In Proceedings of ACM Virtual Reality Software and Technology (VRST), December, 1999, London

Direct 3D Interaction with Smart Objects

Marcelo Kallmann

EPFL - LIG - Computer Graphics Lab

Swiss Federal Institute of Technology, CH-1015, Lausanne, EPFL – LIG

+41–21-693-5248

kallmann@lig.di.epfl.ch

Daniel Thalmann

EPFL - LIG - Computer Graphics Lab

Swiss Federal Institute of Technology, CH-1015, Lausanne, EPFL – LIG

+41–21-693-5214

thalmann@lig.di.epfl.ch

ABSTRACT

Performing 3D interactions with virtual objects easily becomes a complex task, limiting the implementation of larger applications. In order to overcome some of these limitations, this paper describes a framework where the virtual object aids the user to accomplish a pre-programmed possible interaction. Such objects are called Smart Objects, in the sense that they know how the user can interact with them, giving clues to aid the interaction. We show how such objects are constructed, and exemplify the framework with an application where the user, wearing a data glove, can easily open and close drawers of some furniture.

Keywords

Virtual Reality, manipulation, interaction, data glove, virtual objects, virtual environments.

1. INTRODUCTION

Virtual Reality (VR) technology has been employed on various different applications, as for example: human factor analysis, navigation in virtual environments, training, visualisation, and design of objects.

A common point to all applications is the fact that the user wears VR devices, immerses in the virtual environment (VE), and interacts with the virtual world in order to accomplish some specific task. In many cases, such a task involves direct manipulation of virtual objects.

Direct manipulation of objects in virtual environments is often awkward and inconvenient, because of mainly two factors: the use of simplified physical models due to computation time constraints, and limitations of the current VR devices.

A simple task of grabbing and moving a virtual object may be a frustrating experience because of the lack of a tactile feedback, weightlessness of virtual objects, positioning tracker noise, and poor design of interaction techniques, among other factors.

For direct manipulation, the most common used device is a *data glove*. This device has known a lot of enhancements during the last years [19]. However, limitations as the lack of haptic feedback, are still hard to solve.

Although direct manipulation intends to be similar to manipulation in real world, there are significant differences which have to be fully understood in order to exploit the full potential of VR technology.

If virtual systems are to be effective and well received by their users, considerable human factors issues must be taken into account [18]. Will the user get sick? Which are the important tasks to perform? Will the user perceive system limitations (e.g., flicker)? What type of design metaphors will enhance the user's performance in VE?

The main challenge concerns defining an efficient, simple and natural interaction paradigm, in order to overcome the VR limitations.

In this paper, we present an architecture where the virtual object aids the user by giving clues on how to accomplish a desired interaction task.

With our framework, we are interested on high level interactions, instead of a direct manipulation. We take into account interactions with objects having some functionality governing its moving parts, but that cannot be directly displaced by the user. Instead, the user can trigger the movements of the object, according to its functionality.

Objects' functionality is defined during the modelling phase. This is done by using a dedicated graphical interface that permits to define each possible behaviour of the object. In this way, during the simulation, the object knows which are the interactions that the user can trigger and aids the user to choose the desired one. Such objects are called Smart Objects [6].

In this paper, we present how to model Smart Objects for VR applications. We describe also a prototype application where the user, wearing a data glove, interacts with some smart objects, by means of a high level interaction metaphor.

2. RELATED WORK

There are many techniques being used for interaction, manipulation and navigation in VEs. For instance, Mine [9] shows many examples of such techniques, including a VR metaphor for menu selection. In a more recent work [10], the concept of proprioception is exploited in order to enhance the direct manipulation of objects. An overview of techniques for object manipulation, navigation and application control in VEs is presented by Hand [5].

In order to implement a complex VR application, we can identify three main distinct layers, which have to be correctly designed and put together:

- The low-level physical simulation model.

- The direct manipulation metaphor.

- The direct high-level interaction metaphor.

The physical simulation model should give a physically based visual feedback to the user when an object is touched, deformed, or moved, correctly managing all possible intersections.

The manipulation metaphor layer is responsible to define how the user, wearing VR devices (as a data glove), can interact with the virtual objects in order to touch, move and displace them. This metaphor is directly linked to the adopted physical model.

Finally, the high level interaction layer will permit the user to achieve other tasks that are not feasible by means of a direct manipulation, but, for instance, by means of user gestures.

Many physical models have been proposed in the literature. For instance, Sauer and Schömer [17] describe a rigid body simulation tool for the simulation of unilateral contacts, filling the gap of impulse-based and constraint-based simulations, including a friction model. An approach to model a haptic glove force transference was proposed by Popescu et al. [15].

An interesting approach has been proposed to deal with collision and interference in VR systems, making some use of the graphics rendering hardware in order to minimise time computation during a virtual hand grasping application [1].

Many manipulation metaphors have been also proposed. For instance, Poupyrev et al. [16] present a manipulation metaphor based on three main steps: Selection, Positioning, and Orientation. Boulic et al. [2] present an approach where each finger of the virtual hand has spherical sensors for detecting collision with the virtual object. These sensors are used for deciding when the virtual object is grasped or when the virtual hand needs a posture correction.

As commonly stated, object manipulation needs to be optimised [16] in order to let the immersed participant to concentrate on high-level tasks rather than on low-level motor activities. Some solutions to this matter start to be proposed [8].

Unfortunately, less attention has been given to exploit implementations of high-level interaction metaphors. Existing works remain in the theoretical level [22], or mainly concern hand gesture recognition, as for instance, a dynamic two-handed gesture recognition system for object modelling [11].

Aiming to fulfil this gap in the VR research, we propose a framework to perform high level interactions with virtual objects that, by knowing how the user can interact with them, give clues to aid the interaction. Such objects are called Smart Objects [6].

The adjective *smart* has been used on different contexts, as for instance, to refer to computers that can understand human actions [14]. In our case, an object is called *smart* when it has the ability to describe its possible interactions.

Graphical languages as VRML [20], are popular to exchange 3d models, but less suitable as a way to specify object's functionality. In VRML, most of the interactive behaviour must be written in external scripting languages such as Java or JavaScript.

In another direction, some work have been done in order to link language to modelling [12], and also towards a definition of a standard and data structure-independent interface to model geometric objects [3]. In our case, we use language to describe object's functionality.

We do not know any related work that explores specifically this aspect of modelling object's functionality for a real time simulation or animation. However, a close work is presented by Okada et al [21]. In their work, a collection of *intelligent boxes*, each one having a basic specific behaviour, can be connected in order to construct a more complex object. Such approach is suitable to define objects with reactive movements according to its connections. In our case, we are interested on defining, not only movement functionality, but also semantic meaning as its properties and design intent. Such information is stored in a form of a script language that defines each object's behaviour.

Smart objects are modelled using a dedicated graphical interface modeller, where the functionality can be defined. Object's functionality is defined by a sequence of behavioural instructions forming a plan that describes each action to be performed (by the object and by the user). The next section describes this modelling phase of smart objects.

3. MODELLING SMART OBJECTS

We follow a *Feature Modeling* approach that is a vast topic in the engineering field [13]. The basic idea is to identify and define, during the modelling phase, all interesting features of the object.

In a previous work [6] a more detailed description of this approach is done, but towards simulating interactions between virtual humans and Smart Objects.

3.1 Interaction Features

In the scope of our VR applications goals, we can define four classes of different *interaction-features*:

- *Intrinsic object properties*. For example, the description of the movements of each object part.

- *Interaction information*. These are the positions of interactive parts (knobs, buttons), and the hand shapes and locations to interact with them.

- *Object behaviours*. These are available depending on object's state. For example, a door has an available behaviour to *close* itself only if its state is open.

- *Expected user behaviours*. These are associated with object behaviours in order to indicate when and where the user should put his hand.

3.2 A Smart Object Example

Figure 1 exemplifies the modelling phase of a smart desk. This desk has five interactive parts: a lamp, a book, two drawers and a small door. Each of these parts has an indication of where the user should put its hand in order to interact with that part. This indication is modelled by placing a hand object to serve as a clue. These *hand clues* define the expected hand shape and location in order to perform an interaction.

We can also notice in figure 1 the definition of a reference position that is used to indicate a nearby position that the user should be in order to interact with this object.

Once these properties are identified, the behaviours of the objects can then be defined. The smart desk contains a total of five pairs of behaviours, each pair relative to one interactive part.

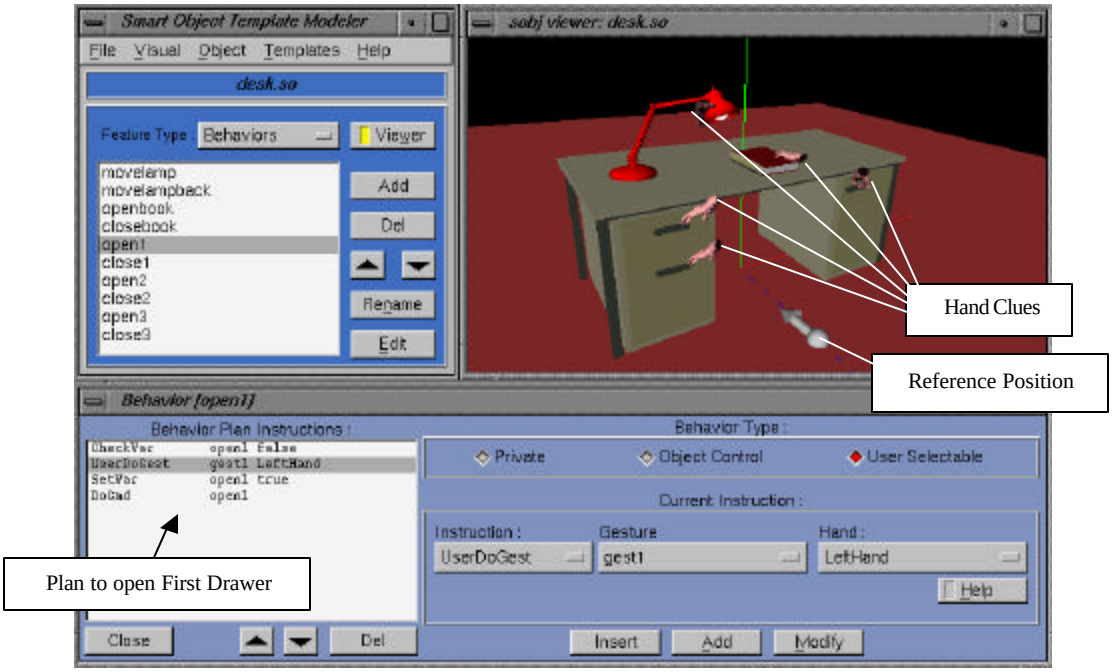

Figure 1. Modelling the behaviours of a smart desk.

For instance, the first drawer of the desk has two behaviours: one for opening (*open1*), and another one for closing. Only one among these two is available to be selected by the user, depending on the state variable of the drawer. Each time one is selected, the state variable changes, switching the available behaviour.

In figure 1, the behaviour *open1* is selected on the behaviour editor dialog box. This behaviour describes a plan to be performed in order to make the first drawer of the desk open. This plan is composed of four simple scripted instructions that are to be interpreted one after another:

- *CheckVar open1 false.* Here the drawer state variable is checked. Only when the state variable means that the drawer is closed that this open behaviour will be available to be selected by the user.

- *UserDoGest gest1 LeftHand.* This instruction says that the user should have its left hand in a similar shape and location given by the hand clue *gest1*. In our prototype implementation, we consider that all gestures can be performed by both the right or left hand.

- *SetVar open1 true.* After the user has put its hand in the correct position, the state variable of the drawer is changed, changing this behaviour to be unavailable, but making the other behaviour of closing available.

- *DoCmd open1.* Finally, this instruction triggers the translation movement to make the drawer to open. The argument *open1* refers to a translation movement interactively defined before.

3.3 Smart Object Behaviours

Objects functionality is defined by the set of its behaviours. A variety of instructions are available in order to model more complex behaviours. Most of these instructions provide similar capabilities of those available on standard programming languages. Once we have the flexibility to describe the logic of any complex behaviour, the limiting aspect is related to the possible instructions to animate the object and to describe expected actions of the user. The more difficult is to define what are the user-related instructions that can be used.

It is possible to compare a smart object to a dialog box of a graphical user interface: both show their interactive parts (like buttons), and react, according to their functionality, when some interaction is selected.

Two distinct phases exist in this process: the interaction selection, and the behaviour interpretation. Depending on the type of the user that interacts with a smart object, these phases must be considered differently.

For example, a virtual human user can compare semantic names of behaviours in order to choose one based on some reasoning process. We can also imagine a (real) user that selects behaviours by choosing one from an available list. In the case of our VR application, the immersed user will select behaviours by matching its virtual hand to hand clues, as will be explained in the next section. Once a behaviour is selected, each instruction is interpreted in order to perform the interaction.

Our current behavioural language considers two main user-related instructions: the *UserDoGest*, and the *UserGotoPosition* instructions. As already described, the *UserDoGest* instruction says that, at a certain moment of the interaction, the user needs to have its hand on a specific location. In a similar way, the instruction *UserGotoPosition* means that the user should be in a certain location.

During the interpretation of the behavioural instructions, when a *UserDoGest* instruction is found, the application shows the related hand clue and waits the user to put its virtual hand near the clue before skipping to the next instruction.

When a *UserGotoPosition* instruction is found, the application shows the goal position clue and waits the user to get closer to it before skipping to the next instruction. In order to support this instruction, the VR interaction metaphor must include some efficient navigation solution.

With these two user-related instructions it is possible to model many complex high-level interactions, as for example, entering in a lift [6].

Although a complete set of behavioural instructions has not yet been defined, some other behaviour definitions can be seen in previous works [6,7].

The prototype VR application described in the next sections does not consider a navigation metaphor for the immersed user. In this way, we consider only smart objects which behaviours are described with the four instructions showed in figure 1.

The next section details how the user, wearing a data glove, can select a desired available object behaviour.

4. INTERACTION METAPHOR

Once smart objects are modelled, they can be loaded into the virtual environment (VE). We make use of the interaction information that each smart object contains in order to facilitate the user interaction. This approach frees the user many difficult low-level motor activities.

We consider that the user is immersed in the virtual environment using a data glove and a six degrees of freedom tracker placed on the glove. In this way, the user can freely move its hand in the virtual environment (however in a restricted space). We consider that the position of the user in the VE is the position of its virtual hand representation, captured by the positional tracker.

Two main modules control our interaction metaphor: The smart object controller, and the interaction manager.

The interaction manager is responsible to aid the user to select available smart object's behaviours, while the controller interprets the selected behavioural instructions.

4.1 The Interaction Manager

The interaction manager monitors the user position in relation to each object reference position. When the user reaches a certain distance from the object reference position, we say that the user is inside the interaction range of the object. In this way, a dynamic list of all objects inside the interaction range of the user is maintained updated.

For each smart object in range, all available behaviours are checked in order to determine those that are closest to the current user's hand position. This is done by measuring the distance of the user's hand position to the clue hand that each behaviour specifies as a parameter of its first *UserDoGest* instruction. Note that we can have a behaviour with more than one *UserDoGest* instruction, for example to describe an interaction requiring a sequence of buttons to be pressed.

All available behaviours in range have an associated hand clue. All hand clues that are within a certain distance (in relation to the user position) are displayed in the virtual environment, and are kept in another dynamic list. This list keeps a link to all available behaviours that are currently in range. Figure 2 depicts this architecture.

In this way, the interaction manager monitors the position of the smart objects and the user's hand, in order to display only the hand clues corresponding to closer available behaviours in range.

Once the user really places its hand near the same position and orientation given by a hand clue, the corresponding smart object behaviour is selected and interpreted by the controller.

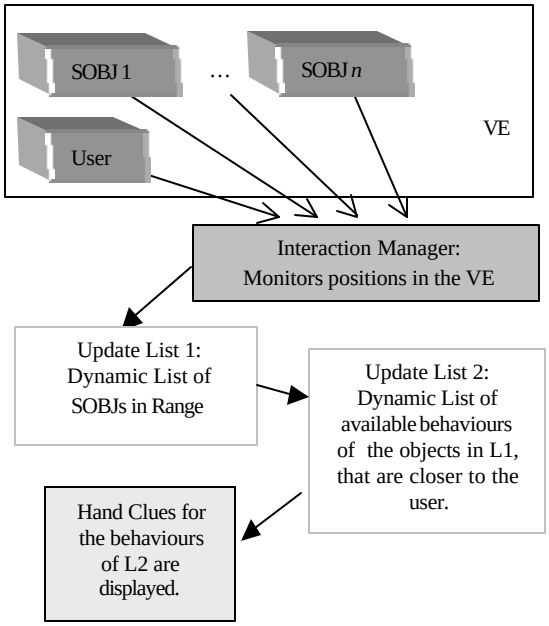

Figure 2. The interaction manager module keeps updated a list of smart objects in range and a list of their available behaviors that are closest to the user.

Note that when the user selects a behaviour, other behaviours may change their availability state, what will cause the interaction manager to dynamically update the displayed hand clues.

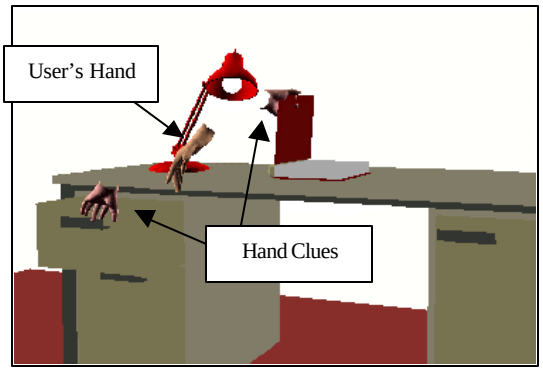

Figure 3. To close the book or the drawer of the smart desk, the user selects the corresponding object behaviour by placing the hand closer to the desired hand clue.

Figure 3 shows a snapshot of our prototype implementation where the user's hand position is in the range of two available behaviours of the smart desk: to close a book on it, and to close its first drawer. To trigger one of these two behaviours the user sees the two related hand clues, so that by just putting its hand near a hand clue, the associated smart object behaviour will be triggered.

Figure 4 shows another smart object that is a dossier containing six drawers. The behaviours definitions are similar to the desk drawer, so that the object has a total of six pairs of behaviours, each pair being related to each drawer (open and close). In this way, only six behaviours are available at the same time. Figure 4 shows two moments of the interaction of opening a drawer.

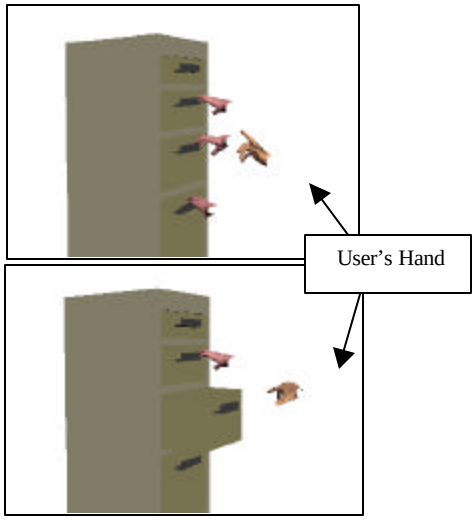

Figure 4. The first image shows three hand clues indicating that there are three available interactions to open drawers of the dossier. The second image shows the final state of the drawer after the middle one is chosen.

4.2 The Smart Object Controller

When a hand due is selected, the smart object controller starts to directly interpret each instruction of the related behavioural plan, animating the object and updating its internal state variables, i.e. performing the interaction.

As the first *UserDoGest* instruction found in the selected behaviour serves as the behaviour hand clue, this one is directly skipped. But, in the case where another *UserDoGest* instruction is found, the controller will wait the user to place its virtual hand near to the associated hand clue to then skip to the next instruction. In this case, all hand clues displayed by the interaction manager are turned off. Only the hand clue related to the current *UserDoGest* being interpreted is displayed.

Similarly, if a *UserGotoPosition* instruction is found, all other clues are turned off, just displaying the goal position clue that the user must reach in order to let the following instructions be executed.

The scenario is simple: the user can navigate with its virtual hand seeing many clues being turned on and off on the screen. The understanding of which interaction is related to a clue is obvious. For example, by seeing a hand clue positioned in the handle of a closed drawer, there are no doubts that the available interaction is to open the drawer.

In this way, all interactions are triggered by means of comparing distances, minimising the needed low-level motor activities of the user. Only when two hand clues are too close to each other that the hand posture of the user will be used in order to decide which interaction to select.

5. THE PROTOTYPE VR APPLICATION

Our prototype implementation uses one Ascension Flock of Birds (FOB) magnetic 3d positional tracker, attached to a Cyber Touch data glove from Virtual Technologies inc. To give a 3d visual feedback, we use a pair of Stereo Glasses attached to a SGI Impact machine.

We minimise the number of VR devices in order to reduce discomfort during usage. For this application, it is sufficient that the user wears only one data glove. Figure 5 illustrates a user wearing the needed VR devices and ready for using the application.

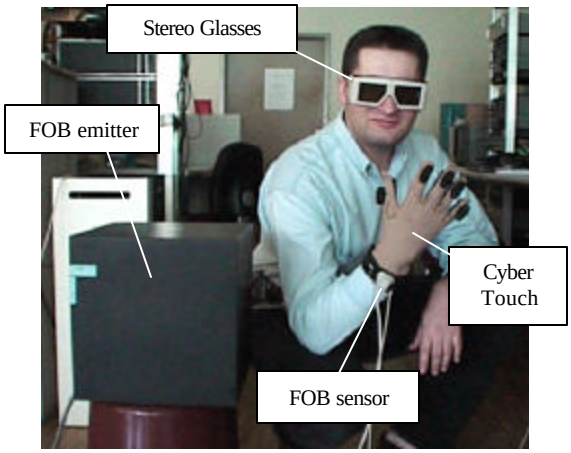

Figure 5. A picture of the real user with the needed VR devices, ready to use the system.

Figure 6 shows the considered scene. It is composed of two smart objects, a desk (figure 1 and 3) and a dossier with six drawers (figure 4). Both objects have only behaviours of the kind depicted in figure 1.

In this way, the user stays in front of the computer screen, and can see its virtual hand being displaced accordingly to its real hand position. Depending on the position of the virtual hand, some hand clues are displayed, indicating that interactions can be selected.

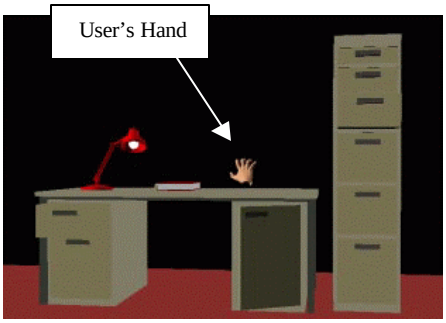

Figure 6. The virtual environment is composed of two smart objects where the user can move its virtual hand freely in order to explore available interactions.

The Cyber Touch data glove contains small special devices on the palm and on each finger, that can generate a variable vibration

sensation. This gives a total of six vibration devices. We use such vibrations to give two different kinds of feedback to the user:

- To indicate how many available behaviours are selectable. To do this, we send a small vibration to *n* vibration devices, where *n* is the number of hand clues that are currently being displayed.

- To indicate that an interaction was selected, by sending for a short period of time, a stronger vibration on all activated vibration devices.

This logic of using vibrations gives an interesting feedback to the user in cases where many close interactions exist. For example, in the first image of figure 4, the user feel 3 fingers vibrating, what makes him to pay more attention to his acts, in order to select the correct one. When he selects an interaction (second image of figure 4), he feels a stronger vibration, what makes him to visually confirm that the object acted accordingly. In the other hand, sometimes the excessive feeling of vibrations may be uncomfortable.

In this application example, no metaphor for navigating in large virtual environments was designed. Just the natural tracked hand position is used, what limits the interaction space to a rather small VE.

Also, in order to select a desired interaction, we consider only distances measured between the user's hand and the clue hands (figures 3 and 4). The shape of the user's virtual hand would be used only to distinguish between two hand clues that are too close one to another. This situation does not occur with the used objects.

In this way, the implemented application runs very fast, easily achieving interactive frame rates on an Impact SGI computer.

This prototype system permits an interesting analysis of the designed furniture regarding human factor aspects. Another direct application for this framework is training the use of complex equipments, by experiencing with them.

The objects used in this application have a simple functionality to open and close some of their parts, but our behavioural language can also be used to model a more complex logic. For example, manufacturers could provide a smart object description of their products together with the user's guide. In this way, the user could see all possible actions to perform with the equipment, virtually seeing what happens when, for instance, some button is pressed.

6. FINAL REMARKS AND FUTURE WORK

We presented in this paper a framework dedicated to model interactive objects that can be used in applications for direct interaction using VR devices.

Moreover, we present our first results towards a definition of a high-level interaction metaphor, naturally achieved from the smart object framework.

The important aspect of our approach is that smart objects are modelled in a general way that is independent of the application.

This introduces a way to have standard interactive object descriptions that can be used to describe many different types of objects. The key idea is that each object contains a complete description of its possible interactions; then its up to the application to interpret this description accordingly to its needs.

For example, users that do not have a VR system, could still interact with smart objects, by using other metaphors based on mouse navigation.

We believe that such approach of defining object's functionality by an interaction oriented description, provides a flexible framework that can be used by different interaction metaphors.

Users that have experienced the system showed that the interaction process is straightforward to learn and understand. However, the action of getting close to a hand clue was sometimes not so easy to perform without activating surrounding clues. But this factor is strongly related to the specific objects used in the application. In summary, we could see that the facility to activate the clues can be an advantage in some cases, but not in all cases, what suggests the use of variable thresholds.

As future work, we intend to incorporate two complementary modules to the application:

- A low-level physical model connected to a direct manipulation metaphor, in order to enable the user to displace objects, for example between the drawers of the smart objects.

- A navigation metaphor, to free the user from the real world space constraints, allowing navigation on large VEs and interaction of many different kinds of smart objects. Such environments have been successfully constructed in the scope of simulating a virtual city populated with virtual humans that can autonomously interact with smart objects [4].

7. ACKNOWLEDGEMENTS

The authors are grateful to Valery Tschopp for the aid on implementing the prototype VR application, to Dr. Tom Molet for the FOB use, and to Fabien Garat for appearing in the photo of figure 5. This research was supported by the Swiss National Foundation for Scientific Research and by the Brazilian National Council for Scientific and Technologic Development (CNPq).

8. REFERENCES

[1] G. Baciu, W. Wong, and H. Sun, "Hardware-Assisted Virtual Collisions", Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST, Taipei, Taiwan, 145-151, 1998.

[2] R. Boulic, S. Rezzonico, and D. Thalmann, "Multi Finger Manipulation of Virtual Objects", Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST, 67-74, 1996.

[3] K. Bowyer, S. Cameron, G. Jared, R. Martin, A. Middleditch, M. Sabin, and J. Woodwark, "Introducing Djinn – A Geometric Interface for Solid Modelling", Information Geometers, ISBN 1-874728-08-9, 24pp, 1995.

[4] N. Farenc, S. R. Musse, E. Schweiss, M. Kallmann, O. Aune, R. Boulic, and D. Thalmann, "A Paradigm for Controlling Virtual Humans in Urban Environment Simulations", Applied Artificial Intelligence Journal, 1999, to appear.

[5] C. Hand, "A Survey of 3D Interaction Techniques", Computer Graphics Forum 16(5), 269-281, 1997.

[6] M. Kallmann and D. Thalmann, "Modeling Objects for Interaction Tasks", Proceedings of the 9th Eurographics Workshop on Computer Animation and Simulation, 73-86, Lisbon, Portugal, 1998.

[7] M. Kallmann and D. Thalmann, "A Behavioural Interface to Simulate Agent-Object Interactions in Real Time",

Proceedings of Computer Animation 99, Geneva, May, 1999.

[8] Y. Kitamura, A. Yee, and F. Kishino, "A Sophisticated Manipulation Aid in a Virtual Environment using Dynamic Constraints among Object Faces", Presence, 7(5), 460-477, October, 1998.

[9] M. Mine, "Virtual Environment Interaction Techniques", UNC Chapel Hill Computer Science Technical Report TR95-018, 1995.

[10] M. Mine, F. P. Brooks Jr., and C. Sequin, "Moving Objects in Space Exploiting Proprioception in Virtual Environment interaction", Proceedings of SIGGRAPH'97, Los Angeles, CA, 1997.

[11] H. Nishino, K. Utsumiya, D. Kuraoka and K. Korida, "Interactive Two-Handed Gesture Interface in 3D Virtual Environments", Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST, Lausanne, Switzerland, 1-14, 1997.

[12] A. Paoluzzi, V. Pascucci, and M. Vicentino, "Geometric Programming: A Programming Approach to Geometric Design", ACM Transactions on Graphics, 14(3), 266-306, July, 1995.

[13] S. Parry-Barwick, and A. Bowyer, "Is the Features Interface Ready?", In "Directions in Geometric Computing", Ed. Martin R., Information Geometers Ltd, UK, Cap. 4, 129-160, 1993.

[14] A. Pentland, "Machine Understanding of Human Action", 7th International Forum on Frontier of Telecom Technology, Tokyo, Japan, 1995.

[15] V. Popescu, G. Burdea, and M. Bouzit, "VR Simulation Modelling for a Haptic Glove", Proceedings of Computer Animation 99, Geneva, May, 1999.

[16] I. Poupyrev, S. Weghorst, M. Billinghurst, and T. Ichikawa, "A Framework and Testbed for Studying Manipulation Techniques for Immersive VR", Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST, Lausanne, Switzerland, 21-28, 1997.

[17] J. Sauer and E. Schömer, "A Constraint-Based Approach to Rigid Body Dynamics for Virtual Reality Applications", Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST, Taipei, Taiwan, 153-161, 1998.

[18] K. M. Stanney, R. R. Mourant, and R. S. Kennedy, "Human Factors Issues in Virtual Environments: A Review of the Literature", Presence, 7(4), 327-351, August, 1998.

[19] D. J. Sturman, and D. Zeltzer, "A Survey of Glove-based Input", IEEE Computer Graphics and Applications, 30-39, January, 1994.

[20] The Virtual Reality Modelling Language, VRML, http://www.vrml.org.

[21] Y. Okada, K. Shinpo, Y. Tanaka and D. Thalmann, "Virtual Input Devices based on Motion Capture and Collision Detection", Proceedings of Computer Animation 99, Geneva, May, 1999.

[22] J. J. Gibson, "The theory of affordances", In R. Shaw & J. Brandsford (eds.), Perceiving Acting and Knowing. Hillsdale, NJ:Erlbaum, 1977.