High-Level Descriptions for Multimodal Interaction in Virtual Environments

Chris Raymaekers
chris.raymaekers@uhasselt.be

Lode Vanacken
lode.vanacken@uhasselt.be

Joan De Boeck
joan.deboeck@uhasselt.be

Karin Coninx
karin.coninx@uhasselt.be

Hasselt University
Expertise Centre for Digital Media
and transnationale Universiteit
Limburg
Wetenschapspark 2,
3590 Diepenbeek, Belgium

CHI 2008, April 5 –10, 2008, Florence, Italy
ACM 1-xxxxxxxxxxxxxxxxx.

Abstract
Designing non-traditional user interfaces is a challenging task for designers. NiMMiT, a high level description for 3D multimodal interaction in virtual environments, provides a means to design, prototype or communicate about interaction techniques. The focus is on making it possible for designers to create new interaction techniques while lowering implementation efforts.

Keywords
Multimodal, Interaction techniques, high-level descriptions

Introduction
User interface design for non-WIMP interfaces is a real challenge; tools for the creation of such interfaces are rather seldom than common. A 3D (multimodal) user interface for a virtual environment not only consists of traditional WIMP interface elements. The user also needs to be able to interact with the virtual environment such that he/she can navigate the virtual world or select and manipulate virtual objects.

In order to facilitate the design of these multimodal interaction techniques high-level descriptions have been developed, such as InTml, ICO and NiMMiT.

In the following sections we will discuss NiMMiT, our own high-level description, followed by an example. Problems and drawbacks of our notation are thereafter elaborated on.

NiMMiT

NiMMiT, Notation for MultiModal Interaction Techniques, is a graphical notation, inheriting the formalism of a state-chart in order to describe multimodal interaction within virtual environments. Furthermore, it also supports dataflow which is important in the user interaction, as well. A more detailed description of NiMMiT can be found in [1]. We shortly describe the most important primitives of NiMMiT. An example of a NiMMiT diagram can be seen in figure 1.

NiMMiT is basically a state chart, in which a state (represented as a circle) indicates the possible events the user can provide and to which the application listens.

An event is an action a user can perform, such as moving a pointing device, speaking a command, clicking a button, etc. When an event or a combination of events has occurred, the associated arrow, points to a task-chain (big rectangles) that is to be executed.

A task-chain is a linear succession of tasks that are executed one after the other. When a task-chain is finished, a state-transition occurs (light, green, arrow) bringing the interaction into a new state, responding to another set of events.

A task (smaller rectangle in a task-chain) is a set of actions defined to `reach a goal'. Tasks are mostly predefined, such as querying device positions and calculating collisions, in order for the designer to easily pick them from a list. For specialised actions, however, custom tasks can be written either using LUA script or C++ code.

NiMMiT also supports dataflow between different tasks. Labels (high level variables) are used to save output from a task (output ports are depicted as small squares at the bottom right of the task symbol), or to provide input to a task (input ports are depicted at the top-left of a task).

In order to support the evaluation of interaction techniques, NiMMiT employs a mechanism, called probes. (not depicted in figure 1) These allow measuring the different states, transitions and labels at different moments during the interaction [2]. This information can further be filtered and logged to disk. Recently, we also added support for contextual and semantic information [3]. Information from an ontology, describing the virtual world, can be used with this extension of NiMMiT to define extra constraints within an interaction technique. For example, with an interaction for opening a door, one can specify in the NiMMiT diagram that the object being manipulated must be a door, without having to hard-code this constraint. By changing the constraint, we can for instance also use this interaction technique for opening windows.

Example

An example of NiMMiT diagram representing a grab metaphor is depicted in figure 1. In this interaction technique the user first selects the object through a selection technique. Afterwards the object moves according to the movements of the pointing device. If

the user is satisfied with the new position, the object can be released with a button press.

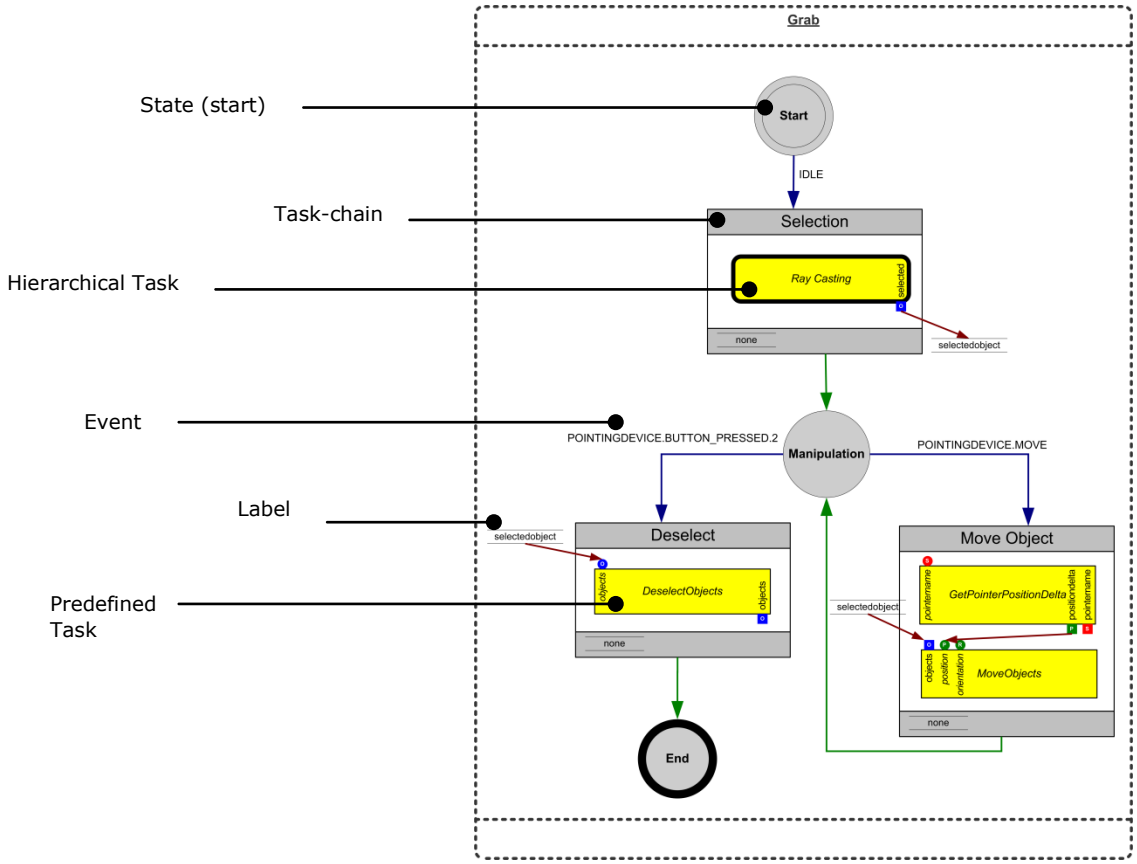

Figure 1 An example of a NiMMiT Diagram. This diagram represents a grab Interaction Technique.

The start state (*Start*) uses the *idle* event, which is fired immediately. This triggers the *Selection* task-chain which performs a selection technique namely *Ray Casting*, which in itself is defined by a NiMMiT diagram. The result is stored in the label *selectedobject*. After execution of the task-chain a state transition is performed to the *Manipulation* state. This state listens to two events, a button press and a move event from a pointing device. The move event triggers the *Move Object* task-chain which moves the selected object according to the movement of the pointer device. If button *2* is pressed the *Deselect* task-chain is executed which deselects the selected object and ends the interaction technique.

Strengths

NiMMiT allows the designer to communicate about the functionality of an interaction technique through an easy-to-read diagram which can be the basis for exploratory prototyping (with other stakeholders).

NiMMiT diagrams can be created using a tool, called CoGenIVE, which can save the diagrams in an xml-format. As this format can be interpreted by our application framework for virtual environments, we can support prototyping of multimodal interaction techniques.

The combination of state charts and the dataflow mechanism in NiMMiT works very well for designing interaction, typically during interaction the user reaches different interaction states and produces data as a result of interacting which sometimes has to be transferred for further usage.

Finally, hierarchical reuse of interaction techniques is supported. An interaction technique can function as a task, as well, allowing to reuse earlier developed interaction techniques.

Problems and Drawbacks

Experience from using NiMMiT has brought up some problems and drawbacks. We will discuss them in this section.

In the example of figure 1 the user's first task is to perform a selection of an object. If the user does not succeed in selecting an object, a rollback mechanism is executed to undo the steps already taken. For this purpose, every task contains information needed to be able to 'unperform' itself, but unfortunately this is not always very straightforward. For example if the user deleted an object during a physical simulation, the entire simulation would have to move back in time. Alternatively, the designer could indicate how to handle the errors or introduce a cancellation process, but this would increases the design complexity.

For complex interaction techniques, *state explosion*, a common problem in state diagrams, can occur. A solution for this problem is not straightforward and we are exploring the possibility of using preconditions at a task-chain or event level.

Another aspect that needs more attention is the incorporation of output modalities, such as haptics and audio. For now, these are always added in a diagram through the addition of a custom task which is scripted or coded. Thus, if a designer would like to add a certain force feedback effect during a manipulation interaction technique, a custom task has to be designed which creates the appropriate force feedback. A better approach of adding such types of feedback is necessary as visual feedback is currently better covered by predefined tasks such as 'highlightobjects'.

Status and Future Work

In our current framework and model based development process NiMMiT is continuously being used. The tool CoGenIVE makes it possible to design and execute NiMMiT diagrams interactively. In the future we would like to concentrate on the problems and drawbacks discussed earlier, in particular how to solve the state explosion problem elegantly and intuitive error handling. The integration of semantic information in NiMMiT seems to be a valuable feature. It might become even more beneficial if automatic and dynamic coupling to predefined tasks exists instead of having to use predefined tasks to introduce semantics.

Acknowledgements

Part of the research at the Expertise Centre for Digital Media is funded by the 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] De Boeck, J., Vanacken, D., Raymaekers, C. and Coninx, K. High-Level Modeling of Multimodal Interaction Techniques Using NiMMiT, Journal of Virtual Reality and Broadcasting, 4:2, non-periodical, ISSN 1860-2037.

[2] Coninx, K., Cuppens, E., De Boeck, J. and Raymaekers C. Integrating support for usability evaluation into high level interaction descriptions with NiMMiT, DSVIS 2006, LNCS, volume 4323, pp. 95-108

[3] Vanacken, L., Raymaekers, C. and Coninx, K. Introducing Semantic Information during Conceptual Modelling of Interaction for Virtual Environments, MISI 2007 (WS at ICMI 2007) (in press).