intuitive feedback about how to perform their tasks, what was found helpful for the users. These initial results will now be studied by the means of a performance evaluation, comparing the completion time of different tasks with and without additional channels.

The continuation of this work will focus on another relevant aspect for the design of CVEs, and in how to integrate it into the communication model proposed: the feedback. This element allows the developer to communicate with the user, guiding her during her tasks [Barrileaux, 01]. Our research will now face the challenge of identifying the required channels and guidelines to easily add feedback to this model.

Acknowledgements

This work has been supported by the Junta de Comunidades de Castilla-La Mancha (PAI06-0093).

References

[Barrileaux, 01] Barrilleaux, J.: 3D user interfaces with Java 3D. Manning Publications Co. 2001.

[Blizzard, 08] Blizzard Entertainment Site. URL: www.worldofwarcraft.com

[Bowman, 99] Bowman, D., Interaction Techniques for Immersive Virtual Environments: Design, Evaluation, and Application, august 1999

[Caligary, 08] Caligary. TrueSpace 7. URL: www.caligari.com

[Croquet, 08] The Croquet Project. URL: croquetproject.org

[Chung, 08] Chung, J., Projects Wii. URL: www.cs.cmu.edu/~johnny/projects/wii

[Exit, 08] Exit Reality home page. URL: www.exitreality.com

[Frecon, 98] Frécon, E., Stenius,M., DIVE: A Scalable network architecture for distributed virtual environments, Distributed Systems Engineering Journal , Vol. 5, No. 3, September 1998, pp. 91-100.

[García, 07] García, A., Martínez, D., Molina, J.P., and González, P. Collaborative Virtual Environments: You can't do it alone, can you? Proc. of 12th International Conference on Human-Computer Interaction, HCII 2007 (Beijing, China, July 22-27, 2007), Volume 14, Lecture Notes in Computer Science, LNCS_4563, Springer-Verlag. ISBN: 978-3-540-73334-8. Pp. 224-233.

[Google, 08] Google Lively homepage. URL: www.lively.com

[Greenhalgh,00] Greenhalgh, C., Purbrick, J., and Snowdon, D., Inside MASSIVE-3: flexible support for data consistency and world structuring. In Proceedings of the Third international Conference on Collaborative Virtual Environments CVE '00. ACM, 119-127

[Jhonson, 98] Johnson, A., Leigh, J., Costigan, J., Multiway tele-inmersion at Supercomputing’97, IEEE Computer Graphics and Applications, July 1998.

[Leavitt, 01] Leavitt, N., 3D Technology: Ready for the PC? IEEE Computer Magazine, Vol. 34, No. 11, November 2001, pp. 17-20.

[Linden, 08] Linden Labs. SecondLife: Official site of the 3D online virtual world. URL: secondlife.com/

[Martinez,07] Martínez, D., García, A.S., Molina, J.P., and González, P., Towards an interaction model for CVEs. Proc. of 12th International Conference on Human-Computer Interaction, HCII 2007 (Beijing, China, July 22-27, 2007), Volume 14.

[Molina, 06] Molina, J.P., García, A.S., Martínez, D., Manjavacas, F.J., Blasco, V. and González, P., An Interaction Model for the TRES-D Framework. Proc. of 13th IEEE Mediterranean Electrotechnical Conference (MELECON 2006), special session "New interaction paradigms in Virtual Environments", Benalmádena, Málaga, May 16-19, 2006. Electronic Proceedings (CD-ROM). ISBN: 1-4244-008-0.

[Pettifer, 00] Pettifer, S., Cook, J., Marsh, J., and West, A., Deva3: Architecture for a large scale virtual reality system. In Proceedings of ACM Symposium in Virtual Reality Software and Technology 2000, October 2000, pages 33-39. ACM Press.

[Pichon, 85] Pichon, E., El proceso grupal de psicoanálisis a la psicología social, Ed. Nueva Visión, Buenos Aires, l985

[Reitmayr, 01] Reitmayr, G., Schmalstieg, D., OpenTracker-an open software architecture for reconfigurable tracking based on XML Virtual Reality 2001, march 2001, pp 285-286 .

[Ogre3D, 08] Ogre3D project homepage. URL: www.ogre3d.org

[OIS, 08] Object Oriented Input System. URL: sourceforge.net/projects/wgois

[OpenSceneGraph, 08] OpenSceneGraph website URL: www.openscenegraph.org/projects/osg

[VRJuggler, 08] VRJuggler project. URL www.vrjuggler.org

[W3D, 08] W3D Consortium. VRML specification. URL www.web3d.org/x3d/vrml

[wiiuse, 08] The Wiimote C Library homepage. URL: http://www.wiiuse.net/