Using Relations between Concepts during Interaction Modelling for Virtual Environments

Lode Vanacken[*] Joan De Boeck[†] Chris Raymaekers[‡] Karin Coninx[§]

Hasselt University
Expertise Centre for Digital Media
transnationale Universiteit Limburg
Wetenschapspark 2 B-3590 Diepenbeek (Belgium)

Abstract

Using semantic information when interacting in a virtual environment is still not a commonly used practice. In the rare cases where it is applied with success, the implementation however is mostly in an ad-hoc manner, incorporating the semantic information in the programming code so that it can be used in the interaction. In this paper, we discuss a possible approach on how the relationships between concepts (such as interpositions of objects) can be applied in a diagram describing the user interaction in an abstract way, with a minimal of coding effort and a maximum of reuse. Therefore, we use NiMMiT as a starting point. From this attempt we discuss the opportunities and the encountered problems of our approach in order to generalise the use of semantic information.

Index Terms: D.2.2 [Software engineering]: Design Tools and Techniques—User Interfaces; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Technique

1 Introduction

The creation of interactive virtual environments is a very technical and challenging task both with regard to the virtual world as well as the interaction available to the user. Usually, low-level programming languages are used to define the virtual world, the objects' behaviours and the interaction between the user and the environment. This is in particular true with 3D multimodal interfaces for virtual environments. Recently, conceptual modelling techniques have been proposed to design or prototype interactive virtual environments [5, 17]. This development process can be supported by advanced interactive tools. With respect to the support of virtual environment realization through conceptual modelling, research has been focusing on scene and interaction development [7, 13].

Through the usage of conceptual modelling for the construction of the virtual world, semantic knowledge can become available. This semantic knowledge is usually represented through ontologies and the information contained depends on the modelling approach [2, 9]. The usage of semantic knowledge is increasing in popularity due to the semantic web. Although the semantic web itself has not yet been fully realized, the supporting technologies (RDF, OWL, SPARQL, ...) are mature enough to be used for other applications. These technologies are nowadays finding their way to other applications [15, 12].

In our previous work on conceptual modelling of VEs [5] we investigated how the use of high-level specifications may help to ease the difficult process of interactive virtual environment creation: instead of coding an interactive virtual environment using a low level programming language, high-level models are used. Our model-based process combines a series of models containing a mixture of manual and automatic processes. We support our model-based process with tool support as it would not be very fertile without tool support.

Figure 1: Setup of the driving simulator case study.

In order to specify the interaction within an interactive virtual environment NiMMiT [6] has been developed. NiMMiT is a diagram based notation, we will discuss NiMMiT in section 2. NiMMiT has been integrated inside the model-based process and our tool support for this process. Some extensions to NiMMiT haven been introduced with regard to interaction evaluation [3] and contextual information [14]. We also introduced the usage of semantic information, using only concept information, during interaction modelling for virtual environments [16]. We will shortly discuss this in section 3. In section 4 we will propose an approach to also introduce the relationships between concepts during interaction modelling in NiMMiT. Afterwards we discuss several problems of this approach and conclude with future work directions.

2 Interaction Modelling: NiMMiT

In this section we will discuss NiMMiT (**N**otat**i**on for **M**ulti**M**odal **i**nteraction **T**echniques), a diagram based notation intended to describe multimodal interaction between a human and a computer, with the intention to automatically execute the designed diagrams.

Several other high level notations exist for designing interaction, some of them are state driven (ICO [13]) while others use a data flow architecture (InTml [7]).

In the remainder of this section, we shortly describe the primitives of our notation and will discuss a simple abstract example. For a more detailed description of the basic concepts of NiMMiT, we refer to [6].

2.1 NiMMiT Primitives

In NiMMiT, interaction with the computer is considered *event-driven*: users initiate an (inter)action by their behaviour, which invokes events into the system. These events can be triggered by different modalities, such as speech recognition, an action with a pointing device, or a gesture. Interaction is also *state-driven*, which means that not in all cases the system responds to all events. The

[*]e-mail: lode.vanacken@uhasselt.be
[†]e-mail: joan.deboeck@uhasselt.be
[‡]e-mail: chris.raymaekers@uhasselt.be
[§]e-mail: karin.coninx@uhasselt.be

response to an event, can bring the interaction in another state, responding to other events. Being *data-driven* is another important property of the notation. It is possible that data needs to be shared between several states of the interaction. For example, a subtask of the interaction can provide data, which has to be used in a later phase of the interaction (e.g. touching an object to push it). Finally, an interaction technique can consist of several smaller building blocks, which can be considered as interaction techniques themselves. Therefore, *hierarchical reuse* should be possible within the notation.

Taking the aforementioned considerations into account, NiMMiT defines the following basic primitives: states, events, task chains, tasks, labels and state transitions. All these primitives are present in figure 2.

State: A state is depicted as a circle. The interaction technique starts in the start-state, and ends with the end-state. A state defines a set of events to which the system responds.

Event: An event is generated by the framework, based upon the user's input. A combination of events can be multimodal, containing actions such as speech recognition, gestures, pointing device events and button clicks. A single event or a specific combination always triggers the execution of a task chain.

Task Chain: A task chain is a linear succession of tasks, which will be executed one after the other. For example in figure 2 'Taskchain1' will execute 'Task1' and 'Task3'.

Task: A task is a basic building block of the actual execution of the interaction technique. Typically, tasks access or alter the internal state of the application. E.g. when running in a typical 3D environment, a task can be 'collision detection', 'moving objects', 'playing audio feedback', etc. Tasks can be predefined by the system, but designers can define their own custom tasks, as well. All tasks can have input and output ports, on which they receive or send parameters or result values. Input ports are required or optional, indicated by a square or circle input port respectively.

Labels: As data can be shared throughout a diagram, NiMMiT needs a system to (temporarily) store values. This is done in 'labels', which can be seen as high level variables (In figure 2 'OutputT3').

State Transitions: Finally, when a task chain has been executed completely, a state transition moves the diagram into the next state. A choice between multiple state transitions is also possible, based upon the value of a certain label.

2.2 Example

By means of figure 2 we will give an brief overview of how the NiMMiT notation should be interpreted. We have chosen an abstract example for illustrative purpose. The start-state of this diagram responds to 4 different events (called EVENT1 to EVENT4). When 'EVENT1' *or* 'EVENT3' is fired, 'Taskchain1' will be invoked. 'Taskchain2' however will only be invoked if 'EVENT2' *and* 'EVENT4' occur at the same time, which is defined by the melting pot principle [4].

When a task chain is invoked, all tasks within the chain are executed one after the other (from top to bottom) using each others output when necessary. The output of a task can be stored in a label in order to be used by a task in an other task chain. In the example the evaluation of 'Taskchain1' will trigger the execution of 'Task1' and 'Task3' of which the last task, 'Task3', results in a boolean value that will be stored in the label 'OutputT3'

When all tasks in the chain are successfully executed, the next state is determined based on the exitlabel of the task chain. In 'Taskchain1' no exitlabel is defined so we return to the 'Start'-state waiting for new events to be fired.

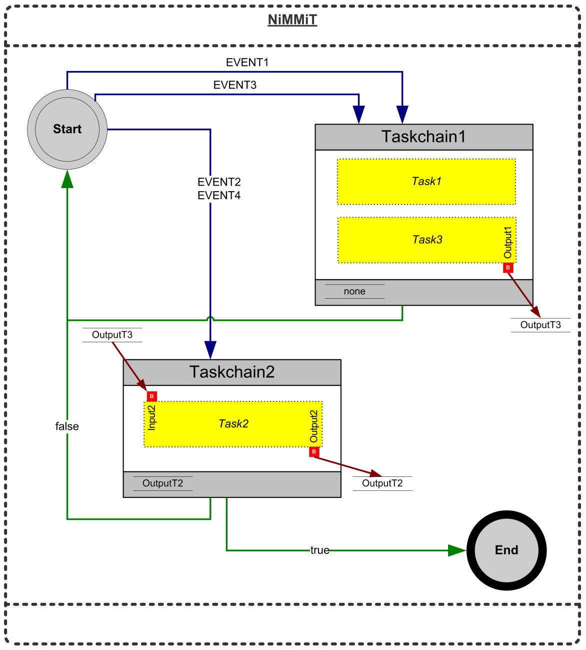

Figure 2: An abstract NiMMiT diagram.

As indicated, 'Taskchain2' will only be executed if 'EVENT2' *and* 'EVENT4' are fired simultaneously. During the execution of 'Taskchain2', the output of 'Task3' (stored in the label 'OutputT3') is used as input for 'Task2', which again results is a boolean value (stored in label 'OutputT2'). Since 'OuputT2' is used as exitlabel for this task chain, the result of 'Task2' will determine the next state of execution: if the value in 'OutputT2' is false, the next state will again be the 'Start'-state; if however, the result of 'Task2' is true, the 'End'-state is reached and the execution of the interaction finishes.

3 Semantic Information during Interaction Modelling of VEs

Semantic information in Virtual Environments has already been used by several researchers [10, 8, 1]. They primarily focused on integrating this semantic information within a framework for virtual environment development while we focus on integrating semantic information inside our conceptual modelling phase.

In the remainder of this section we will shortly discuss our introduction of conceptual information into NiMMiT [16].

3.1 Concepts as Semantic Information

In order to make use of conceptual data available in the ontology, describing the virtual scene, NiMMiT can represent this data as a datatype (Concept). This data can be used during interaction modelling in tasks. For example the 'GetObjects' task filters object(s) with regard to concept(s) and the 'IsOfConcept' task checks if object(s) are of certain concept(s).

In order to show the flexibility of introducing concepts into NiMMiT we will shortly elaborate on two examples in the remainder of this section.

3.1.1 Example: Selecting Objects

In most virtual environment applications not all objects are dynamic and interactive. To make only a certain subset of all objects interactive, a lot of ad-hoc, hard-coded solutions exist in which code

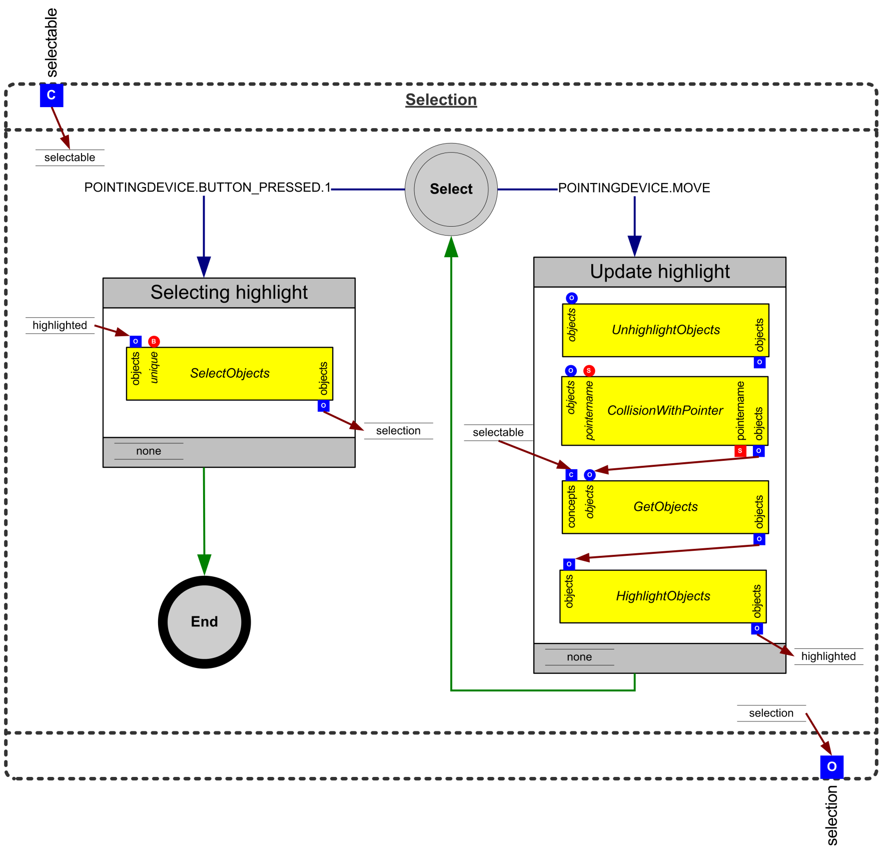

Figure 3: The selection tasks which only selects objects of certain concept(s).

is written such that the application knows which objects are interactive. In most situations the user first needs to select objects to interact with, therefore the easiest solution for the designer is to make only the interactive objects selectable.

In figure 3, a NiMMiT schema is presented which makes it only possible to select objects which are of concept(s) represented by the 'selectable'-label which receives its values from the input port of the interaction technique. Independent of the design, it is easy to allow all objects to be selected by attributing the root concept value (e.g. 'Thing') to the input port of the interaction technique. Remark that concepts are organized in a hierarchical manner, which makes it possible to pick a parent-concept instead of all children concepts individually. Also, note that this allows for the ontology to not be specifically designed for interaction, Irawati et al. [8] introduce concepts as 'Interactive_Object' from which all other interactive concepts are derived which would allow the system to search for only objects of the concept 'Interactive_Object'. Our approach gives more flexibility as any concept(s) can be selected at runtime without having to belong to a certain type of objects which has been decided upon during the coding of the system. This also means that the semantic selection technique can be used in all applications and thus is application independent.

It is important to remark that when multiple objects are highlighted this interaction technique highlights those that belong to the correct concept(s). If we would like the interaction technique to be more exclusive and only let it allow to complete successfully when all objects being collided with the pointing device belong to the concept(s) indicated in the 'selectable'-label, then only minimal changes are necessary such as using the 'IsOfConcept' task and adding a new state and task chain.

3.1.2 Driving Simulator

Another example study, that is more close to a realistic application, is a driving simulator in which a steering wheel (Logitech G25) is integrated. In this application (see figure 1), we need to know the surface which we are driving on such that realistic force feedback can be given through the steering wheel. The original designer of the driving interaction at first hard-coded the objects to collide with and to check which object, such as a road or grass, we are driving on.

Concepts are used here to define on which surface the car is driving and which feedback has to be returned to the driver, more information on this case study can be found in [16].

4 Using Semantic Relations Between Concepts

During the conceptual modelling phase of interactive virtual environments it is also interesting to use relationships between concepts. For example in a certain virtual environment application a user wants to place lampposts alongside a road, one could try to use the fact that there exists a relationship between the concepts lamppost and road (namely 'alongside'). This relationship indicates that a lamppost resides alongside a road (wrt. the current application) and we could help the user in placing a lamppost alongside a road by giving multimodal feedback (e.g. force enabled snapping locations). Note that this might not be the only possible relationship, the lamppost could also have a relation, 'on', with the ground.

It is not straightforward to introduce relations during the interaction modelling such that the solution is generic. We will first introduce a possible approach and afterwards discuss problems which have to be solved to achieve a generic solution.

In a first approach we try to create a manipulation technique, that similarly as the selection technique from section 3.1.1 is an application independent manipulation technique. There are currently two possible strategies used to create a manipulation technique. One possibility is to create a manipulation technique per sets of concepts that need different handling of the manipulation. Another possibility is to create one manipulation technique that on its own checks which concept it is manipulating and depending on that changes its way of manipulating. The first possibility makes the designer move the enabling of the manipulation technique to a higher level (the dialog model of the model-based design approach) and might overcomplicate this. The second one does not contaminate the dialog model but might become unmanageable to design or maintain. We believe that different types of manipulation can be directly linked to the different relations, so therefore using these relations inside NiMMiT and the manipulation technique could form a more generic solution.

In figure 4, an application independent manipulation technique is depicted. The input port 'selected' contains the object which is currently selected and thus will be manipulated. This data is stored in the 'selected'-label and will be used during manipulation. If the user moves the pointing device an event 'MOVE' will be fired and the corresponding task chain ('Manipulate') will be executed. The first tasks, 'GetPointerPosOriDelta' returns the delta movement which will be used in a new predefined task 'MoveUsingRelations'. This tasks will check the semantic world for possible relations between the currently selected object and other objects and execute the according relations. These relations can be defined through programming code or LUA scripting. Using this new task 'MoveUsingRelations' it is possible to use one manipulation task which autonomously implements the manipulation by moving the possible manipulation to the relations. With good tool support this will help the designer not to forget about certain possible relationships between concepts which are important for the application. Finally, if the user decides to stop manipulating the object, a button press will fire another task chain ('Drop Object') which drops (deselects) the object and ends the manipulation technique.

During the design of the relations the designer can use the hierarchical representation of an ontology. The designer can first implement a certain relation, which can hold between all the concepts in the hierarchy, for a parent concept and if a specialisation is needed implement it for a certain child concept. For example, the following hierarchy: Object - Furniture - Table - (RegularTable, DesignTable, etc.). The designer can first implement the relation 'on' for Furniture and use that design for a Table and if necessary further specify it for other types of tables, such as DesignTable which might have a completely different structure than a general table represented by the parent concept. Note that a general table will be represented by a RegularTable because of the fact that we can not specify this in a property of Table as RegularTable or DesignTable could also have children.

We have not yet formalized completely how we will define relations to use them during manipulations, in the following section we will give an overview of problems we currently encounter.

4.1 Discussion

In previous work, as discussed in section 3, we successfully introduced concepts during interaction modelling inside our model-based design approach. But the introduction of relations, during interaction modelling, brings out some points for discussion.

An ideal relation definition would be generic to such a level that it is application independent and can be used in any newly designed application. For example defining the relation 'onTopOf' might seem simple at first, constrain an object such that it remains on top of another object. But what is the *top* of an object, how do we define this? Do we always want that 'onTopOf' means that an object *remains* on top of another object? Is it also possible that this

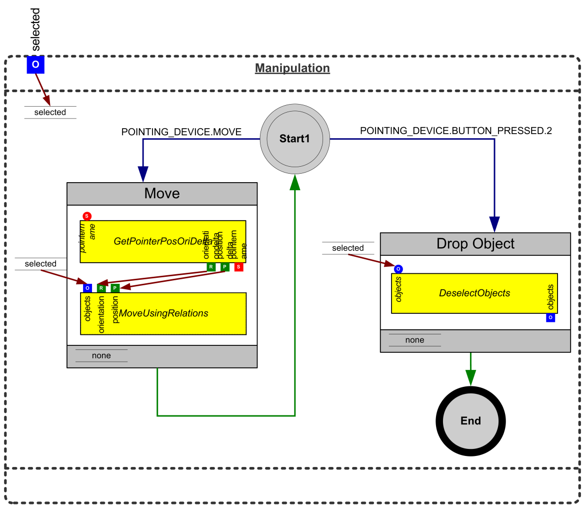

Figure 4: A semantic manipulation technique

relation can end and thus the object is not on top of another object anymore, this also brings us to the problem when do we want a relationship to exist? All these questions seem to not have one such answer that makes it possible to implement a relationship in such a way that it becomes application independent.

Imagine that we would succeed in defining a mechanism to define a relation application independent, we also have to take into account several other relationships which might exist at the same time, how would they work together, could we end up in a local minima where one constraint cannot be completely satisfied, because another limitation strives for another optimal value.

A problem with not being able to define relations application independent might induce a complex web of relation definitions. If a certain relation can be used several times between different concepts which have a different meaning inside the same application, that would oblige the designer to define the relation on a concept-relation-concept level instead of only the relation level. Off course without extensive testing in the form of case studies this is hard to foresee.

Besides using relations for manipulation techniques they could also be used for behaviour modelling such that objects can behave autonomously according to the existing relations similarly as in the work of Lugrin et al. [11] in which action representations, grounding and common sense are used to make knowledge based system work together with computer graphics systems.

5 Conclusion

In this paper we discussed our attempt at introducing relationships defined in the semantic information of the interactive virtual environment application. This work has been inspired by previous work on introducing semantic information during conceptual modelling of virtual environments. We presented concepts in NiMMiT using a new datatype and predefined tasks. We introduced a generic manipulation technique using relationships and discussed remaining problems we currently encounter. In the future we planned to further investigate these problems and try to design our system into a fully flexible framework using all semantic information available.

Acknowledgements

Part of the research at EDM is funded by ERDF (European Regional Development Fund), the Flemish Government and the Flemish Interdisciplinary institute for BroadBand technology (IBBT).

The VR-DeMo project (IWT 030248) is directly funded by the IWT, a Flemish subsidy organization.

REFERENCES

[1] T. Abacý, J. Cýger, and D. Thalmann. Action semantics in smart objects. In *Workshop towards Semantic Virtual Environments (SVE 2005)*, pages 1–7, 2005.

[2] W. Bille, B. Pellens, F. Kleinermann, and O. De Troyer. Intelligent modelling of virtual worlds using domain ontologies. In *Proceedings of WIC*, pages 272 – 279, Mexico City, Mexico, 2004.

[3] K. Coninx, E. Cuppens, J. De Boeck, and C. Raymaekers. Integrating support for usability evaluation into high level interaction descriptions with NiMMiT. In *Proceedings of DSVIS'06*, volume 4385, Dublin, Ireland, July 26–28 2006.

[4] J. Coutaz, L. Nigay, D. Salber, A. Blandford, J. May, and R. M. Young. Four easy pieces for assessing the usability of multimodal interaction: The CARE properties. In *Proceedings of INTERACT95*, pages 115–120, Lillehammer, June 1995.

[5] J. De Boeck, C. Raymaekers, and K. Coninx. A tool supporting model based user interface design in 3d virtual environments. In *Proceedings of GRAPP08*, Funchal, Portugal, January 22–25 2008.

[6] J. De Boeck, D. Vanacken, C. Raymaekers, and K. Coninx. High-level modeling of multimodal interaction techniques using nimmit. *Journal of Virtual Reality and Broadcasting*, 4(2), September 2007. `urn:nbn:de:0009-6-11615`.

[7] P. Figueroa, M. Green, and H. J. Hoover. InTml: A description language for VR applications. In *Proceedings of Web3D'02*, pages 53–58, Arizona, USA, Februari 2002.

[8] S. Irawati, D. Calderón, and H. Ko. Spatial ontology for semantic integration in 3d multimodal interaction framework. In *ACM Int. Conf. on VRCIA*, pages 129–135, 2006.

[9] E. Kalogerakis, S. Christodoulakis, and N. Moumoutzis. Coupling ontologies with graphics content for knowledge driven visualization. In *Proceedings of the IEEE VR 2006*, pages 43–50, 2006.

[10] M. E. Latoschik and C. Fröhlich. Towards intelligent vr - multi-layered semantic reflection for intelligent virtual environments. In *Proceedings of GRAPP07*, pages 249–260, 2007.

[11] J.-L. Lugrin and M. Cavazza. Making sense of virtual environments: action representation, grounding and common sense. In *Proceedings of IUI'07*, pages 225–234, 2007.

[12] D. S. McCorkle and K. M. Bryden. Using the semantic web technologies in virtual engineering tools to create extensible interfaces. *Virtual Reality*, 11(4):253–260, 2007.

[13] D. Navarre, P. Palanque, R. Bastide, A. Schyn, M. Winckler, L. Nedel, and C. Freitas. A formal description of multimodal interaction techniques for immersive virtual reality applications. In *Proceedings of IFIP TC13 Int. Conf. on HCI*, 2005.

[14] L. Vanacken, E. Cuppens, T. Clerckx, and K. Coninx. Extending a dialog model with contextual knowledge. In *TAMODIA*, volume 4849 of *LNCS*, pages 28–41, 2007.

[15] L. Vanacken, C. Raymaekers, and K. Coninx. Automatic speech grammar generation during conceptual modelling of virtual environments. In *Intuition 2007*, Athens, Greece, Octobre 2007.

[16] L. Vanacken, C. Raymaekers, and K. Coninx. Introducing semantic information during conceptual modelling of interaction for virtual environments. In *Proceedings of WMISI'07*, pages 17–24, 2007.

[17] J. Willans and M. Harrison. A toolset supported approach for designing and testing virtual environment interaction techniques. *International Journal of Human-Computer Studies*, 55(2):145–165, August 2001.