Designing Transition Networks for Multimodal VR-Interactions Using a Markup Language

Marc Erich Latoschik
AI & VR Lab [28], Faculty of Technology, University of Bielefeld, Germany
(marcl@techfak.uni-bielefeld.de)

Abstract

This article presents one core component for enabling multimodal—speech and gesture—driven interaction in and for Virtual Environments. A so-called temporal Augmented Transition Network (tATN) is introduced. It allows to integrate and evaluate information from speech, gesture, and a given application context using a combined syntactic/semantic parse approach. This tATN represents the target structure for a multimodal integration markup language (MIML). MIML centers around the specification of multimodal interactions by letting an application designer declare temporal and semantic relations between given input utterance percepts and certain application states in a declarative and portable manner. A subsequent parse pass translates MIML into corresponding tATNs which are directly loaded and executed by a simulation engines scripting facility.

Keywords: *multimodal interaction, Virtual Reality, multimodal integration, transition networks, XML interaction representation*

1 Introduction

Virtual environments (VE) and virtual reality (VR) applications demand appropriate methods for human-computer-interaction (HCI). Desktop oriented input-devices and interaction metaphors are not adequate with respect to the requirements and the users mobility in large screen immersive surroundings like workbenches, projection walls or caves. Nevertheless, by applying existing 2D paradigms to 3D (by, e.g., using a space mouse and floating menus), the WIMP (Windows, Icons, Menu, Pointer) style interaction found its way into many of today's VR-applications. The utilization of natural human communication capabilities—so far with a strong emphasis on speech input—has, on the other hand, a long research history and still promises the realization of a somehow ideal human-machine interface. But the potential of natural gestures as a modality for HCI has not been fully utilized. This is partly due to the spatial restrictions of user movements when operating desktop systems for which the term *gestures* often relates to typical pen-based input strokes on the screen.

Figure 1. Two examples of speech and coverbal gesture interaction in VR. In the top row a user triggers an action just by saying "...open the door...". The bottom row illustrates how an object is interactively manipulated using a (distant) gesture and speech by saying "...turn it this way..." with an accompanying kinemimic gesture describing the action.

Compared to desktop systems, large screen immersive VEs allow a much higher degree of freedom for a user (Fig. 1). In addition, such systems often rely on tracking hardware, e.g., to follow the user's head position for calculating a perspective scene projection or for 3D pointing devices. This tracking can also be utilized for gesture detection purposes. Hence, VR-systems seem to be ideal test-beds for research on multimodal interfaces. VR, on the other hand, might benefit from new gesture/speech interaction metaphors that free a user from carrying input devices when entering a VE, particularly if we consider cable-less (optical) tracking systems nowadays available. Typical *point-and-click* operations could be complemented or replaced by a type of a natural communication

which in addition seems to be more adequate when we incorporate embodied conversational agents as communication partners into the VE. Figure 1 illustrates two examples of multimodal VR-interactions realized with *integration* techniques explained in this paper. It presents a XML-compliant markup language and supporting concepts that allow a declarative specification of the speech and gesture integration process while additionally taking application context into account.

2 Related work

Deploying multimodal interaction for graphics systems can be tracked back to 1980 and Bolt's Put-That-There system [3], where (2D) graphical objects could be moved around on a large screen in a strictly dialog driven interaction utilizing pointing directions. Likewise, the integration of deictic utterances for 2D applications was explored by Hauptmann and McAvinney [8], Koons et al. [13], Maybury [18], or Lenzmann [16]. Böhm et al. [1][2] developed one of the first gesture interfaces for a VR-system. They used *symbolic*—unambiguous and predefined—gestures to trigger system actions. Weimer and Ganapathy [26] focused on the creation and modification of curves in space by analyzing aspects of arm movements. The ICONIC system by Sparrell and Koons [13][22] is remarkable for the exploitation of *iconic* gestures (shape gestures) to specify objects or their manipulation. To utilize multimodal input specifically for VR-setups is the goal of Cavazza et al. [6] and—more recently—by Lucente [17]. Additional overviews of related work can be found in [5] and [21].

With respect to multimodality, we follow Nespoulous and Lecour [20] and distinguish *deictic*, *spatiographic*, *kinemimic* and *pictomimic* as subtypes of *coverbal illustrative* gestures on a functional basis. Their definitions focus on speech related gesture functions when expressing spatial content and hence are closely related to parameters that are desirable to be manipulated during a 3D interaction.

Advances regarding general VR techniques include tools and standards like Java 3D, VRML97 [4] or X3D [25] that support the creation of virtual environments including (on a low level) methods to enable interactions with these worlds. These approaches center around a scene graph based representation with the incorporation of application logic through an explicit event routing mechanism or the introduction of specialized node types, e.g., script and sensor nodes. VR research tools like AVANGO [23] etc. adopt many of the above principles and add features like distribution, a general scripting support, the configuration of advanced display systems and they support specific VR-input hardware. Regarding interactions, the 3dml [7] approach aims at a generic way to declaratively specify not only scene content but the interaction layer as well.

Several methods for the integration of multimodal utterances can be identified. In the beginning, dialog and/or speech-driven approaches were favored in which the place where a gesture could occur was specified in advance. Work done by Bolt [3], Neal and Shapiro [19], Koons et al. [13], and Lenzmann [16] falls into this category. In such systems, speech interpretation resulted, e.g, in a type-token representation where some tokens could (and sometimes had to) be accompanied by a specific gesture to be complete. Other approaches condense type-token structures using a frame notation for the multimodal input information (Koons et al. [13], Vo and Wood [24]). In all these systems integration knowledge—what can be integrated with what and how—is represented in a procedural manner.

More recent work, e.g., Johnston [9] (Johnston et al. [12]), favors an advanced frame notation by using *feature structures*, attribute matrices that can be (recursively) combined: values in a feature structure can themselves be a feature structure. The structure values are controlled by *constraints* which represent integration knowledge and establish context sensitivity regarding certain feature values. A central *unification* operation tries to fill all possible structure attributes using a) collected input and b) other matching structures while c) taking the respective constraints into account. The unification scheme is very powerful in its expressiveness. But as pointed out by Johnston et al. [10][11] and in our own work [15], unification has some drawbacks when implemented in real applications, namely an inherent computational complexity and the absence of methods for a tight coupling to an application. Therefore, Johnston et al. revised their unification approach in favor for a finite-state automata (FSA) model [11] for the multimodal parsing and integration. It is based on a multimodal grammar, where the terminals contain three (possibly empty) components, for speech, gesture, and combined meaning. Such grammars do not incorporate any explicit temporal relations between the percepts and can be approximated in an application by FSAs. Example applications for the FSA approach were given in the area of speech accompanied pen-based input systems, where pen strokes were used to express deictic content (sometimes expressed with iconics), namely to identify places or objects.

The following section will introduce an alternative way for integrating speech and gestures particularly useful in the area of VR. It overcomes some of the limitations of unification and FSA methods described above, which avoided—or at least made it extremely cumbersome—to exploit them in the context of multimodal interaction for real-time immersive application. The method was explicitly designed taking the recent research results about declarative scene and interaction specification for VR into account.

3 Multimodal integration in VR using tATNs

Natural coverbal gestures incorporate a large variety of temporal, spatial, and dynamic features. Although there is ongoing work on the cross-modal temporal relations between certain speech percepts (stressed or unstressed phonemes, words or phrases) and identifiable gesture parts, e.g., the gesture *stroke*, the experimental results today can not in general be utilized for a strict and formal parsing scheme. For the latter, we would need a kind of a gesture—or finally a multimodal—grammar. Whether such a grammar exists is still an active research topic and presumes the existence of identifiable units like words constitute for natural language. The problem of finding a clean structure in the gesture stream might be one of the reasons for the sometimes vague or even contradictory results reported in the context of cross-modal temporal relations. Nevertheless there are some relations that hold and that should be considered when developing tools for the design of multimodal interfaces. This includes support for temporal constraints between input streams of varying granularity, incorporation of integration methods based on the inputs semantic content, and—regarding the HCI utilization—access to information from the application context level.
Regarding interaction techniques, discrete interactions which will be executed in one atomic operation after a complete multimodal parse do not exploit the VR-specific interactivity advantage. Here, kinemimic gestures lead to *continuous* interactions which should be started before a multimodal utterance (and hence its parse) ends and in which the gesture stream takes over and controls the interaction.
Hence, to conduct research in the area of multimodal VR interactions, tools are required that should flexibly support the implementation and parameterization of a variety of approaches. The primarily required features are:

- Processing of parallel incoming percepts
- Partial parse execution
- Varying level of percept granularity
- Support for temporal and semantic relations
- Easy parameterization, e.g., of relation attributes
- Support for application context integration
- Latching into real-time applications

Considering these requirements, we have developed a concept called a temporal augmented transition network (tATN) [15]. A basic ATN consists of states and possible transitions between the states. In contrast to a FSA, transitions can be accompanied with the generation of actions or events and can be augmented with guarding constraints to control if a transition is allowed to be carried out or not. The states of an ATN might also include registers to hold certain state specific values. Constraints, actions, and registers are closely bound to an ATN's environment and allow a sophisticated coupling to external, e.g., application specific objects. An action can further represent the transition of a complete sub-ATN, e.g., figure 2 depicts a sub-tATN which is—as a whole—referenced in figure 3 by the function (ObjDesc1). A tATN enhances the ATN formalism with methods to incorporate *temporal* information about transitions. It makes this information available to guarding constraints and it supports more than one active tATN state at a time. These features—in the context of multimodal utterances—enable a tATN to process parallel incoming percepts. In the tATN figures, guarding constraints and functions are depicted in round brackets. Constraints are directly located at the appropriate transition arcs (filled arrows) and conjuncted by boolean operators, whereas white arrows point to functions to be called at certain events, e.g., when reaching a state. Grey states represent end states.

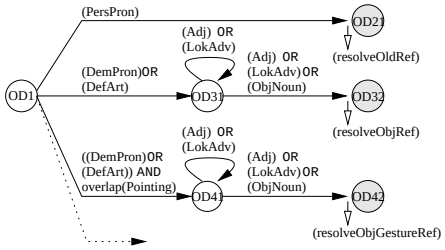

Figure 2. A tATN branch for processing uni- and multimodal object descriptions (see text).

Figure 2—taken from an actual realized virtual construction application—illustrates how uni- and multimodal utterances are processed using a tATNs specific functionality. The short upper branch only parses personal pronouns (PersPron) referring to previously specified references. The middle and lower branches process definite noun phrases starting with a) definite articles (DefArt) or b) demonstrative pronouns (DemPron) accompanied without (`OD1->OD31`) and with (`OD1->OD41`) a temporally overlapping pointing gesture. The predicates `overlap(...)` and `before(...)` (see figure 3) denote examples for temporal constraints which test if a condition holds true based on the tATNs temporal state information. This is achieved as follows. Each state `S` in the tATN has time-stamp registers to store temporal information about the transitions. Should `S` be left on grounds of an existing lexical constraint, a register `S.current` is set to the beginning of the recognized percept (`S.current=lexical.start`), in the actual system this is the time when the speech recognizer identifies the start of the word utterance. In absence

of such a constraint it is set to the latest percept time that triggered the state change. If there is no percept time, e.g., when processing application states, the register value of the predecessor state is copied. This semantics can be altered from the outside by specifying a `state.get-time` function per state and register. Predicates can now be defined with respect to the time-stamps, e.g., the temporal predicates implicitly refer to the `current` register. Following figure 2, adjectives, location adverbs or nouns will leave states `OD31` or `OD41` and *may* lead to two different end-states or may activate multiple active branches when an adjective or location adverb is uttered. The end states `OD32` and `OD42` will execute different functions to resolve the references using the current application context. Figure 3 illus-

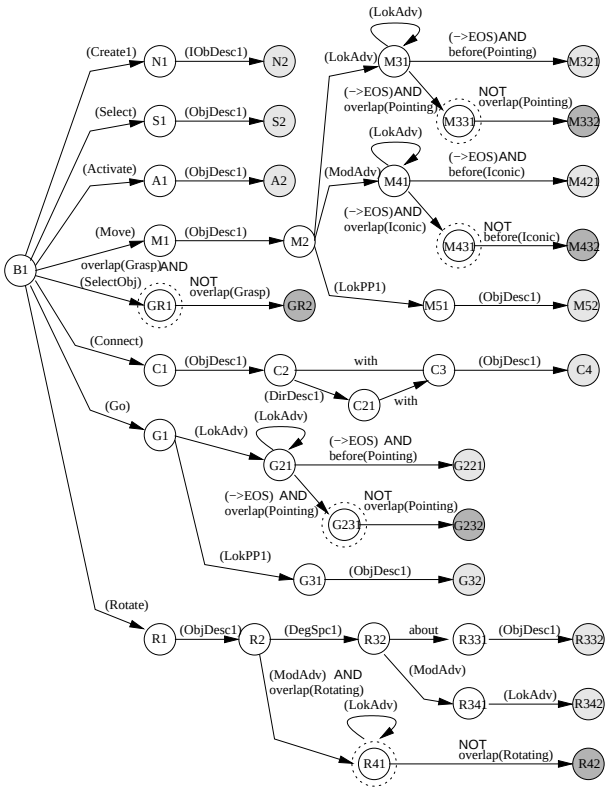

Figure 3. Core tATN of a virtual construction application. It includes processing of continuous object manipulations by utilizing kinemimic gestures (see text).

trates the implemented core tATN that enables multimodal interactions for a virtual construction application. The additional symbols have the following meaning: Constraints that look for specific words are written without parantheses. States surrounded by a dotted circle denote a switch-over to the continuous interaction state where interaction control is handed over to the gesture analysis and detection process (see transition between `R41->R42`). This gesture driven manipulation is enabled via bindings between specialized scene graph node types where actuators route the movement data, e.g., of the fingertip trajectory, to so-called *motion-modifiers*. Motion-modifiers filter the unprecise motion data and map movement to possible object changes. Their output is routed to manipulator nodes that apply incremental manipulation steps to the object for each simulation frame (for more information about gesture processing see [14]). The look-ahead symbol: `->` checks the next percept in the according input stream but does not remove that percept (`->EOS` checks for a end of sentence symbol delivered by the speech recognition).

To latch the tATN evaluation into a typical VR simulation loop, it is not event driven by its incoming percepts, but triggered during each simulation step. Input is asynchronously stored in buffers for the upcoming evaluation step, e.g., the numerical results of the gesture detection modules are stored in so-called *histories* which hold data values corresponding to the gesture processing's own rate. For example, the temporal predicate `before(Pointing)` (see figure 3 at transition `M31->M32`) evaluates a history of values—that represent the saliency of a pointing gesture—with respect to state `M31`'s `current` register. The presented tATN concepts have been implemented using ELK SCHEME (a LISP dialect), the main scripting language supported by the VR-toolkit AVANGO [23]. tATNs build the outer shell of a developed toolkit consisting of modules for gesture analysis, gesture detection, real-time coupling and semantic scene representation.

4 MIML - The Multimodal Interaction Markup Language

Although the tATNs have proven their usefulness in a couple of applications, there are still a few drawbacks. Interaction implementation is closely bound to a specific application. Hence, the specification and modification of a tATN requires certain skills or in-depth knowledge about application internals by the interaction designer. Modifications of an existing tATN can become complicated due to side-effects of the transition actions. The current implementation is based on *SCHEME* as the scripting language. An independent declarative way to specify tATNs would enable support for different implementations and would ease the tATNs design burdens. Therefore we combine approaches from [7] and [27] with the tATNs capability to describe and parse multimodal interactions by choosing a XML representation for the tATN based interaction specification. The following fragments of this multimodal interaction markup language (MIML) illustrate the key concepts and tags. At the top level, a MIML interaction is encapsulated by:

```
<definition start="userInstruction">... </definition>
```

The connection to a given application is carried out by a generic application layer file which basically defines function stubs to be implemented. This decouples interaction specification and design from the actual implementation. For example, the following tags define three different types of application stubs. The first one declares a generic function call, the second one declares a word type classification function and the last one declares a gesture detection access function respectively, where in our system the latter will evaluate the appropriate gesture detection result history (prBHistory).

```
<extern apiCommand="apiRotateObjByHand-On"/>
<wordtype function="defArticle">
  <word>the, that, this</word>
</wordtype>
<gesturetype function="rotating" type="prBHistory">
  <history name="hist-rotating"/>
  <field name="axis"/>
  <field name="degree"/>
</gesturetype>
```

The basic building block for an interaction is embedded in a *requirement* tag. Requirements consist of 1) a *frame* tag which specifies the interaction parameters and 2) a *description* tag for specifying the multimodal integration. *select/choice* tags inside a description represent possible transitions of the tATN and hence can be augmented with constraints and functions. *temporalrelation* defines a temporal relation that has to hold for the included content. This tag has additional attribute slots to specify the time-stamp referenced and an according interval in which the relation should hold (initialized with default values). A *require* tag branches to a sub tATN. The following excerpt illustrates the MIML representation of the (slightly modified) rotate-object interaction defined by the transition `R1->R2->R41->R42->R43` in figure 3.

```
<requirement name="rotateObject" function="userInstruction">
  <frame>
    <slot name="object" type="singleslot"/>
    <slot name="degree" type="singleslot" default="30"/>
    <slot name="axis" type="singleslot" default="0 0 1"/>
    <slot name="rotcenter" type="singleslot" default="@object" />
  </frame>
  <description>
    <temporalrelation type="sequential">
      <speech>
        <function name="rotateAction"/>
      </speech>
      <requires>
        <function name="objectDescription"/>
        <fill-slot source="identifier" target="object"/>
      </requires>
      <select>
        <choice>
          ⋮
        </choice>
        <choice>
          <temporalrelation type="overlap">
            <speech>
              <function name="modalAdverb"/>
            </speech>
            <gesture>
              <function name="rotating"/>
              <exec-on-start>
                <apiCommand name="rotateObjectByHand-On"/>
              </exec-on-start>
              <exec-on-end>
                <apiCommand name="rotateObjectByHand-Off"/>
              </exec-on-end>
            </gesture>
          </temporalrelation>
        </choice>
      </select>
    </temporalrelation>
  </description>
</requirement>
```

The opening frame defines all slots required for a *discrete* object rotation (with default values), which would be set by the transition(s) starting at the `R1->R2->R32` branch in figure 3. The description starts with a declaration that requires all upcoming functions (on the same level) to be sequential regarding their temporal relations to each other. The first utterance access function performs a lexical test for utterances like "turn" or "rotate" etc. . The next *requires* tag branches to a sub tATN for parsing DNPs (see figure 2). The first choice branch to `R32` was skipped for clarity at the following select/choice tag which continues with a specification of a required temporal overlap between utterances like "this way" or "like this" (called modalAdverb) and a rotation gesture access function. This is representing the transitions `R2->R41`. The inclusion of the *apiCommand* tags inside the respective *gesture* tag depicts how switching to the continuous interaction is performed by augmenting the rotation access function with start and stop actions (not depicted in figure 2). The subsequent closing tags just satisfy the syntax. Translating the MIML definitions into the according tATN scheme objects is done automatically using the xalan/xerxes XSLT parsing approach.

5 Conclusion

After the first initial work, MIML and its supporting tATN modules are now used for several projects, e.g., for the *Virtual Workspace* and the SFB360 sponsored by the Deutsche Forschungsgemeinschaft (DFG) as well as for several PhD projects at our VR-lab. Starting a couple of

years ago with straight forward processing of command sequences using about 300 words and designing required tATNs by hand, current and ongoing work significantly expands the vocabulary as well as the grammar and even introduces a dialog, all facilitated by the convenient MIML specification.

Acknowledgment: The initial framework was sponsored by the German Federal State of North-Rhine Westfalia in the context of the *SGIM* (Speech and Gesture Interfaces for Multimedia) project as part of the collaborative research group *Virtual Knowledge Factory* and later by the DFG in the context of the SFB360. The author likes to thank Lars Gesellensetter and Malte Schilling for their work on the SCHEME tATN implementation and the MIML spec.

References

[1] K. Böhm, W. Broll, and M. Sokolewicz. Dynamic gesture recognition using neural networks; a fundament for advanced interaction construction. In S. Fisher, J. Merrit, and M. Bolan, editors, *Stereoscopic Displays and Virtual Reality Systems, SPIE Conference Electronic Imaging Science & Technology*, volume 2177, San Jose, USA, 1994.

[2] K. Böhm, W. Hübner, and K. Väänänen. Given: Gesture driven interactions in virtual environments; a toolkit approach to 3D interactions. In *Interfaces to Real and Virtual Worlds*, 1992.

[3] R. A. Bolt. Put-That-There: Voice and gesture at the graphics interface. In *ACM SIGGRAPH—Computer Graphics*, New York, 1980. ACM Press.

[4] R. Carey, G. Bell, and C. Marrin. ISO/IEC 14772-1:1997 virtual reality modeling language (VRML). Technical report, The VRML Consortium Incorporated, 1997.

[5] J. Cassell. A framework for gesture generation and interpretation. In R. R. Cipolla and A. Pentland, editors, *Computer Vision for Human-Machine Interaction*. Cambridge University Press, 1998.

[6] M. Cavazza, X. Pouteau, and D. Pernel. Multimodal communication in virtual environments. In *Symbiosis of Human and Artifact*, pages 597–604. Elsevier Science B. V., 1995.

[7] P. Figueroa, M. Green, and H. J. Hoover. 3dml: A language for 3d interaction techniques specification. In *Proceedings of the EUROGRAPHICS 2001*, 2001.

[8] A. Hauptmann and P. McAvinney. Gestures with speech for graphic manipulation. *International Journal of Man-Machine Studies*, 38:231–249, 1993.

[9] M. Johnston. Unification-based multimodal parsing. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th Annual Meeting of the Association for Computational Linguistics COLING-ACL*, 1998.

[10] M. Johnston and S. Bangalore. Finite-state multimodal parsing and understanding. In *Proceedings of COLING 2000*, Saarbrücken, Gemany, 2000.

[11] M. Johnston and S. Bangalore. Finite-state methods for multimodal parsing and integration. In *Finite-state Methods Workshop, ESSLLI Summer School on Logic Language and Information,Helsinki, Finland*, august 2001.

[12] M. Johnston, P. R. Cohen, D. McGee, S. L. Oviatt, J. A. Pittman, and I. Smith. Unification-based multimodal integration. In *35th Annual Meeting of the Association for Computational Linguistics, Madrid*, pages 281–288, 1997.

[13] D. Koons, C. Sparrel, and K. Thorisson. Intergrating simultaneous input from speech, gaze and hand gestures. In *Intelligent Multimedia Interfaces*. AAAI Press, 1993.

[14] M. E. Latoschik. A general framework for multimodal interaction in virtual reality systems: PrOSA. In W. Broll and L. Schäfer, editors, *The Future of VR and AR Interfaces - Multimodal, Humanoid, Adaptive and Intelligent. Proceedings of the Workshop at IEEE Virtual Reality 2001*, number 138 in GMD report, pages 21–25, 2001.

[15] M. E. Latoschik. *Multimodale Interaktion in Virtueller Realität am Beispiel der virtuellen Konstruktion*. PhD thesis, Technische Fakultät, Universität Bielefeld, 2001.

[16] B. Lenzmann. *Benutzeradaptive und multimodale Interface-Agenten*. PhD thesis, Technische Fakultät, Universität Bielefeld, 1998.

[17] M. Lucente, G.-J. Zwart, and A. D. George. Visualization space: A testbed for deviceless multimodal user interface. In *Intelligent Environments Symposium*, American Assoc. for Artificial Intelligence Spring Symposium Series, Mar. 1998.

[18] M. T. Maybury. Research in multimedia an multimodal parsing and generation. In P. McKevitt, editor, *Journal of Artificial Intelligence Review: Special Issue on the Integration of Natural Language and Vision Processing*, volume 9, pages 2–27. Kluwer Academic Publishers Group, 1993.

[19] J. G. Neal and S. C. Shapiro. *Intelligent User Interfaces*, chapter Intelligent Multi-Media Interface Technology, pages 11–45. Addison-Wesley Publishing Company, 1991.

[20] J.-L. Nespoulous and A. Lecours. Gestures: Nature and function. In J.-L. Nespoulous, P. Rerron, and A. Lecours, editors, *The Biological Foundations of Gestures: Motor and Semiotic Aspects*. Lawrence Erlbaum Associates, Hillsday N.J., 1986.

[21] S. Oviatt, P. Cohen, L. Wu, J. Vergo, L. Duncan, B. Suhm, J. Bers, T. Holzman, T. Winograd, J. Landay, J. Larson, and D. Ferro. Designing the user interface for multimodal speech and pen-based gesture applications: State-of-the-art systems and future research directions. In *Proceedings of the HCI2000*, 2000.

[22] C. J. Sparrell and D. B. Koons. Interpretation of coverbal depictive gestures. In *AAAI Spring Symposium Series*, pages 8–12. Stanford University, March 1994.

[23] H. Tramberend. A distributed virtual reality framework. In *Virtual Reality*, 1999.

[24] M. Vo and C. Wood. Building an application framework for speech and pen input integration in multimodal learning interfaces. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, 1996.

[25] http://www.web3d.org. WWW.

[26] D. Weimer and S. Ganapathy. Interaction techniques using hand tracking and speech recognition. In M. Blattner and R. Dannenberg, editors, *Multimedia Interface Design*, pages 109–126. ACM Press, 1992.

[27] http://www.w3.org/tr/xtnd. WWW.

[28] http://www.techfak.uni-bielefeld.de/techfak/ags/wbski/wbski_engl.html. WWW.