A User Interface Framework for Multimodal VR Interactions

Marc Erich Latoschik
AI & VR Lab
University of Bielefeld
PO 100131, 33501 Bielefeld, Germany
marcl@techfak.uni-bielefeld.de

ABSTRACT

This article presents a User Interface (UI) framework for multimodal interactions targeted at immersive virtual environments. Its configurable input and gesture processing components provide an advanced behavior graph capable of routing continuous data streams asynchronously. The framework introduces a Knowledge Representation Layer which augments objects of the simulated environment with Semantic Entities as a central object model that bridges and interfaces Virtual Reality (VR) and Artificial Intelligence (AI) representations. Specialized node types use these facilities to implement required processing tasks like gesture detection, preprocessing of the visual scene for multimodal integration, or translation of movements into multimodally initialized gestural interactions. A modified Augmented Transition Nettwork (ATN) approach accesses the knowledge layer as well as the preprocessing components to integrate linguistic, gestural, and context information in parallel. The overall framework emphasizes extensibility, adaptivity and reusability, e.g., by utilizing persistent and interchangeable XML-based formats to describe its processing stages.

Categories and Subject Descriptors: I.3.6 [Computer Graphics] Methodology and Techniques [Inter-action techniques]; I.3.7 [Computer Graphics] Three-Dimensional Graphics and Realism [Virtual reality]; I.2.1 [Artificial Intelligence] Applications and Expert Systems [Natural language interfaces]; I.2.7 [Artificial Intelligence] Natural Language Processing [Language parsing and understanding]; H.5.2 [Information Interfaces and Presentation (e.g., HCI)] User Interfaces [Natural language]

General Terms: Design, Algorithms.

Keywords: multimodal interaction, user interface framework, Virtual Reality, gesture and speech processing, semantic scene description.

1. INTRODUCTION

Virtual Environments (VEs) are characterized by a continuous human-computer interaction loop with a tight coupling between a user and the simulated environment to achieve believable impressions of immersion and presence. In such environments, multimodal—speech and gesture—interfaces are promising alternatives to desktop input devices and WIMP (windows, icons, menus, pointer) metaphors. They provide a wide spectrum of appropriate interaction ranging from gesture-based direct manipulation to distant multimodal instruction or even discourse based communication with artificial humans (see figure 1).

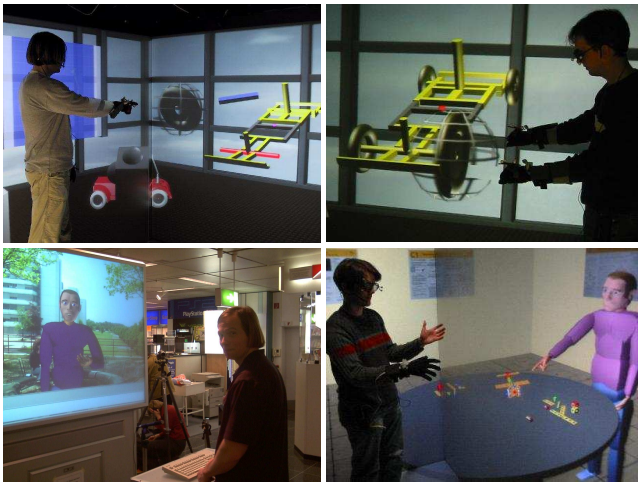

Figure 1: Four example applications using the Interface framework: Distant multimodal interaction and direct manipulation (Virtuelle Werkstatt), typed input and camera based tracking (HNF Max), Multimodal discussion with Max about referenced objects (SFB 360) (from top-left to bottom-right).

AI, natural language and gesture processing, and human cognition research provide various concepts and methods required for multimodal interfaces. Integrating these methods in immersive environments must account for continuous environment changes under real-time constraints. This includes animated scenes as well as changing frames of reference of users or communication partners during movements.

Furthermore, technical achievements age. To avoid error-prone reconstruction of former achievements, e.g., after projects ended or originators left, software maintenance calls for reusable, extensible and adaptive tools. This is commonly approached by explicit interface specifications, well-defined architectures, concepts, and development paradigms as well as by open persistent interchange formats. With respect to VR, 3D graphics and real-time simulation, such solutions have, e.g., lead to the development of scene graphs, behavior graphs, standards for scene content and behavior description as well as of several open development tools.

ICMI'05, October 4–6, 2005, Trento, Italy.

Similar demands motivated the development of the interface framework introduced here. It proposes several general techniques for the implementation of multimodal interactions in highly interactive systems while it provides applications designers with interconnectable processing modules conveniently configurable using declarative and persistent description formats.

2. RELATED WORK

Multimodal interaction for graphics displays was first explored by Bolt [4]: The Put-That-There system allowed to displace 2D graphical objects on a large screen in a dialog controlled interaction by evaluating the pointing direction measured with a 6DOF (Degree Of Freedom) sensor at certain dialog states. The integration of deictic utterances for 2D applications found several successors in [8], [14] or [21]. The work by Sparrell and Koons in the context of the ICONIC system [25] and later by Koons et al. [14] was remarkable for the utilization of *iconic* gestures (gestures that describe, e.g., shape) to specify objects or actions.

An early approach of a gesture interface specific for a VR-system was developed by Böhm et al. [3] which used *symbolic* gestures—unambiguous gestures with a predefined meaning—to trigger system commands. Cavazza et al. [6] analyze multimodal pointing gestures in VR where they define *extended pointings* as selection processes of one or more objects. The approach by Lucente et al. [20] realizes different multimodal object manipulations like selection, dragging and scaling similar to the ICONIC system but is remarkable for its camera based input. Cohen et al. give a brief summary of their comprehensive work in [7] where they explicitly describe their system's architecture which is capable of running in an immersive surrounding.

More recent work is again targeted at the analysis of deixis. A command type speech interface is complemented by unimodal reference analysis using ray-casting methods in [31]. Arangarasan and Phillips [2] focus on the referencing problem as an abstract task for object selection in multimodal VR interactions. Conceptual work regarding multimodal referencing based on *reference domains* is found by Landragin et al. [15], but only uses examples based on a static 2D domain.

Besides command-like interfaces, Thalmann [27] states that perception and interpretation are important for a Virtual Human located in a VE. These skills are particularly important for multimodal interfaces to provide dialogue style interactions. The related work now covers a wide area of publications, a comprehensive overview is given by Oviatt [23].

Architectural and technical issues of multimodal VR interactions are illuminated by Althoff et al. [1]. To process multimodal utterances for navigation tasks, they augment VRML [5] with specialized nodes and hence closely couple input processing with the continuously updated scene representation. The architecture of Touraine et al.'s framework [28] for multimodal VR interaction supports distribution of device services for motion trackers, speech or gesture detection, or multimodal integration on distinct hosts. They recognize that distribution—especially in closely coupled processes—raises a data synchronization problem for which they introduce two conceptually specialized network layers. The work by Kaiser et al. [13] describes a complete architecture for multimodal interaction. They focus on mutual disambiguation between input channels to improve interpretation robustness.

The majority of current VR, 3D graphics and real-time simulation frameworks mainly center around two design paradigms: Scene graphs are used to hierarchically arrange the simulated objects while behavior graphs allow the specification of simple environment animation and reaction. Both representations interconnect nodes or their attributes which encapsulate a certain functionality. For example, local transformations in scene graphs are expressed by interconnecting nodes—each having its own local frame of reference—via child_of-type relations while an animation of an object can be expressed in a behavior graph by interconnecting—or routing—the transformation attribute (or field) of an interpolation-node to a scene graph node's transformation field. In addition to animation, behavior graphs are used to process simple user interaction, e.g., by routing a button-pressed event to an enable field of an animation node. These techniques can be tracked back to toolkits like Open Inventor and its roots [26] and are nowadays found in many tools and standards, e.g., VRML97, Java 3D, or X3D. VR-frameworks like AVANGO [29] adopt these techniques and additionally provide methods for network distribution or scripting.

2.1 Discussion

The main challenges during the development of a UI toolkit for multimodal VR interactions can be separated into *methods* and *techniques*. The methods layer is concerned with the general approaches used for gesture and speech processing, multimodal integration and interpretation. For example, gesture recognition can be done with methods like template matching, neural networks, statistical methods based on Hidden Markov Models (HMM), and several others. The majority of research focuses on methods which now provides us with a multitude of choices for the tasks involved.

The techniques layer is concerned with the implementation of the respective methods. In contrast to Bolt's work and similar successors, processing multimodal utterances in immersive environments is deeply complicated by the tight interaction loop typical for VR. The indexical integrity of deictic gestures or of definite noun phrases (DNPs) like "*...the left thing...*" heavily depends on the scene configuration (maybe the object pointed at moves) or the user's changing view (which can invalidate "*left*" easily when the view is head-tracked). Multimodal expressions with spatial content, e.g., referential utterances or shape, size or movement descriptions, are anchored in the actual scene perceived at certain times.

The spatial dependency between utterances and the environment calls for a close coupling between the processing modules and the scene representation as motivated by [28] and [13]. One solution is the enhancement of behavior and scene graphs with specialized nodes which implement a given method and which latch and synchronize its application flow to the simulation flow as in [1]. But a close coupling to the simulation loop leads to problems which are summarized as follows: (1) Gesture and speech processing depend on sensor data with own sampling rates and requirements for consistent data whereas VR-setups usually take only the most current sensor sample into account. (2) Input processing will usually lag behind the current simulation step but needs access to lexical, conceptual and spatial information of the simulated scene not only for the current simulation time but delayed for the elapsed utterances' times. (3) Furthermore, the simulation rate can be unsteady and changing. In contrast to a close coupling as depicted this calls for a complete decoupling between input processing and the simulation.

3. OVERVIEW

Figure 2 depicts an overview of the UI framework's architecture illustrated here. It consists of three main modules for gesture processing (PrOSA–Patterns On Sequences of Attributes) [16] multimodal integration and analysis (tATN) [17], and a Knowledge Representation Layer (KRL) [18]. Persistent XML-based formats are defined for all three modules to provide a convenient system design as well as parameterization, reuse and exchange of once developed components. The architecture accounts for the boundary condi-

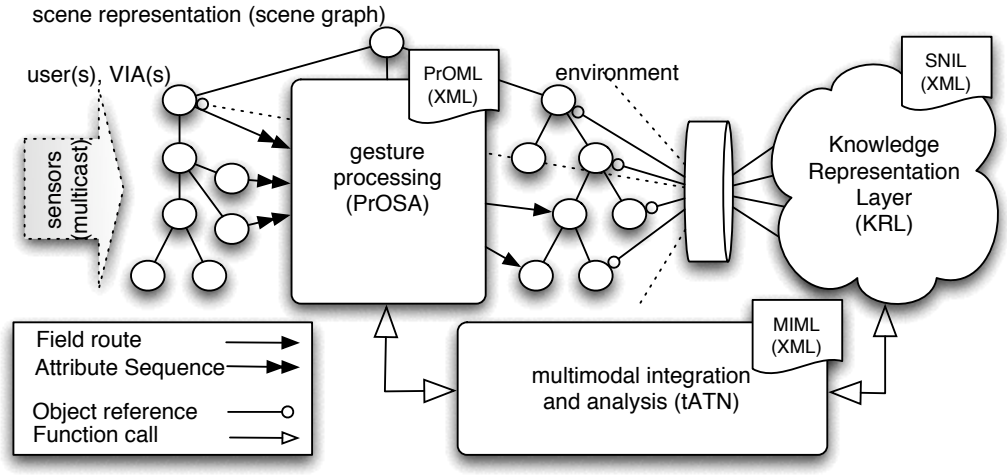

Figure 2: The framework's architecture. All core components for gesture processing, knowledge representation, and multimodal integration are configured and their internal representations are initialized using specialized XML derivatives. Four different communication techniques provide a modular and extendible system design while simultaneously taking module specific data processing characteristics into account (see text).

tions of the involved processes as illustrated in the last section and proposes techniques to decouple input processing while simultaneously providing close coupling where required.

All modules are centered around an enhanced scene representation which additionally represents the communicating instances (humans as well as Virtual Interface Agents) as hierarchical sub-graphs in the VR-system's scene graph. Ongoing body movements are reflected by special nodes used to model the skeletal frames. Sensor data is transfered to those nodes and accessible via an alternative routing mechanism which decouples gesture processing from the simulation loop while concurrently reflecting all scene changes due to their updated representation in the scene graph. This alternative routing mechanism uses so-called **Attribute Sequences (AS)** as connection-points in the gesture processing modules.

Gesture processing modules are evaluated by two complementing techniques. First, they can directly influence the environment via well-known behavior graph metaphors and route data via fields to the nodes in the scene graph, e.g., to move an object according to the movement of a hand. Second, they can be evaluated via a function call interface by the multimodal integration and analysis component.

Besides information of the ongoing user articulation, the integration component additionally requires access to semantic scene content. The KRL represents the environment's knowledge about objects and their features, possible (inter)actions as well as lexical bindings for these representations. The KRL establishes bidirectional references between objects in the scene graph and their semantic representation and hence provides mutual access from both sides.

The following interaction example will illustrate the framework's architecture and process flow throughout this article: The user is located in an immersive VE wearing two data gloves, 6DOF trackers at each palm and the head and a microphone. She utters the following multimodal expression: "*Turn* [pointing gesture] *the wheel* [begin rotation gesture] *like this* [end rotation gesture]". The system reacts as follows: After analyzing the opening verb phrase (VP) it processes the deictic content of the definite noun phrase (DNP) to select the appropriate object (see beginning of figure 1). Then it analyzes "*... like this*" accompanied by a mimetic rotation gesture which continues for some seconds after the verbal phrase ends. As soon as the system complements a possible interaction description using the matching verbal and gestural parts, it shifts into a continuous interaction state where the selected object is moved according to the user's gesture. Movement inaccuracy is smoothed and the resulting transformation is applied to the object until the gesture and hence the interaction stops. We will follow the processing of the example interaction starting with the input processing and its flow of information. This will lead to an illustration of the knowledge representation and final multimodal integration.

4. EMBEDDED PROCESSING

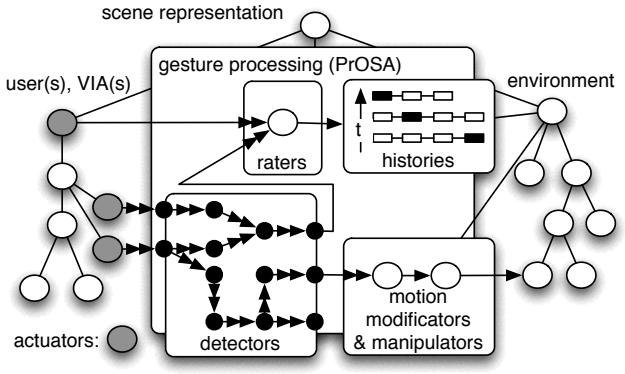

Figure 3: PrOSA components for gesture detection (detectors), gesture analysis (raters and histories), and manipulation (motion modificators and manipulators). Components are closely coupled into the scene representation but loosely coupled with respect to the frame rate due to their attribute sequence interconnections.

4.1 Actuators and Detectors

Actuators represent the lowest PrOSA level and provide an abstraction layer for sensor input. As specialized node types, their

Attribute Sequences decouple qualitatively annotated sensor data from the underlying hard- and software and embed these input sources into scene graph structures (see figure 3 left). Each actuator handles connections to input sources using one or more of so-called channels. Channels are technically bound to a certain computer communication facility, e.g., serial I/O or network communication (see figure 2 far left). Each channel can have it's proprietary sample rate and data format. One thread is assigned for each channel to read the incoming data packets from the hardware. This guarantees non-blocking operations of the actuators which are integrated into the VR-simulation and hence evaluated once during the render loop. Communication between the read-threads and the actuators is handled by common multiprocess-methods.
Channels are seldomly synchronized and hardware often lacks an external method for synchronization. Hence each channel incorporates functions for data interpolation and extrapolation. For each simulation step, actuators trigger their channels with a set of points in time for which the actuators request data. Actuators can either synthesize these times or they can choose a reference channel all other channels have to synchronize with using software interpolation.
Actuators may access multiple channels for micro-temporal sensor fusion of complementing data, e.g., to merge position and orientation data delivered by separate hardware devices into a combined 6DOF representation which is then normalized. This often involves transformation of 6DOF data since it is desirable to represent all spatial data using a common frame of reference for the following gesture processing components. This function is supported by the actuator's position in the scene graph which provides actuator data with respect to the simulated scene.
Attribute Sequences and Actuators conceptually establish a concurrent execution model that handles various independent input channels and which is latched into the given execution model of the closed simulation and rendering loop. Attribute sequences add a new stream routing facility which decouples multimodal processing from the simulation loop whereas the actuators latch the processing into the simulated scene. Attribute sequences are connection points between components which pass *multiple* and *timestamped* values per simulation step.
The example interaction is realized using four actuators. The first one provides speech input from a speech recognition system, the second one provides access to the 6DOF-tracker located at the head whereas the third and fourth one encapsulate the ongoing movement and shape of the left and right hands. Each of the latter two actuators feature two channels, one for 6DOF-information of the palms and one for bending angles delivered by two data-gloves. This information is fused to gain the position and orientation of the index fingers' fingertip using forward kinematics. The four example actuators provide access to the following data via connectable attribute sequences: 6DOF data of the head as an approximation of the view-direction and -orientation, the hands (palms) and of the index fingers' fingertips, the bending values of the fingers and finally the incoming words recognized by a speech recognition system. We are currently using a research prototype capable of speaker independent recognition of fluent speech but we have used of-the-shelf commercial tools (e.g., Dragon Dictate) as well, all with varying lexicon sizes between 300 and 5000 words. The recognition systems' specific differences are all hidden for following processing components by the actuator layer.
The PrOSA module provides a standard gesture detection method based on spatiotemporal preprocessing of movement data followed by a template matching approach. Several attributes are calculated from the output data provided by the actuator layer. Small **calculator nodes** with attribute sequence in- and outputs perform simple arithmetic and logic functions. The nodes build more powerful **detector nets** by interconnecting attribute sequences and routing their values as illustrated in the lower left center of figure 3 for an example detector net. Several different detector node types have been developed to build basic detector nets for gesture types frequently found during communication of spatial content. The functional decomposition of the gesture detection task into small interconnected nodes allows an easy adaption and modification (even during run-time) of detector nets until successful combinations are found which then can be reused for other projects.
Detector nets and the gesture processing tasks are described using PrOML (PrOSA Markup Language). PrOSA provides a development paradigm similar to those found in the VR-context by introducing nodes, (attribute sequence) fields and routes and hence should be easily comprehendible for developers familiar with concepts introduced by VRML or X3D. PrOSA provides a construction kit for the development of gesture-detectors based on spatiotemporal gesture features. Several projects using PrOSA have adopted its development paradigm and have contributed about 20+ gesture detectors. These include simple stroke, stop/go, pointing, pushing, rotating, or symmetry and complex iconic detectors.
Interface: The resulting attribute sequences of the detector layer are accessed via two methods. As illustrated in figures 2 and 3, they provide AS-route access carrying timestamped values which predicate, if—in the example interaction—a pointing or a rotational gesture is performed in general and if so, which is the pointing frame of reference and direction, the angular speed and the axis of rotation. As illustrated in figure 2, these values can additionally be accessed via a function layer where the functions constitute predicates like `pointing(t)` which denote if the respective gesture occured to a given time `t`.

4.2 Raters and histories

Actuator and detector nets provide data about gestural expressions whereas the surrounding dynamic scene represents the semantic context for the interpretation of multimodal utterances refering to position (deictic), configuration or shape (spatiographic/pictomimic), or movement (mimetic/kinemimic). A special node type called **rater** receives such quantitative data via attribute sequences and maps it once per simulation step to the rendered and hence perceivable scene to an intermediate more qualitative representation necessary during the analysis of the complete multimodal utterance (see figures 3 and 4). As nodes of the scene representation, raters are closely coupled to the scene representation and its objects while they are latched into the asynchronous gesture processing simultaneously.
As pointed out in section 2.1, multimodal interpretation will technically be executed delayed. Hence the *relative* scene configuration has to be reconstructed in temporal correlation to the different multimodal expressions. To gain this delayed access, so-called **histories** receive rater data and cache it, thereby—on a conceptual basis—acting as a kind of technical short-term memory needed for interpretation purposes. This principle is an acceptable trade-off between a usually technically impractical buffering of the whole scene and a complete evaluation of spatial attributes for every frame.
The pre-processing of the user's point of view is illustrated in figure 4 which is a detailed version of the top of figure 3. A spacemap rater is directly connected to an actuator output which provides the frame of reference of the user's view. It calculates distance (d) and direction-angular values for left/right (a) and up/down for all objects which is denoted by the appropriate axes in figure 4. This is

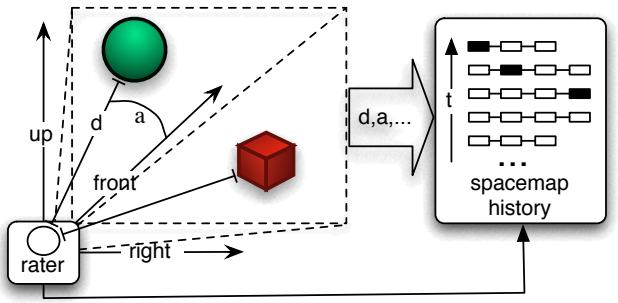

Figure 4: Pre-evaluation of location attributes. Objects are sorted according to a given view reference system with limited visibility. Culling determines the objects inside the defined view frustum to be stored in a spacemap history for delayed access.

done using the objects' positions and shape extensions with respect to a ray based at the actuator's frame of reference and elongated in the main rater direction. The output is routed to a history called spacemap since it stores pre-processed positions relative to a given frame of reference for multiple successive simulation steps. The shaded object in the history reflects the relative change of a for this object for some sample frames (each row representing the calculation result for one frame).

Raters sort the objects with respect to a given frame of reference and sorting criteria, e.g., closeness to a ray as it is for spacemap raters used for view and pointing analysis. The respective frames of reference and sorting parameters can be parameterized—even dynamically, e.g., to adopt results from cognitive research about perception of spatial reference systems. Multiple active raters are necessary to analyze user or object intrinsic references as well as references relative to anthropomorphic entities as can be seen during the interaction in the bottom right of figure 1. The figure illustrates a sequence during a clarification dialogue about a previously referenced object between the user and the anthropomorphic agent Max. Relevant parts which embody Max and which provide him with perception of his environment (including the user since she is represented in the same scene representation, see figure 2) are implemented symmetrically with the same concepts used for the processing of the user's multimodal utterances which demonstrates the framework's flexibility.

Interface: Rater provide direct field route access for each frame. Additionally, histories support access functions like `get-value-at(t)`, `get-next-value-at(t)`, or `get-average-of-interval(s,e)` parameterized by time `t` or interval `[s,e]`. The example interaction requires a view history, two pointing histories (left/right)—and a word history which buffers incoming words provided by the speech actuator.

5. INTERPRETATION AND INTEGRATION

5.1 Scene knowledge and access

Interpretation of multimodal utterances requires a semantic representation of objects, (inter)actions, their features as well as their lexical bindings which is provided using a Knowledge Representation Layer as illustrated in the right part of figure 2 and in detail in figure 5. To implement the KRL, we have developed a Functional Extendable Semantic Net (FESN). The FESN is capable of linking various required scene representations of diverse simulation modules to provide an integrated and semantically augmented scene representation. Figures 2 and 5 depict linking of the KRL to the scene graph as one typical representation of a graphics module. Other modules, e.g., for collision detection or dynamics simulation, can be linked likewise.

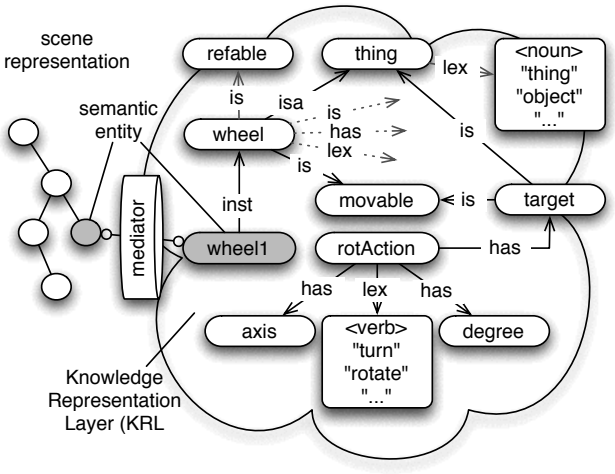

Figure 5: A section from the KRL: Relational interconnection between the wheel, a rotate action and its associated lexical information.

Figure 5 illustrates a fraction of the KRL required for the example interaction. On the left, the link between the KRL and the scene representation establishes mutual access from both sides between dedicated end-points in both representations. These end-points provide an entity-centered access to the semantic scene representation and establish a novel object model called **Semantic Entities** (SEs). The figure's left part depicts the target object `wheel1` linked to the respective counterpart in the scene graph (grey nodes). The KRL reveals that `wheel1` is of type `wheel` and hence of type `thing` and that it can be verbally addressed by their lexical bindings ("wheel", "thing", etc.) since the `lex` relation is defined to be inherited along the `inst-isa` taxonomy. Additionally, the KRL defines all instances of type `wheel` to be `movable` due to the `is` relation. The same is true for a `rotAction` concept which is defined to have an `axis`, a `degree`, and a `target` object. The latter is linked to the same `movable` concept as all wheels are, which maps possible interactions to certain objects. This information is evaluated during the interpretation step to validate if an interaction is permitted for an entity and to define required information to be provided to fulfill the operation.

The FESN supports an event system which propagates changes between different simulation modules automatically using a mediator layer (see figure 5 left). If modules require similar data, e.g., 6DOF for both dynamics and graphics, an explicit synchronization step collects proposed changes for the next simulation frame. The data is collected at dedicated FESN-nodes where simple filters allow the application to define which proposed change will be propagated back to the connected modules. An example would favor changes from the interaction components of a graphics system over dynamics to implement user interaction, e.g., dragging of objects. This semi-automatism is an effective data replication and synchronization method. It includes a layer that translates between different representations of equal data or attributes, e.g., to allow a transition between object centered positions in world coordinates for a physics engine and the scene graph counterpart represented as graph paths [9]. SNIL (Semantic Net Interchange Language), an

XML-based notation is used to define the scene content. It provides external link tags to existing file formats of other simulation modules and hence captures all required information using one central formalism.
Interface: Semantic Entities provide the central interfacing facility to the KRL using predefined query methods to read and update the knowledge base. For example, simple attribute queries like `has_attribute(attr)` can be called on SEs to find out about a specific feature of an entity. This KRL-interface capability of SEs is provided to the simulation modules by augmenting their object representations, e.g., using well known object oriented multiple inheritance methods where applicable. We have chosen to represent all relevant nodes in the scene graph as SEs, e.g., actuators provide information about the part of the user's body they represent or the hemisphere they belong to. Now, this information is accessed automatically by other components embedded in the same structure. For example, a spatial rater automatically finds entities it should sort by querying `has_attribute(refable)` (see `refable` concept in figure 5)on all objects during a scene graph traversal. Any changes made to underlying components, e.g., a change in their scene graph position will not influence the overall system's performance. This design provides a powerful abstraction layer for behavior specification via the KRL.

5.2 Multimodal integration

Several multimodal integration methods exist: Dialog and/or speech-driven approaches often predefined the place where a gesture could occur like in work by [4], [22], [14] and [19]. Often the 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 [14], [30] or use *feature structures* (attribute matrices that can be recursively combined) as in [10] and [12]. The structure values are connected by *constraints* which represent integration knowledge through context sensitivity between different values. A central *unification operation* tries to fill all possible structure attributes using collected input and matching other structures.
We are currently favoring a different concept which is more comparable to work in [11] and that is easily integrated into existing frameworks which provide script engines: A temporal augmented transition network (tATN) [17] which provides an XML-based format called MIML (Multimodal Integration Markup Language). A tATN accounts for temporal relations between utterances by an additional time-stamp *reached-at* register for each state `S`. Is `S` reached on grounds of an existing lexical constraint, this time-stamp is set to the beginning of the recognized percept. In absence of such a constraint it is set to the latest percept time that triggered the state change. Setting this special register is performed automatically and is not illustrated in the upcoming explanation of figure 6. There is evidence that the tATN and the unification approach are notational variants. In contrast to the latter, a tATN serializes possible combinations of (virtual) feature structures per state during creation-time and hence might reduce processing time during runtime in cases of unguided unification approaches.
Figure 6 illustrates one fragment of the tATN utilized for the example interaction which is the top-level processing module. The tATN implements a combined syntactic and semantic parse and integration process. It evaluates the required information sources for every active state and per simulation step even if the input events occurred in parallel. The arcs are separated into guarding conditions depicted as `if(<expr>)` and actions depicted as `-><function>`. The conditionals either parse complete sub-tATNs (see `if(ObjDsc)`)

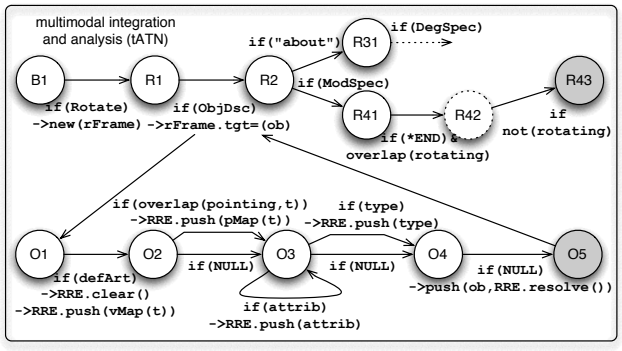

Figure 6: A tATN branch that incorporates lexical, gestural and application context information during a combined syntactic and semantic parse process. The arcs are labeled with constraints which guard the arc traversal. * denotes a look-ahead w.r.t. the current time, END denotes the end of the sentence. (see text).

or they access the PrOSA components via the provided predicate functions (see section 4) as well as the KRL as shown in figure 2. For example, the starting condition `if(Rotate)` at `B1->R1` performs an atomic test of incoming lexical percepts from the PrOSA word history. `if(Rotate)` compares the most recent word in the history with lexical entries in the KRL. Following figure 5, a successful match will be found for the `rotAction` which will result in the instantiation of a rotation frame `->new(rFrame)` which requires a rotation axis and degree as well as a `movable` target object.
State `R1` branches to the sub-tATN `ObjDsc` to parse object descriptions. During its traversal, the tATN incrementally feeds a reference resolving engine (RRE) (see `->RRE.push(<expr>)`) with semantic content related to the respective guarding condition. Besides a simple intersection set approach, we have recently developed an open constraint satisfaction solver [24]. Referential expressions may consist of various constituents with different selectivity which the RRE sees as constraints to satisfy with respect to a given frame of reference. A constraint specifies property values including information about objects being in the view (see `->RRE.push(vMap(t))` at `O1->O2`), objects being pointed at or referenced by other gestures (see `->RRE.push(pMap(t))` at upper `O2->O3`), as well as features derived from the analyzed speech (types, shapes, colors, positions, functions, etc. see `->RRE.push(type)` at upper `O3->O4` and `->RRE.push(attrib)` at `O3->O3`). The required parameter `t` is initialized automatically with the time-stamp found in the *reached-at* register of the left state. Ambiguity in multimodal utterances is handled by using fuzzy constraints instead of boolean values. In addition, a hierarchical ordering of constraints allows them to express different selectivities during the evaluation. The fuzzy approach implicitly provides a method for mutual disambiguation between modalities since each additional constraint (no matter which modality it is based on) refines the set of possible targets.
The traversal of the sub-tATN `ObjDsc` illustrates the integration of simultaneous percepts using temporal relations like `overlap(pointing,t)` at upper `O2->O3` which tests if a pointing history has a positive entry at its time `t`. In detail, such transitions are decomposed into two parallel transitions where one is guarded by the first constituent of the temporal comparison, e.g., the word. The other transition is guarded by the second constituent. Both will lead

to different end-states, each carrying its own respective *reached-at* register which then can compared with each other. Hence, the tATN concept has to provide multiple active branches, a feature also useful if constraints have not been mutually exclusive at previous edges. Traversing `ObjDsc` to the final state `O5` will finally query the RRE (`RRE.resolve()`) and will push its result into an object register (`ob0`) which then fills the target attribute of the rotation frame (`->rFrame.tgt=(ob)` at `R1->R2`). After parsing "like this" at the `if(ModSpec)` condition, state `R41` test for an overlapping rotation gesture and a special look-ahead token `*END` which signals an empty word history. If `R42` is reached, the rotation frame will be executed since the gesture leading to this state is semantically defined by the KRL to deliver a missing rotation axis and degree. Hence, the system shifts into a continuous interaction mode (illustrated by the dotted circle around state `R42`).

The continuous interaction maps imprecise user movements to precise transformations of the object in real-time. This simultaneous gesture translation is again enabled via attribute sequence and field route connections between specialized node types called **motion-modificators** and **manipulators** (see lower right of figure 3). This close coupling of the gesture interaction proved to be necessary for interactions needing continuous feedback where a loose coupling of a distributed architecture introduced a significant lag into the system. This was due to the closed render-loop being able to process incoming messages only at certain times which couldn't be compensated using multiprocessing, hence it was a synchronization issue. Embedding the gesture processing into the simulation representation worked perfectly well in that case.

Motion-modificators are implemented using the same calculator nodes like the detector nets, which emphasizes the usefulness of reusable components. During the example interaction, a rotation motion modificator temporarily connects to the outputs of the detector net as illustrated in figure 3 from which it receives the averaged normal of the movement plane and the angular speed. The incoming values are compared to a raster of possible allowed values and the closest ones are chosen, resulting in filtered output values having a fixed stepping. These values are now routed via fields to an ordinary manipulator node which applies the values to the wheel's own local transformation: The wheel rotates according to the user's gesture. This mode is controlled by the gesture and can only be left when the rotation gesture is discontinued as defined by the `if not(rotating)` condition between `R42->R43`. The interaction has successfully been processed and the system's components a reinitialized.

6. SUMMARY AND OUTLOOK

This article presented a UI framework for multimodal interaction in VR. The design of its architecture proposes conceptual solutions for the integration of multimodal processing tasks into simulation systems. Its design takes the characteristics of immersive real-time simulations into account which require specific techniques for temporal synchronization and data access and replication. Temporal synchronization is achieved using the Attribute-Sequence concepts of nodes embedded into the simulation system's scene representation. Attribute Sequence interconnections provide an advanced behavior graph that interweaves basically independent and decoupled process flows—simulation loop and multimodal processing—which have to be closely coupled at certain defined steps during their cycles. Data access and replication is approached using two complementing methods: First, a Knowledge Representation Layer interconnects objects of the simulation's scene representations with entities in a knowledge base using Semantic Entities as a novel object model. The KRL automatically synchronizes redundant data through its mediator concept to instantly reflect changes in one representation by the other, e.g., to match a changing RGB value in the simulation system with a different color concept in the KRL. Second, specialized node types access both representations—the simulation system's as well as the KRL's—which enables a modular decomposition of tasks into modules implemented by certain nodes. Having access to both representations, these modules can now automatically match their functions to appropriate target nodes as defined via the KRL. For example, raters embedded into the scene representation automatically scan the scene for entities with a given semantic property as defined by the KRL where histories buffer rater results for delayed access during the multimodal integration and analysis.

The framework provides several off-the-shelf methods for the required processing modules: Specific approaches for input and gesture processing (PrOSA), multimodal integration (tATN), reference resolving (RRE) and Knowledge Representation Layer (FESN, Semantic Entities) were proposed. These approaches each represent possible solutions for the particular functions required during multimodal processing which proved to be useful under real-time constraints. Here, the framework's open architecture is designed to support implementation of alternative methods using its provided module integration and interconnection techniques. Persistent XML formats for the gesture processing components (PrOML), for the semantic scene content (SNIL), and for the multimodal integration rules (MIML) (see figure 2) provide a high-level interface for the definition of multimodal interfaces in simulation systems. By enhancing well proven development methods for virtual environments with extensible and interconnectable components defined by persistent description formats, the framework provides developers with reusable, interchangeable, and adaptive components and it suggests its own design paradigm for the development of multimodal interactions for such environments.

The framework is now the foundation for various projects including virtual prototyping applications, empirical studies on human cognition, research on gesture semantics and representation, or on interaction with our articulated humanoid communication partner (see, e.g., figure 1). These projects continuously use and extend the framework, e.g., by fine-tuning existing or designing new gesture detectors, by extending multimodal grammars, or by creating new interaction types. We have lately conducted a small user survey to gain a qualitative tendency for the framework's usefulness. 8 Developers from 10 projects—with well distributed development skills— were asked to rate the framework according to five evaluation criterias: Extensibility, performance, reusability, customization and overall benefit in the range from 0 (poor) to 5 (very good). In general it showed a positive tendency. The lowest average score per criteria was 3.6 (development time improvement, the participants criticized insufficient documentation in followup questionings) and the highest was 4.4 (for reusability and extensibility) with an average of 4.1. The inter-criteria variance was quite low and may be explained by the varying participant's development skills.

Future work is targeted at further enhancing the modularity of the system, e.g., to provide high-level interface methods to link its modules to components from other applications areas not necessarily associated with VR simulation systems. For example, ubiquitous environments and virtual environments technically have quite similar characteristics, where the latter usually requires a rendering component for the scene. Additionally, if the framework meets the VR conditions, it can as well be used for applications with minor technical requirements, e.g., non immersive desktop systems. Several approaches aim at supporting new methods for the components off-the-shelf. This includes the development of a rule-based

AI-representation for the KRL as well as a neural network layer which will support the KRL as well as the matching stage of the gesture processing.

Implementation and acknowledgment: The core framework is implemented in C++. Additionally, all objects have bindings to the SCHEME scripting language. Its scene graph oriented concepts currently use AVANGO [29] as the target VR platform. Several persons contributed to the current implementation, including Peter Biermann, Lars Gesellensetter, Malte Schilling, Guido Heumer and Thies Pfeiffer. The framework's development was supported by the Ministry of Science and Research (MWF) of the Federal State of North Rhine-Wesfalia in the "SGIM project" and by the German Research Foundation (DFG) in the Collaborative Research Center (SFB 360) and the "Virtuelle Werkstatt".

7. REFERENCES

[1] F. Althoff, G. McGlaun, B. Schuller, P. Morguet, and M. Lang. Using multimodal interaction to navigate in arbitrary virtual vrml worlds. In *Proceedings of PUI 2001*, 2001.

[2] R. Arangarasan and G. N. J. Phillips. Modular Approach of Multimodal Integration in a Virtual Environment. In *Proceedings of the Fourth IEEE International Conference on Multimodal Interfaces ICMI'02, Pittsburgh, Pennsylvania*, pages 331–336. IEEE, 2002.

[3] 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.

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

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

[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. Cohen, D. McGee, S. Oviatt, L. Wu, J. Clow, R. King, S. Julier, and L. Rosenblum. Multimodal interactions for 2d and 3d environments. *IEEE Computer Graphics and Applications*, pages 10–13, 1999.

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

[9] G. Heumer, M. Schilling, and M. E. Latoschik. Automatic data exchange and synchronization for knowledge-based intelligent virtual environments. In *Proceedings of the IEEE VR2005*, pages 43–50, Bonn, Germany, 2005.

[10] 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*, pages 624 – 630, 1998.

[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] E. Kaiser, A. Olwal, D. McGee, H. Benko, A. Corradini, X. Li, P. Cohen, and S. Feiner. Mutual disambiguation of 3d multimodal interaction in augmented and virtual reality. In *Proceedings of the 5th international conference on Multimodal interfaces*, pages 12–19. ACM Press, 2003.

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

[15] F. Landragin, N. Bellalem, and L. Romary. Referring to Objects with Spoken and Haptic Modalities. In *Proceedings of the Fourth IEEE International Conference on Multimodal Interfaces ICMI'02, Pittsburgh, Pennsylvania*, pages 99–104. IEEE, 2002.

[16] M. E. Latoschik. A gesture processing framework for multimodal interaction in virtual reality. In A. Chalmers and V. Lalioti, editors, *AFRIGRAPH 2001, 1st International Conference on Computer Graphics, Virtual Reality and Visualisation in Africa, conference proceedings*, pages 95–100. ACM SIGGRAPH, 2001.

[17] M. E. Latoschik. Designing Transition Networks for Multimodal VR-Interactions Using a Markup Language. In *Proceedings of the Fourth IEEE International Conference on Multimodal Interfaces ICMI'02, Pittsburgh, Pennsylvania*, pages 411–416. IEEE, 2002.

[18] M. E. Latoschik and M. Schilling. Incorporating VR Databases into AI Knowledge Representations: A Framework for Intelligent Graphics Applications. In *Proceedings of the Sixth International Conference on Computer Graphics and Imaging*. IASTED, ACTA Press, 2003.

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

[20] 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.

[21] 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. 1993.

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

[23] S. Oviatt. *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*, chapter Multimodal Interfaces. Lawrence Erlbaum Assoc., 2003.

[24] T. Pfeiffer and M. E. Latoschik. Resolving Object References in multimodal Dialogues for Immersive Virtual Environments. In *Proceedings of the IEEE Virtual Reality conference 2004*, pages 35–42, 2004.

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

[26] P. S. Strauss and R. Carey. An object-oriented 3D graphics toolkit. In *Computer Graphics*, volume 26 of *SIGGRAPH Proceedings*, pages 341–349, 1992.

[27] D. Thalmann. The virtual human as a multimodal interface. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 14–20. ACM Press, 2000.

[28] D. Touraine, P. Bourdot, Y. Bellik, and L. Bolot. A framework to manage multimodal fusion of events for advanced interactions within virtual environments. In *Proceedings of the workshop on Virtual environments 2002*, pages 159–168. Eurographics Association, 2002.

[29] H. Tramberend. A distributed virtual reality framework. In *IEEE Virtual Reality Conference*, pages 14–21, 1999.

[30] 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.

[31] E. Zudilova, P. Sloot, and R. Belleman. A Multi-modal Interface for an Interactive Simulated Vascular Reconstruction System. In *Proceedings of the Fourth IEEE International Conference on Multimodal Interfaces ICMI'02, Pittsburgh, Pennsylvania*, pages 313–318. IEEE, 2002.