An Intelligent Interface for a Virtual Environment*

Britta Lenzmann, Ipke Wachsmuth, Yong Cao

University of Bielefeld,
Faculty of Technology,
D-33501 Bielefeld, Germany
{britta,ipke,yong}@techfak.uni-bielefeld.de

Abstract

This paper describes an application of a multiagent system for interaction with a virtual environment carried out in the VIENA project. The VIENA project has the following goals: enable an intelligent communication with a virtual environment, develop a multiagent system as a mediator in human-computer interaction, and explore its usefulness for interactive design and exploration. Instead of using the mouse and menus to manipulate scene details in a virtual environment we communicate with the system by way of natural language. Our aim is to keep the user (designer) free from technical considerations such as planning of geometric details, etc. Taking special responsibilities in mediating verbal instructions, the VIENA agents cooperate with each other and with the user to offer a goal scene meeting the user's wants. We explain the VIENA multiagent system, give an example application in virtual design, and discuss the specific advantages we found in using agent techniques for building an intelligent interface.

1 Introduction

With immensely growing capabilities of information processing systems, the complexity of their features also places increased demand on users. For example, operating a high-end computer graphics design tool often requires complex manual and intellectual skill in using mouse and a multitude of menus. Thus, the design of more intelligent user interfaces has become a core issue.

*Research in the VIENA Project is partly supported by the Ministry of Science and Research of the Federal State North-Rhine-Westphalia under grant no. IVA3-107 007 93. The authors are indebted to Majid Amanzadeh, Dirk Evers, Stefan Fischer Rivera, Martin Fröhlich, Carla Intrup, Tanja Jörding, Karsten Otto, and Norbert Siekmann for assisting during the research.

One idea is to develop specialized systems which act as "interface agents" between the user and the application system [10]. An example of this idea are agents which can accumulate knowledge about tasks, habits, and preferences of their users to act on their behalf [12]. The idea of interface agents has great prominence in visions of future user interfaces [14], but it has also found critique with respect to privacy of user data [13, 15].

In our work we pursue a view of interface agents which use knowledge of the application itself and add comfort in human-computer interaction by relieving the user from technical detail. In addition, we want interface agents to take better advantage of the ways humans communicate. Particularly the idea of "situated agents" is of interest to us. A situated agent has the ability to exploit the actual situation, to the extent possible, as a source of information in perceiving and manipulating its environment and communicating with cooperating partners[1] [4, 11]. On the other hand, we found multiagent architectures [1] of great advantage to our goal of building an intelligent interface for a complex application.

In this paper we describe an application of a multiagent system for interacting with a virtual environment by way of situated communication, carried out in the VIENA project [18]. Due to the advancements in rapid generation of high-quality computer graphics, virtual environments have become attractive as a means for interactive visualization. To make better profit of this new technology, e.g., in virtual design, new ways of human-computer interaction are called for [7]. Most of the work so far has concentrated on gesturing and pointing, for instance, by using a data glove (e.g., [3]). An alternative we explore in the VIENA project is to use natural language to communicate alterations which are put in effect by a multiagent system. That is, we instruct the system to carry out changes where gestural manipulation is impossible or unnatural. Eventually, voice input and gestures should be used as parallel input modalities.

2 The VIENA Project

VIENA is a project in a new research focus theme of "Artificial Intelligence and Computer Graphics" at the University of Bielefeld. The overall goal is to apply artificial intelligence techniques in interactive 3D computer graphics. The VIENA project has the following goals:

- enable an intelligent communication with a virtual environment
- develop a multiagent system as a mediator in human-computer interaction
- explore its usefulness for interactive design and exploration

[1] Situated communication is a focus theme in a long-term special research unit at the University of Bielefeld (SFB 360, "Situated Artificial Communicators").

We have chosen interior design as an example domain. Instead of using the mouse and menus to manipulate objects in a virtual scene we use a set of specialized agents to convey changes. The agents produce an action-oriented semantic interpretation to translate the user's input to new images. Our aim is to keep the user (designer) free from technical considerations such as planning of geometric details, etc. Input in simple written natural language is currently used for communication. In the near future we also want to use voice input and simple hand gestures, indicating directions. The current working environment for VIENA includes a Silicon Graphics Indigo ELAN R4000 equipped with the SOFTIMAGE modeling and rendering software.

Our interface 'agency' communicates and cooperates with a human user in an overlapping perceptual situation. A key idea is to have agents inspect the internal description of a scene which the user can experience by eye inspection, so both of them – agent and user – communicate about scene details from the same "point of view." But whereas the user is likely to have a qualitative grasp of the scene, the agents have knowledge of exact object locations, colorings, etc. Qualitative communications of the user to change scene details are evaluated by the agents so as to produce appropriate quantitative changes in the scene description. The altered scene description is then visualized for the user. We refer to this overall process as "intelligent mediation."

The VIENA parser translates an instruction to an internal deep-level representation which outputs to the mediating agents. A bookkeeping agent is authorized to access and modify the augmented data base to supply current situation information to agents on request. A space agent translates qualitative relations such as 'left of' to appropriate scene coordinates. Other agents, in similar ways, take special

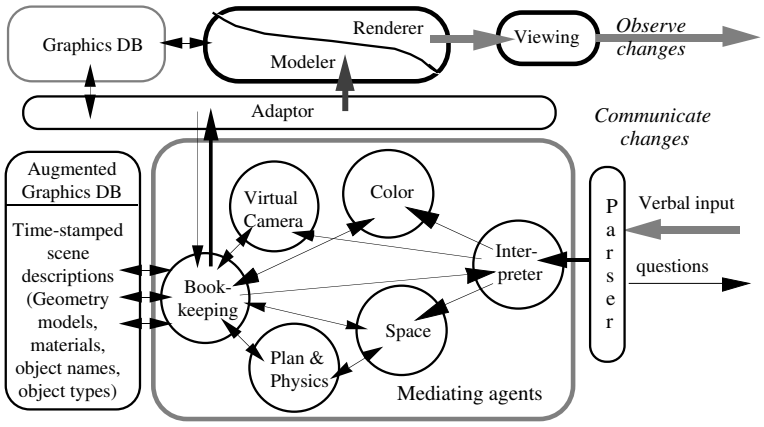

Figure 1: In the VIENA system a multiagent system mediates qualitative verbal instructions by translating them to quantitative commands that are used to update the visualization scene model.

responsibilities in mediating an instruction. Agents cooperate to offer a goal scene corresponding to a user's inquiry. The offer can be changed in further interaction, that is, the user can negotiate the computed semantics of instructions. An adaptor unit converts data back and forth between the mediating agents and the technical modeler (see fig. 1).

3 The multiagent mediator system

For the interactive manipulation of 3D graphical scenes, the multiagent mediator system has to solve a large number of different tasks. In VIENA these tasks are distributed among specialized agents of the multiagent system. In order to compute the entire solution of a user's instruction, each agent has to fulfill mainly the following tasks: Execute allocated tasks for solving subproblems, and exchange messages with other agents to take part in the cooperation process.

With respect to these tasks, in VIENA an agent model was chosen similiar to [16] which subdivides the functional structure of an agent into three components: local functionality, communication, and cooperation (see fig. 2). The local functionality

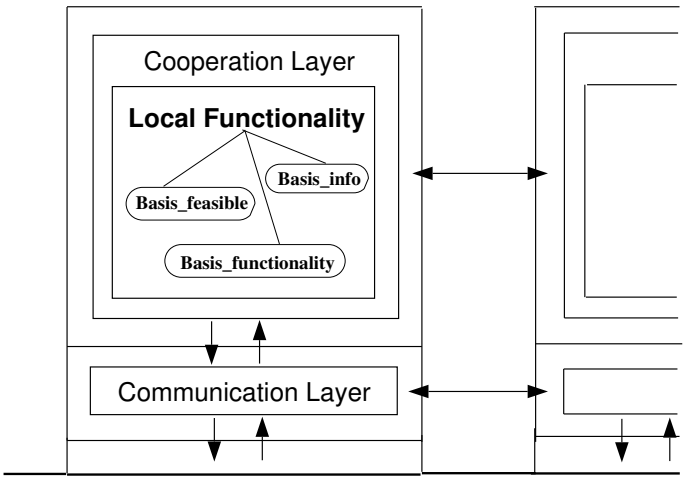

Figure 2: The agent model in VIENA.

consists of functions for announcing the agent's existence (Basis_info), for checking the feasibility of tasks (Basis_feasible), and for solving subtasks (Basis_functionality). The communication component provides the functionality for sending and receiving messages, whereas the cooperation component handles the distribution of subtasks and partial solutions. To guarantee a trouble-free and effective agent interaction, a stand-alone framework was developed which standardizes the communicative and cooperative behaviors of all agents. In order to extend the system's features, it suffices to create a new instance of the agent scheme and provide its local functionality.

By this, a MACE-type environment [8] is realized which enables a fast, incremental construction of multiagent systems.

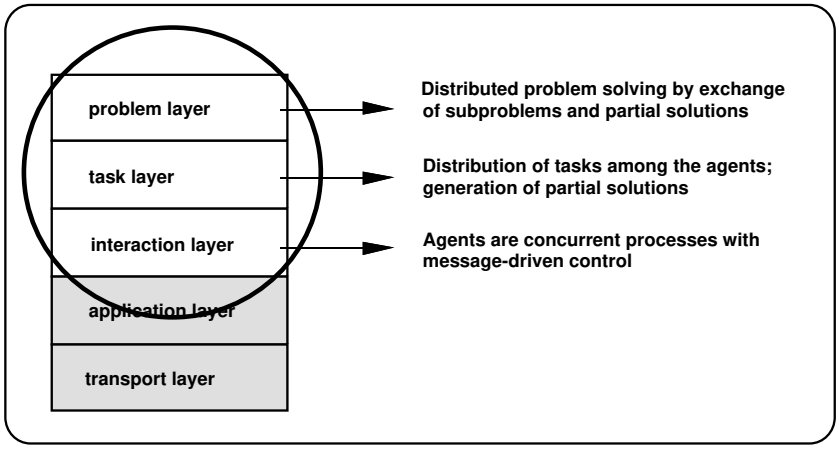

Figure 3: Functional layers of an agent system in the OSI model: Further layers for interaction control, task distribution and distributed problem solving are put on top of the application layer (after [14]).

Following a proposal in [17], our agent model incorporates three functional layers on top of the application layer in the OSI reference model [9]. At the lowest layer, the *interaction layer*, agents are considered as concurrent processes with a message-driven control. The *task layer* handles the distribution of tasks among the agents and the generation of partial solutions. At the top layer, the *problem layer*, the entire task of the interface system is regarded as a distributed problem solving process (see fig. 3).

3.1 Communication

Agents in VIENA are autonomous processes which are handled by the UNIX operating system and which can be installed on different computers in a network. Only at the interaction layer can agents exchange messages with each other. By this, agents are independent from the technical realization of the communication. The interprocess communication of agents is realised with the "Remote-Procedure-Call" (RPC) mechanism developed by SUN Microsystems. Using the RPC model, a Client-Server concept was designed which formally defines the agents as clients and a communication subsystem as server to handle the messages.

Agent communication is based on a uniquely defined language which is common to all participants. This language, on the one hand, defines the data structures for the message protocol and, on the other hand, it defines the communication

primitives to exchange messages by the use of those data structures. In VIENA the data structure MESSAGE is defined as follows:

```
MESSAGE == (MESSAGE-ID,
            SENDER,
            RECIPIENT,
            TYPE,
            REFERENCE,
            TIME-STAMP,
            SOLUTION-TIME,
            CONTENT)
```

The arguments in fields one to seven represent the message head. Depending on the message TYPE, a message can induce subsequent messages which is kept track of in the REFERENCE field. The message TYPE refers to different cooperation protocols (see below). By way of a name convention, individual agents as well as groups of agents can be specified as RECIPIENT of the message. The time behavior of agents can be controlled by way of the TIME-STAMP and SOLUTION-TIME fields. The last argument, in field eight, represents the message body, specifying the CONTENT of the message.

The required communication primitives include functions for connecting and disconnecting agents to the interface system and, secondly, functions for sending and receiving messages to carry out the cooperation among agents.

One problem is to synchronize the agents running on a multi-tasking operating system because the time for execution and message passing is hard to assess. For this reason, we have chosen an asynchronous communication. By appropriate cooperation laws a task-specific synchronization is achieved at the task layer.

3.2 Cooperation

The task layer handles the distribution of tasks among the agents and the generation of partial solutions by the cooperation of the agents. The method developed for VIENA combines different cooperation models found in the literature. Basically, it is characterized by a negotiation process similiar to the "contract-net" approach [6]. Each agent can take the role of a contractor as well as the role of a bidder. In detail, the process consists of a sequence of message passing operations which are: the posting of tasks, the generation of bids, the allocation of tasks or the rejection of bids, resp., and the return of the computed results. This process is realized by the following cooperation primitive:

```
extern Comm_List *Coop_posting_with (Comm_List *message_id,
                                     Comm_List *recipient,
                                     Comm_List *solution_time,
                                     Comm_List *content,
```

```
                                        int         any_time);
```

The boolean `any_time` takes account of offering solutions under time-critical conditions. More precisely, the task is allocated to each agent which has generated a bid and the first solution presented is used for further computations. Since the return of results is not always desired, a further cooperation primitive `Coop_posting_without` is realized where the operation of returning results is skipped.

In addition, it is possible to model a "master-slave"-type of behavior and allocate tasks to agents directly. In this case the posting of tasks and the generation of bids are dropped. The cooperation primitive corresponding to this process is:

```
extern Comm_List *Coop_allocation_with (Comm_List *message_id,
                                        Comm_List *recipient,
                                        Comm_List *solution_time,
                                        Comm_List *content);
```

As above, a further primitive `Coop_allocation_without` is defined where the return of results is skipped. Finally, messages can be addressed to groups of agents simultaneously, which can be interpreted as a "blackboard"-oriented procedure. Therefore, the RECIPIENT field of the message must specify the intended group of agents;

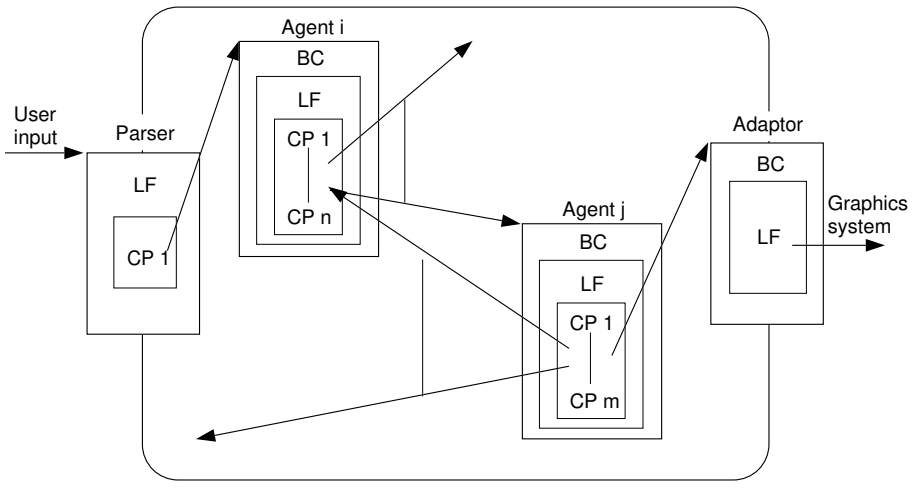

Figure 4: The cooperation framework in the VIENA system.

e.g., by specifying `recipient = '*'`, all agents will receive the message.

These aspects are realized, in total, by the implementation of a cooperation framework with the four cooperation primitives presented above. The cooperation framework reflects the role of the agents as bidders, whereas the cooperation primitives can be used by the agents as contractors. Each agent of the VIENA system is equipped with the bidder-specific frame which triggers the activation of the agent's local functionality. Out of this, contractor-specific primitives are inserted depending on the situation and the task (see fig. 4).

Thus, in the VIENA system all activities resulting from communication and cooperation are defined in the same way for all agents. Further agents can easily be integrated in the system by using the cooperation method and defining their local functionalities. This gives a maximum of liberty in the realization of various kinds of problem solving processes. The problem layer behavior of the VIENA system is described in an example application, below.

4 Example Application

In our current application, the VIENA agent system acts as an intelligent mediator in the human-computer interaction with a virtual environment. However, to the extent possible the system was developed with regard to utilization in other scenarios. The multiagent system meets requirements of robustness and flexibility in dynamically changing situations. Furthermore, the system delivers adequate results under time-restricted conditions. For the sake of interactivity we wish to meet the requirement of real-time. With regard to this, in many cases an integration of classic and knowledge-based techniques was used. Because of the various demands, above all to the quality of visualization, so far a more or less "step-keeping processing" is feasible in practice[2].

The use of the agent system is tested in a prototype scenario with various items of furniture as well as color and light impressions of a virtual office room which can be changed interactively (fig. 5).

The problem solving methods developed account for implicit assumptions of the human when manipulating such an environment; physical laws were reconstructed to avoid the collisions of massive objects in the virtual world; cognitive factors of space are recognized when converting verbal instructions to scene alterations [5].

For example, a space agent knows how to calculate an object's transformation in space (translation, rotation), and it knows that an object must be supported by something and can only be placed in positions not occupied by other objects. The space agent acquires current object positions from the augmented graphic database and informs it about new object positions. Furthermore, the space agent recognizes if an object supports others and, if need be, includes them in a transformation. Finally the space agent is able to act from a deictic or intrinsic perspective (that

[2]On an Indigo Elan, the benchmark shown in fig. 6 currently requires less than 3 minutes altogether. This benchmark is part of an IJCAI-95 demo video.

Figure 5: Example scene (virtual office room) from the VIENA test application in wire frame presentation; in addition, color and light impressions are shown on the screen.

is, the position of the observer or the object itself) in order to interpret the orientations in space. In case the visualized solution does not meet the expectation, the user can correct it. The system then modifies its previous solution accordingly by inspection of the current and the previous scene models. In this way, the semantics of instructions are negotiated in the system-user cooperation.

In a similiar way, a color agent (under construction) has knowledge about how to acquire a current object color (r,g,b), how to calculate a color transformation (e.g., blue, lighter) as a modification of a rgb-vector, and how to identify an object by means of a color description ("the red chair"). A more detailed description of the VIENA application system is given in [18].

A sequence of possible interactions with the VIENA system is shown in fig. 6. The inputs shown allow interactive modification of the visible scene as well as changes of the viewpoint and, finally, the processing of simple deictic instructions; these can either refer to the observer's viewpoint or to a detail pointed out from this viewpoint (in this situation an indicating arm becomes visible which could, in future, be controlled by a gestural input medium).

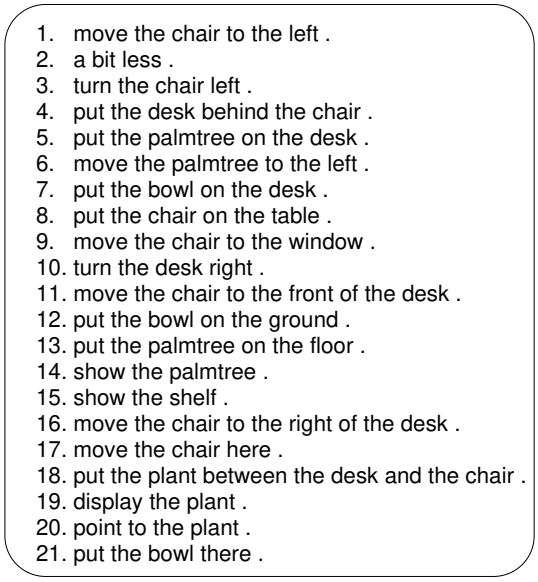

Figure 6: Sample sequence of interactions with the VIENA system.

Extrapolating from this test application, advanced features in the interactive design of a virtual world can be envisaged. Partial sequences of actions could be planned automatically by further agents ("move the desks to the right wall", "make the desk tops grey", etc.). As a further application feature, models of individual objects can be exchanged; for example "real" CAD models of a kitchen manufacture company were imported and manipulated in the VIENA environment to probe adaptability of our prototype system in more realistic conditions of use.

5 Discussion and Future Work

In this paper, an application of a multiagent system was described in which agents serve as an interface between a user and a virtual environment. In our example domain of interior design, the VIENA multiagent system mediates qualitative verbal instructions by translating them to quantitative commands that are used to update the visualization scene model. That is, natural interaction with the application is enabled by an intelligent, situated communication brought about by the mediating interface agents.

It is our impression that multiagent systems can play an important part in human-computer interaction and in future user interface design, in general. The main idea is to relieve technical detail from the user by making knowledged-based agents do part of the work (cf. also [2]). What are the advantages of using systems of interface agents – rather than using a single-agent architecture? In our work,

we found the use of multiagent systems justified in a domain with distributed resources, tasks, and competences. To name a few rationals for using a multiagent architecture:

- Modularization: The multiagent architectures gives rise to natural modularization which is also easily understood among the members of a project team.
- Efficacy and flexibility: Alternative solutions can be negotiated among agents; incl. the user can negotiate an offered solution by issuing a corrective command in the following interaction.
- Robustness and stability: Missing information can be gained via communication among agents; failures of single components could be discovered and outweighed without locking the system.
- Efficiency: Independent subtasks can be solved locally, and speed is increased when complex instructions can be processed in parallel by agents working on partial solutions.
- Anytime-solutions: Crude solutions can be offered by simple agents early, while more fine-tuned agents still work on a better solution as long as allocated compute time has not elapsed.
- Multimodal input: Different modes of interaction can be integrated by the cooperation of agents; e.g., we have started to work on integrating language and gestures.
- Adaptivity: We have also begun to work on agents that make offers qualified by their success in the preceding session, as a way to adapt the interface system to individual users' preferences without accumulating user models.

In closer regard of our example application: Once the agent metaphor is present in a view of mediated interaction with a virtual environment, we can envisage an "embodied" virtual agent to be visible, and become effective, in the scene. In our most recent work, we manipulate the virtual environment via a virtual interface agent (VIA) which is present in the virtual scene. By the use of a VIA, designers can be better immersed in the scene and can make use of anthropomorphic features. They might adopt different body sizes, and switch from an external to a "situated" view at various positions. This is where we want to go next.

References

[1] Adler, M., Durfee, E., Huhns, M., Punch, W. & Simoudis, E. (1992). AAAI Workshop on Cooperation Among Heterogeneous Intelligent Agents. *AI Magazine, Vol.13(2)*, 39-42.

[2] Bishop, G., Bricken, W., Brooks, F., Brown, M., Burbeck, C., Durlach, N., Ellis, S., Fuchs, H., Green, M., Lackner, J., McNeill, M., Moshell, M., Pausch, R, Robinett, W., Srinivasan, M., Sutherland, I., Urban, D. & Wenzel, E. Research Directions in Virtual Environments. *Computer Graphics 26(3)*, 153-177.

[3] Böhm, K., Hübner, W. and Väänänen, K. (1992). GIVEN: Gesture driven interactions in virtual environments: A toolkit approach to 3D interactions. *Proc. Conf. Interface to Real and Virtual Worlds* (Montpellier, France, March 1992).

[4] Brooks, R.A. (1991). Intelligence without reason. *Proceedings IJCAI-91*, 569-595.

[5] Cao, Y. & Wachsmuth, I. (1993). Situated Space Agent for 3D Graphics Design. Extended abstract in *Virtual Reality Vienna 1993, the Global VR-Focus in Europe – Abstracts* (pp.8-9). Full paper: KI-NRW Report 93-13.

[6] Davis, R. & Smith, G. (1983). Negotiation as a Metaphor for Distributed Problem Solving. In Bond, A.H. & Gasser, L.: *Readings in Distributed Artificial Intelligence*, 333-356, Morgan Kaufmann.

[7] Furness, T. (1991). Exploring Virtual Worlds with Tom Furness. A supplement to *The Communications of the ACM, Vol.34*, No.7, 1/11.

[8] Gasser, L., Braganza, C., Herman, N. (1987). Implementing Distributed AI Systems Using MACE. In Bond, A.H. & Gasser, L.: *Readings in Distributed Artificial Intelligence*, 445-450, Morgan Kaufmann.

[9] Kerner, J. (1992). *Rechnernetze nach OSI*. Bonn: Addison-Wesley.

[10] Laurel, B. (1990). Interface agents: Metaphors with character. In Laurel, B. (Ed.): *The art of human-computer interface design*, 355-365, Addison-Wesley.

[11] Lobin, H. (1993). Situiertheit. *KI 1/93* (Rubrik "KI-Lexikon"), 61.

[12] Maes, P. & Kozierok, R. (1993). Learning interface agents. *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)* (pp. 459-465). Menlo Park: AAAI Press/The MIT Press.

[13] Maes, P. (1994). Agents that Reduce Work and Information Overload. *Communications of the ACM, Vol.37*, No.7, 31-40.

[14] Marcus, A. (1993). Future directions in advanced user interface design. In N.M. Thalmann & D. Thalmann (Eds.), *Communicating with virtual worlds* (pp. 2-13). New York: Springer.

[15] Norman, D.A. (1994). How Might People Interact with Agents. *Communications of the ACM, Vol.37*, No.7, 68-71.

[16] Steiner, D., Mahling, D. Haugeneder, H. (1990). Human Computer Cooperative Work. In M. Huhns (eds.), *Proceedings of the 10th International Workshop on Distributed Artificial Intelligence*. MCC Technical Report No. ACT-AI-355-90.

[17] von Bechtolsheim, M. (1993). *Agentensysteme: Verteiltes Problemlösen mit Expertensystemen*. Braunschweig: Vieweg-Verlag.

[18] Wachsmuth, I., Cao, Y. (1995). Interactive Graphics Design with Situated Agents. In W. Strasser & F. Wahl (eds.): *Graphics and Robotics* (pp. 73-85), Springer.