A Formal Description of Multimodal Interaction Techniques for Immersive Virtual Reality Applications

David Navarre[1], Philippe Palanque[1], Rémi Bastide[1], Amélie Schyn[1], Marco Winckler[1], Luciana P. Nedel[2], and Carla M.D.S. Freitas[2]

[1] LIIHS-IRIT (Université Paul Sabatier),
118 route de Narbonne, 31062, Toulouse, France
{navarre, palanque, schyn, winckler}@irit.fr
[2] Informatics Institute, Federal University of Rio Grande do Sul,
Caixa Postal 15.064, CEP 91501-970, Porto Alegre, Brazil
{nedel, carla}@inf.ufrgs.br

Abstract. Nowadays, designers of Virtual Reality (VR) applications are faced with the choice of a large number of different input and output devices leading to a growing number of interaction techniques. Usually VR interaction techniques are described informally, based on the actions users can perform within the VR environment. At implementation time, such informal descriptions (made at design time) yield to ambiguous interpretations by the developers. In addition, informal descriptions make it difficult to foresee the impact throughout the application of a modification of the interaction techniques. This paper discusses the advantages of using a formal description technique (called ICO) to model interaction techniques and dialogues for VR applications. This notation is presented via a case study featuring an immersive VR application. The case study is then used to show, through analysis of models, how the formal notation can help to ensure the usability, reliability and efficiency of virtual reality systems.

1 Introduction

Virtual reality (VR) applications feature specificities compared to classical WIMP (Window, Icon, Menu and Pointers) interactive systems. WIMP interfaces may be considered as *static* (as the number of interactive widgets is usually known beforehand). Besides, they provide users with simple interaction techniques based on the use of the keyboard and/or the mouse, and the events produced (click, double click, etc.) are easy to manage. On the contrary, VR systems are based on 3D representations with complex interaction techniques (usually multimodal) where inputs and outputs can be very complex to manage due to the number of potential devices (data gloves, eye trackers, 3D mouse or trackball, force-feedback devices, stereovision, etc.). Designing or implementing VR applications require to address several issues like immersion, 3D visualisation, handling of multiple input and output devices, complex dialogue design, etc.

As for multimodal applications, when implementing VR applications, developers usually have to address hard to tackle issues such as parallelism of actions, actions sequencing or synchronization, fusion of information gathered from different input devices, combination or separation of information (fission mechanism) to be directed

M.F. Costabile and F. Paternò (Eds.): INTERACT 2005, LNCS 3585 , pp. 170 – 183, 2005.

to different devices. These issues make modelling and implementation of VR systems very complex mainly because it is difficult to describe and model how such different input events are connect to the application [21].

Many reports in the literature are devoted to invent new interaction techniques or describe software and hardware settings used in specific applications, most of them presenting also user studies for experimental evaluation. Empirical evaluation of interaction techniques for VR and 3D applications has been addressed recently [6, 7] as well as more general approaches addressing the usability of VR and multimodal interfaces [16, 18, 23]. Usually, interaction techniques for VR applications are described informally sequentially presenting the actions the users can perform within the VR environment and their results in terms of triggered events and modifications of objects appearance and/or location. Such informal descriptions make it difficult finding similarities between different techniques and often result in some basic techniques being "re-invented" [21]. Moreover, informal descriptions (due to their incomplete and ambiguous aspect) leave design choices to the developers resulting in undesired behaviour of the application or even unusable and inconsistent interaction techniques.

The growing number of devices available makes the design space of interaction techniques very large. The use of models has been proved to be an effective support for the development of interactive systems helping designers to decompose complex applications in smaller manageable parts. Formalisms have been used for the modelling of conventional interaction techniques [3, 8, 10, 15], and the benefits of using them for simulation and prototyping are well known [9].

Following the ARCH terminology, this paper presents the modelling of the dialogue part of VR applications and its relationship with the multimodal interaction of the presentation part. The modelling and implementation of the rendering techniques themselves are beyond the scope of this paper.

This paper aims at showing the benefits from using the ICO formalism to model interaction in virtual environments. It presents with details the impact of changing input devices and/or interaction techniques, detecting similarities and dissimilarities in the behaviours, and to allow measurements of the effects of these dissimilarities in the prediction of user performance. Both the interaction techniques and the dialogue part of the application are modelled using the ICO formalism [3], a formalism which was recently extended to support the modelling of multimodal and virtual reality applications. The case study shows the use of this formal notation for modelling a manipulation technique in an immersive virtual environment based on a chess game to allow a deeper discussion of formal notation advantages in the light of quantitative results obtained from experimental evaluation [16]. Such kind of application has also been used for customizing heuristic evaluation techniques for VR environments [1].

The paper is structured as follows. Section 2 informally presents the ICO formalism. The aim of that section is to present the basics of the formalism in order to allow the reader to understand the models presented in Section 4. We also emphasize the extension made on the ICO formalism in order to make it suitable for modelling interaction techniques of VR applications. Section 3 briefly describes the Virtual Chess case study while Section 4 presents its formal modelling using extended ICO. Section 5 is dedicated to related work.

2 Informal Description of ICO

The Interactive Cooperative Objects (ICO) formalism is a formal description technique designed to the specification, modelling and implementation of interactive systems [5]. It uses concepts borrowed from the object-oriented approach (i.e. dynamic instantiation, classification, encapsulation, inheritance, and client/server relationships) to describe the structural or static aspects of systems, and uses high-level Petri nets [12] to describe their dynamics or behavioural aspects.

In the ICO formalism, an object is an entity featuring five components: a cooperative object (CO), an available function, a presentation part and two functions (the activation function and the rendering function) that correspond to the link between the cooperative object and the presentation part.

The **Cooperative Object** (CO) models the behaviour of an ICO. It states (by means of a high-level Petri net) how the object reacts to external stimuli according to its inner state. As the tokens can hold values (such as references to other objects in the system), the Petri model used in the ICO formalism is called a high-level Petri Net. A Cooperative Object offers two kinds of services. The first one is called system devices and concerns to services offered to other objects of the system, while the second, event services, is related to services offered to a user (producing events) or to other component in the system but only through event-based communication. The availability of all the services in a CO (which depends on the internal state of the objects) is fully stated by the high-level Petri net.

The **presentation part** describes the external appearance of the ICOs. It is a set of widgets embedded into a set of windows. Each widget can be used for interacting with the interactive system (user interaction -> system) and/or as a way to display information about the internal state of the object (system -> user interaction).

The **activation function** (user inputs: user interaction -> system) links users' actions on the presentation part (for instance, a click using a mouse on a button) to event services.

The **rendering function** (system outputs: system -> user interaction) maintains the consistency between the internal state of the system and its external appearance by reflecting system states changes through functions calls.

Additionally, an **availability function** is provided to link a service to its corresponding transitions in the ICO, i.e., a service offered by an object will only be available if one of its related transitions in the Petri net is available.

An ICO model is fully executable, which gives the possibility to prototype and test an application before it is fully implemented [4]. The models can also be validated using analysis and proof tools developed within the Petri nets community and extended in order to take into account the specifications of the Petri net dialect used in the ICO formal description technique.

3 Informal Description of the Virtual Chess

The Virtual Chess application is inspired on the traditional chess game. It was originally developed as a testing ground application to support user testing of the selection of 3D objects in VR environments using two interaction techniques (*virtual hand* and

ray casting) [16]. The Virtual Chess is composed by a chessboard with 64 squares (cells) and contains 32 chess pieces. The interaction includes the manipulation (selecting, moving, releasing) of chess pieces and the selection of the view mode (plan view or perspective view).

The manipulation of pieces can be done either by using a classic mouse or a combination of data glove and motion capture device. When using a mouse the selection is done by first clicking on the piece and then clicking on the target position (x, y). We can replace the mouse by the data glove 5DT[1] and a motion captor[2] as the ones presented in Fig. 1.a. This data glove has a rotation and orientation sensor and five flexion sensors for the fingers. In this case, the motion captor is used to give the pointer position (x, y, z) while the fingers flexion is used to recognize the user gesture (closing the hand is recognized as a selection, opening the hand after a successful selection is recognized as a release). The selection of the view mode is done by pressing the key 0 (for the top view) or key 1 (for the perspective view) on a classic keyboard. In addition to these input devices, a user can wear stereoscopic glasses (see Fig. 1.b) in order to have a stereo experience. Fig. 1.c provides the general scenario for the user physical interaction with devices.

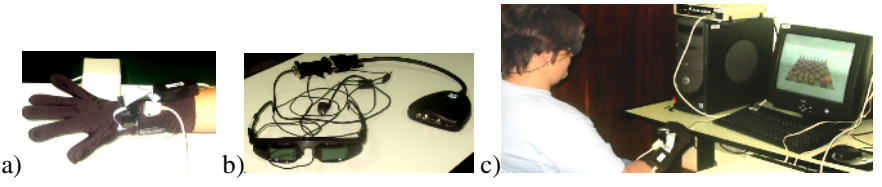

Fig. 1. Some of the devices employed: motion captor attached to a 5DT data glove (a); 3D stereoscopic glasses (b); scenario of user testing (c)

The users can move one piece at a time (horizontally, vertically and/or in diagonal). The Virtual Chess application does not take into account the game rules. All that the users can do are to **pick** a piece, **move** it to a new position and **drop** it. If a piece is dropped in the middle of two squares it is automatically moved to the closest square. Users cannot move pieces outside the chessboard but they can move pieces to a square occupied by another chessman.

In a real chess game, the movement of the pieces over the game board is performed with the hand. This has leaded to the implementation of the *virtual hand* interaction technique which represents the pointer position by a virtual 3D hand as shown in Fig. 2. Visual feedback is given by automatically suspending the selected piece over the chessboard and changing its colour (from grey or white to red).

Fig. 2 and Fig. 3.a show the chessboard in the perspective view mode while Fig. 3.b shows it in the top view mode (2D view).

[1] 5DT from Fifth Dimension Technologies (http://www.5dt.com/)
[2] Flocks of Birds from Ascension Technology (http://www.ascension-tech.com/)

Fig. 2. User interaction using direct manipulation (virtual hand technique) with visual feedback. From left to right: *picking*, *moving* and *dropping* a chessman.

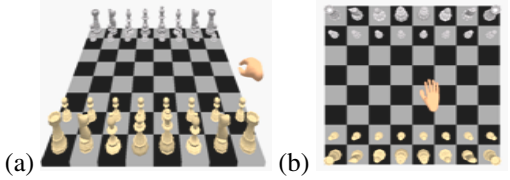

Fig. 3. View modes: (a) perspective view; (b) top view

4 Modelling the Virtual Chess with the ICO Formalism

As for other interactive systems, the modelling of VR applications must describe the behaviour of input and output devices, the general dialogue between the user and the application and the logical interaction provided by the interaction technique. Thus, modelling the Virtual Chess application was accomplished following steps 1 to 5 of the modified architecture Arch (the original architecture may be found in [2]) presented in Fig. 4. This model is useful for representing the various architectural components of an interactive application and the relationships between them. However, as the considered application is mainly interactive the left hand side of the Arch is not relevant. Section 4.1 discusses the modelling of steps 1 and 2, covering the treatment of low-level events and logical events from input devices. Section 4.2 describes the dialogue modelling of the Virtual Chess while Section 4.3 discusses the modelling of logical and concrete rendering.

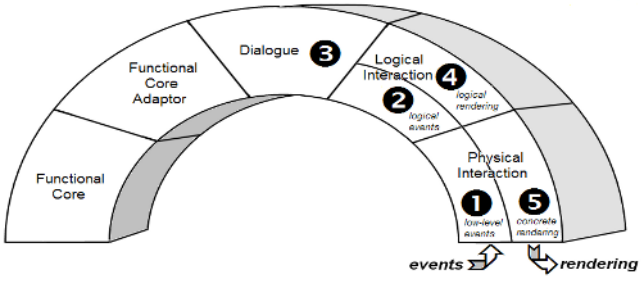

Fig. 4. The modified Arch architecture

4.1 Input Devices Modelling

The behaviour of our application is based on three main logical events: *pick(p), move(p)* and *drop(p)*, where *p* represents the piece being manipulated. In this section we present the different models which describe the way of physical inputs (actions performed by users on input devices) are treated in order to be used as logical events by the dialogue controller. At this point, we need one ICO model for each input device. Fig. 5, Fig. 6, and Fig. **7** present the ICO models describing the behaviour of the mouse, the coupling of motion captor and data glove and the keyboard, respectively.

When using a mouse, these so-called logical events are represented as a composition of the low-level events *move(x,y)* and *click(x,y)*, which are triggered by the physical mouse. Each time the user moves the mouse, a *move(x,y)* event is triggered and captured in the ICO by means of the *Move* service. A service is associated to one or more transitions having similar names in the ICO model; for example, in Fig. 5 the service *Move* is associated to the transitions *Move_1* and *Move_2*. Whatever the actual system state, a mouse's move action triggers a *move(x,y)* event causing a transition in the model.

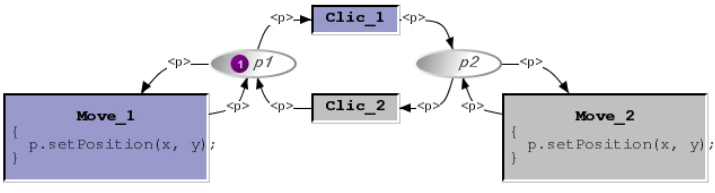

Fig. 5. Logical level behaviour for the Mouse

The logical events *pick(p)* and *drop(p)* are associated to the low-level event *click(x,y)* that is triggered by the physical mouse. The events *pick(p)* and *drop(p)* are determined by the sequence of low-level events (the first *click(x,y)* implies a *pick(p)*, the second *click(x,y)* implies a *drop(p)*, the third implies a *pick(x,y),* and so on). The incoming events the such as low events *click(x,y)* and *move(x,y)* are described by the Activation Function presented in Table 1.a while the triggered events are described by the Event Production Function presented in Table 1.b. Table 1.a and Table 1.b complete the model by showing the events activating the Petri Net presented in Fig. 5 and the events triggered to other models and/or devices.

Table 1. Event producer-consumer functions as described in Fig. 5.

a) Activation Function

Event Emitter	Interaction object	Event	Service
Mouse	None	*move(x,y)*	Move
Mouse	None	*click(x,y)*	Click

b) Event Production Function

Transition	Event produced
Move_1	*move(p)*
Move_2	*move(p)*
Clic_1	*pick(p)*
Clic_2	*drop(p)*

Figure 6 presents how the events *pick(p), move(p)* and *drop(p)* are produced when using the pair data glove and motion captor. Every time an event *idle()* is triggered, it enables the transition *init* to capture the current fingers' flexion from the data glove and the spatial hand's position from the motion captor. The information concerning to the flexion of the fingers and the position of the hand are stored on variables *g* and *p*, respectively. The event *idle* is produced in the internal loop implemented by graphic libraries, such as OpenGL, which was used to implement the Virtual Chess. The transitions *pick, notPick, drop* and *notDrop* compare the current and previous positions (which is given by the token from the place *last*). If the current position is different from the previous one, and the hand is opened, the system triggers an event *drop(p)* in the current hand position. If the hand is closed and its position is different from the previous one, then the system triggers an event *move(p)*.

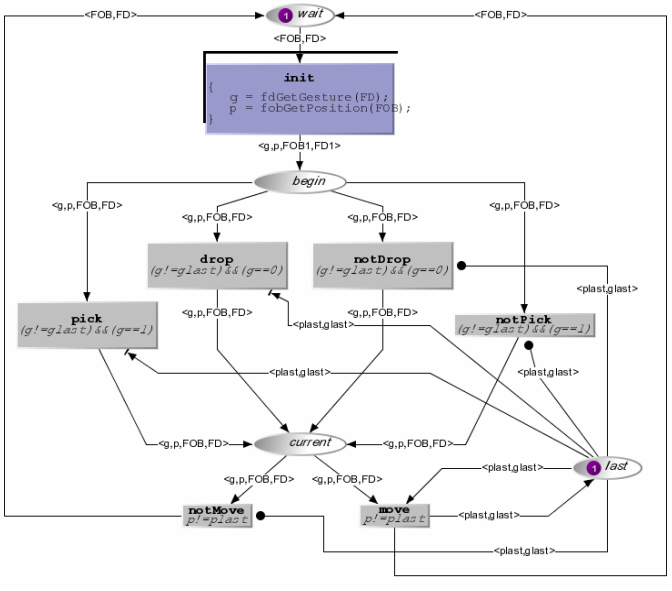

Fig. 6. Low-level behaviour (pick, move and drop) when using a data glove combined with a motion captor

Table 2 presents the list of incoming and triggered events in the model described in Fig. 6. In this model, the data sent back by the data glove and the motion captor can only be individually identified when comparing the current and the previous position.

Table 2. Event production-consumption functions as described in Fig. 6

a) Activation Function

Event Emitter	Interaction object	Event	Service
OpenGL loop	None	*idle*	init

b) Event Production Function

Transition	Event produced
drop	*drop(p)*
pick	*pick(p)*
move	*move(p)*

The role of the keyboard is to allow the users to choose the visualization mode (perspective or top view). The model that describes the keyboard behaviour is presented in Fig. 7. There are only two states available, each one corresponding to one of the pre-defined view modes (perspective or up). The incoming events in ICO for the keyboard are presented in Table 3; this modelling does not trigger any event.

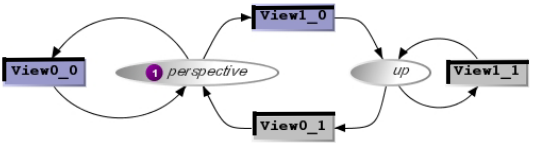

Fig. 7. Logical level modelling of the keyboard

Table 3. Activation Function as described in Fig. 7

Event Emitter	Interaction object	Event	Service
Keyboard	None	*keyPressed(0)*	View0
Keyboard	None	*keyPressed(1)*	View1

4.2 Dialogue Modelling

Independent from the input device employed (mouse or data glove and motion captor), the dialogue controller will receive the same events *pick(p), drop(p)* and

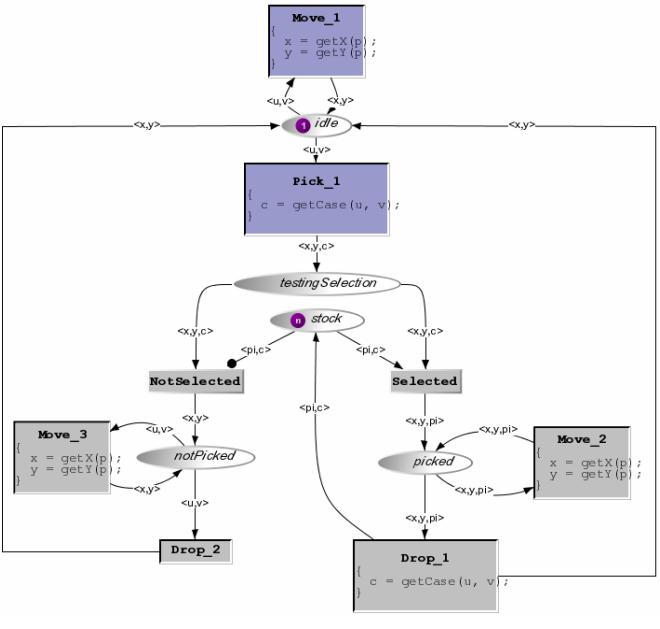

Fig. 8. Dialogue controller modelling

Table 4. Activation Function as described in Fig. 8

Event Emitter	Interaction object	Event	Service
Low-level events from mouse or the pair data glove plus motion captor	Chess piece *p*	*move(p)*	Move
	Chess piece *p*	*pick(p)*	Pick
	Chess piece *p*	*drop(p)*	Drop

move(p). As represented in Fig. 8, when an event *pick(p)* occurs (in the transition *Pick_1*) the square cell *c* corresponding to the position of the piece *p* is captured. If an event *pick(p)* occurs and the place *stock* contains a reference to square *c*, then the user can move the corresponding piece *p* (using the transition *Move_2*) or drop it (using the transition *Drop_1)*. Otherwise, the user can just move the hand over the chessboard for a while and then the system return to the initial state. This behaviour is also presented in Table 4.

4.3 Rendering and Interaction Technique Modelling

In this section we introduce the extensions to ICO formalism related to the *rendering events*. We include rendering events in the modelling whenever a change in the state of the system modifies something in the graphical display. We represent this by means of the *Rendering Function*. Table 5 describes the *Rendering Function* associated to the behaviour of the keyboard when selecting the visualization mode (see Fig. 7) and Table 6 presents the *Rendering Function* associated to the behaviour described in Fig. 8 for the dialogue controller. In these examples, the *rendering events* are triggered when entering into a place (a *token-enter* event) or leaving a place (a *token-out* event).

Table 5. Rendering Function associated to the keyboard modelling as described in Fig. 7

Place	Event	Rendering event
perspective	*token-enter*	*view(0)*
up	*token-enter*	*view(1)*

Table 6. Rendering functions associated to the behaviour described in Fig. 8.

Place	Event	Rendering event
idle	*token-enter*	*paintOpen(x,y)*
picked	*token-enter*	*paintClose(x,y,pi)*
notPicked	*token-enter*	*paintClose(x,y,null)*
stock	*token-enter*	*table(pi,c)*
	token-out	*hand(pi,c)*

The *rendering events* are comparable to other events except by the fact that they also notify the high-level ICO objects about changes in the presentation. This kind of events delegation to high-level ICO objects is required because we do not have all the information concerning to the rendering at the level where the input events were originally triggered.

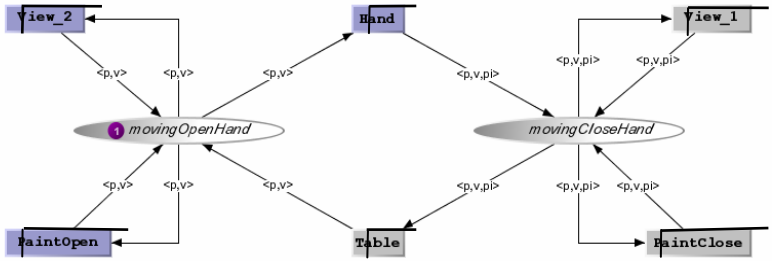

Fig. 9. General behaviour for the rendering

In order to provide a general understanding of how rendering events affect the graphical presentation, Fig. 9 presents another ICO model which describes how the Virtual Chess makes the fusion of events coming from other lower-level ICO models (describing the keyboard's behaviour as well as the mouse and/or the data glove and motion captor's behaviour).

Table 7 presents the activation function for the ICO model presented in Fig. 9. We can notice that the incoming events for that model correspond to rendering events triggered in lower level ICO models (i.e. keyboard and dialogue controller).

Table 7. Activation Function as described in Fig. 9

Event Emitter	Interaction objects	Events	Services
Low-level events from Fig. 7 (keyboard)	None	*view(0)*	View
	None	*view(1)*	View
Low-level events from Fig. 8 (dialogue controller)	None	*paintOpen(x,y)*	PaintOpen
	None	*paintClose(x,y,pi)*	PaintClose
	None	*paintClose(x,y,null)*	PaintClose
	None	*table(pi,c)*	Table
	None	*hand(pi,c)*	Hand

In Fig. 9, the incoming events are fused and translated into classical methods calls to the Virtual Chess application. In this example, each place and each transition is associated to a particular rendering (see Table 8 For example, when entering the place movingOpenHand the system calls the method for showing the open hand at the

Table 8. Rendering functions associated to the model presented in Fig. 9

a) Rendering triggered over places

Place	Event	Rendering
movingOpenHand	Token-enter	*paintOpenHand(v,p)*
movingCloseHand	Token-enter	*paintCloseHand(v,p,pi)*

b) Rendering triggered over transitions

Transition	Rendering
View_1	*changeView(v)*
View_2	*changeView(v)*
PaintOpen	*paintOpenHand(v,p)*
PaintClose	*paintCloseHand(v,p,pi)*
Table	*paintPiece(pi,c)*
Hand	*deletePiece(pi)*

position of piece p, while entering place movingCloseHand will cause the system calls for the method showing the closed hand hanging piece p (or null, if no piece was previously selected) at the p position (see Table 8.a). Similarly, when a transition is fired, the system calls the corresponding rendering method (see Table 8.b).

4.4 Dealing with Changes

In our case study, the physical rendering is done by means of a single 3D visualization display. However, we can easily extend our model to work with several output devices at a time just by replacing the method calls presented in Table 8 by other rendering methods (causing the fission of events) or other events captured by another ICO model describing output devices (in this case, one ICO model is required for each device). More information about the modelling of multimodal application using ICO formalism and how models can be interactively modified is available in the following papers [17, 3].

5 Discussion and Related Work

There are two main issues concerning the modelling of the VR applications: the use of a notation able to represent VR issues and the use of different devices. On one hand, we have extended the ICO notation in order to support the modelling of multimodal aspects such as fusion of several inputs, complex rendering outputs and 3D scenes. On the other hand, we have evaluated how changing input devices might require changes in the modelling and that ICO formalism makes these changes local to the concerned model thus lighten the burden of the designers.

Interaction in virtual environment or, more generally, 3D interaction can not be described using 'conventional' notations for interactive systems due to the inherent continuous aspect of information and devices manipulated in virtual reality applications. Actually, virtual environments are hybrid systems, and researchers in this field tried to extend their formalism to cope with a combination of discrete and continuous components [21]. Flownet [20, 21] is a notation to specify virtual environments interaction techniques using Petri Nets as the basis for modelling the discrete behaviour and elements from a notation for dynamics systems to model the continuous data flow in 3D interaction [25]. The same authors also proposed another formalism [21] so called HyNet (Hybrid High-level Petri Nets), that allows the description of hybrid interfaces by means of a graphical notation to define discrete and continuous concurrent behaviours, the availability of object oriented concepts and the high-level hierarchical description to specify complex systems. Jacob et al. [19] developed another visual hybrid formalism to describe interaction on VE. This formalism results in more compact specifications than HyNet, but the use of separate notations for the discrete and continuous parts makes the comprehension more difficult. More recently, Latoschik [14] introduced tATN (temporal Augmented Transition Network) as a mean to integrate and evaluate information in multimodal virtual reality interaction considering the use of speech and gesture in a VR application, and Dubois et al. [11] have proposed the ASUR notation to describe augmented reality systems in high-level.

The current paper does not address the issue of continuity because, even though the interaction and visualisation can be seen, at a higher level of abstraction, as continuous, when it comes to low level modelling the events produced and processed are always dealt with in a discrete manner. Indeed, both in the modelling and execution phases the explicit representation of continuous aspects was not needed.

VR applications and Multimodal systems have many aspects in common, such as parallelism of actions, actions sequencing or synchronization, fusion of information gathered through different devices to the combination or separation of information to be directed to different devices. In fact, description techniques devoted to the modelling of VR applications are similar to those employed to model multimodal applications.

As far as multimodal interaction is concerned, several proposals have been made in order to address the specific issue of formally describing various elements such as fusion and fission engines. For instance work from Hinckley [13] proposes the use of colored Petri nets for modelling two handed interaction by extending Buxton's work on Augmented Transition Networks [8]. Other work, based on process algebra such as CSP [22], Van Schooten [24] or LOTOS [10] have addressed (but only at a high level of abstraction) multimodal interactive systems modelling.

However, none of the approaches mentioned above are able to define a clear link between application and interaction. Besides, most of them do not have a precise semantics of the extensions proposed while the ICO formalism provides both a formal definition and a denotational semantics for each new construct (see the web site http://liihs.irit.fr/palanque/ICOs.htm). Last but not least, none of the approaches above are executable, i.e. provide a precise enough modelling power to allow for execution. This may not be a problem, as modelling can also be used for reasoning about the models, for instance in order to check whether or not some properties are valid on the models.

6 Conclusion and Future Work

In this paper, we have presented new extensions for the ICO in order to deal with complex rendering output as requested in VR applications. The ICO formalism has been previously extended and presented in [3, 17] to deal with the modelling of multimodal issues in interactive-system (e.g. event-based communication, temporal modelling and structuring mechanism based on transducers in order to deal with low level and higher lever events).

This paper has proposed a multi-level modelling approach for dealing with all the behavioural aspects of multimodal immersive interactive applications. We have shown how to deal with these issues from the very low level of input devices modelling, to the higher level of dialogue model for a 3D application. We presented how models can be gracefully modified in order to accommodate changes in the input devices and also in the interaction technique for such applications. Though relatively simple, the case study presented in the paper is complex enough to present in details all the aspects raised by the modelling of VR immersive applications and how the ICO formalism has been extended to tackle them.

This paper belongs to a long more ambitious research project dealing with the modelling of interactive applications in the field of safety critical application domains such as satellite control operation rooms and cockpits of military aircrafts. For these reasons the ICO formalism has been extended several times in order to address the specificities of such real time interactive applications.

Acknowledgements

The work presented in the paper is partly funded by French DGA under contract #00.70.624.00.470.75.96 and the R&T action IMAGES from CNES (National Centre on Space Studies in France) and CS Software Company.

References

1. Bach, C., Scapin, D. Adaptation of Ergonomic Criteria to Human-Virtual Environments Interactions. In: INTERACT 2003, Zurich. Amsterdam: IOS Press, (2003) 880-883
2. Bass, L., Pellegrino, R., Reed, S., Seacord, R., Sheppard, R., Szezur, M. R. The Arch model: Seeheim revisited. In: User Interface Developer's workshop version 1.0, (1991)
3. Bastide, R., Navarre, D., Palanque, P., Schyn, A., Dragicevic, P. A Model-Based Approach for Real-Time Embedded Multimodal Systems in Military Aircrafts. Sixth International Conference on Multimodal Interfaces (ICMI'04), Pennsylvania State University, USA. October 14-15, (2004)
4. Bastide, R., Navarre, D., Palanque, P. A Model-Based Tool for Interactive Prototyping of Highly Interactive Applications. In: ACM SIGCHI'2002 (Extended Abstracts) (2002) 516-517
5. Bastide, R., Palanque, P., Le Duc, H.; Muñoz, J. Integrating Rendering Specification into a Formalism for the Design of Interactive Systems. In: 5th Eurographics Workshop on Design, Specification and Verification of Interactive Systems (DSV-IS'98), Springer Verlag (1998)
6. Bowman, D., Johnson, D. B., Hodges, L. F. Testbed evaluation of virtual environments interaction techniques. In: ACM Symposium on Virtual Reality Software and Technology (1999) 26-33
7. Bowman, D., Kruijff, E., Laviola Jr., J. J., Poupyrev, I. An introduction to 3-D User Interface Design. Presence: Teleoperators and Virtual Environments, vol. 10, no. 1, (2001) 96-108
8. Buxton, W. A three-state model of graphical input. In: 3rd IFIP International Conference on Human-Computer Interaction, INTERACT'90, Cambridge, UK, 27-31 August (1990) 449-456
9. Campos, J. C., Harrison, M. D. Formally verifying interactive systems: A review. In Design, Specification and Verification of Interactive Systems '97, Springer Computer Science, (1997), 109-124
10. Coutaz, J., Paterno, P., Faconti, G., Nigay L., A comparison of Approaches for Specifying Multimodal Interactive Systems, In Proceedings of ERCIM, Nancy, France, (1993)
11. Dubois, E., Gray, P.D., Nigay, L., ASUR++: a Design Notation for Mobile Mixed Systems, IWC Journal, Special Issue on Mobile HCI, vol. 15, n. 4, (2003) 497-520,
12. Genrich, H. J. (1991) Predicte/Transiion Nets, in K. Jensen & G. Rozenberg (eds.), High-Level Petri: Theory and Applications, Springer Verlag, pp. 3-43.

13. Hinckley, K., Czerwinski, M. and Sinclair, M., Interaction and Modeling Techniques for Desktop Two-Handed Input. http://research.microsoft.com/users/kenh/papers/two-hand.pdf (1998)
14. Latoschik, M. E. Designing Transition Networks for Multimodal VR-Interactions Using a Markup Language. In: IEEE International Conference on Multimodal Interfaces (ICMI'02) Proceedings (2002)
15. Märtin, C. A method engineering framework for modeling and generating interactive applications. In: 3rd International Conference on Computer-Aided Design of User Interfaces, Belgium, (1999)
16. Nedel, L. P., Freitas, C. M. D. S., Jacob, L. J., Pimenta, M. S. Testing the Use of Egocentric Interactive Techniques in Immersive Virtual Environments. In IFIP TC 13 Conference INTERACT 2003, on Human Computer Interaction, Zurich. Amsterdam: IOS Press, (2003) 471-478
17. Palanque, P., Schyn, A. A Model-Based Approach for Engineering Multimodal Interactive Systems. In: IFIP TC 13 INTERACT 2003 conference, Zurich. Amsterdam: IOS Press (2003)
18. Poupyrev, I., Weghorst, S., Billinghurst, M., Ichikawa, T. Egocentric Object Manipulation in Virtual Environments: Empirical Evaluation of Interaction Techniques. Computer Graphics Forum, Eurographics'98 issue, Vol. 17, n. 3, (1998) 41-52
19. Jacob, R., Deligiannidis, L., Morrison, S. A software model and specification language for non-WIMP user interfaces, ACM ToCHI, v.6 n.1, p.1-46, March 1999.
20. Smith, S., Duke, D. Virtual environments as hybrid systems. In Eurographics UK 17th Annual Conference Proceedings, (1999) 113-128
21. Smith, S., Duke, D. The Hybrid World of Virtual Environments. In: Computer Graphics Forum, v. 18, n. 3, The Eurographics Association and Blackwell Publishers. (1999)
22. Smith, S. and Duke, D., Using CSP to specify interaction in virtual environment. In *Technical report YCS 321*, University ok York – Department of Computer Science (1999)
23. Sutcliffe, A., Gault, B., de Bruijn, O. Comparing Interaction in the Real World and CAVE virtual environments. In.: 18th HCI'2004. Leeds Metropolitan University, UK 6-10 September (2004)
24. Van Schooten, B. W., Donk, O. A., Zwiers, J. Modelling Interaction in Virtual Environments using Process Algebra In Proceedings TWLT 15: Interactions in Virtual Worlds, May 19-21, (1999).
25. Willans, J. Harrison, M. A toolset supported approach for designing and testing virtual environment interaction techniques. International Journal of Human-Computer Studies, Vol. 55, n.2, (2001) 145-165