Knowledge in the Loop: Semantics Representation for Multimodal Simulative Environments

Marc Erich Latoschik, Peter Biermann, and Ipke Wachsmuth

AI & VR Lab, University of Bielefeld

Abstract. This article describes the integration of knowledge based techniques into simulative Virtual Reality (VR) applications. The approach is motivated using multimodal Virtual Construction as an example domain. An abstract Knowledge Representation Layer (KRL) is proposed which is expressive enough to define all necessary data for diverse simulation tasks and which additionally provides a base formalism for the integration of Artificial Intelligence (AI) representations. The KRL supports two different implementation methods. The first method uses XSLT processing to transform the external KRL format into the representation formats of the diverse target systems. The second method implements the KRL using a functionally extendable semantic network. The semantic net library is tailored for real time simulation systems where it interconnects the required simulation modules and establishes access to the knowledge representations inside the simulation loop. The KRL promotes a novel object model for simulated objects called *Semantic Entities* which provides a uniform access to the KRL and which allows extensive system modularization. The KRL approach is demonstrated in two simulation areas. First, a generalized scene graph representation is presented which introduces an abstract definition and implementation of geometric node interrelations. It supports scene and application structures which can not be expressed using common scene hierarchies or field route concepts. Second, the KRL's expressiveness is demonstrated in the design of multimodal interactions. Here, the KRL defines the knowledge particularly required during the semantic analysis of multimodal user utterances.

1 Introduction

VR tools have historically been built around the graphics system as the major simulation module. Nowadays, they additionally support the VR design process with a variety of features. These include support for application or behavior graphs via field-routing, input and output device abstraction, networking as well as scripting capabilities. Specifically required features, like a thorough or approximated physical simulation or gesture and speech processing for novel—multimodal—interaction methods (see Fig 1), have to be integrated using the extension methods provided by the VR and simulation tools utilized. Here, a promising research direction strives for a conceptual combination of VR and AI related methods to support the design of Intelligent Virtual Environments (IVEs) (Luck & Aylett, 2000).

A. Butz et al. (Eds.): SG 2005, LNCS 3638, pp. 25 – 39, 2005.

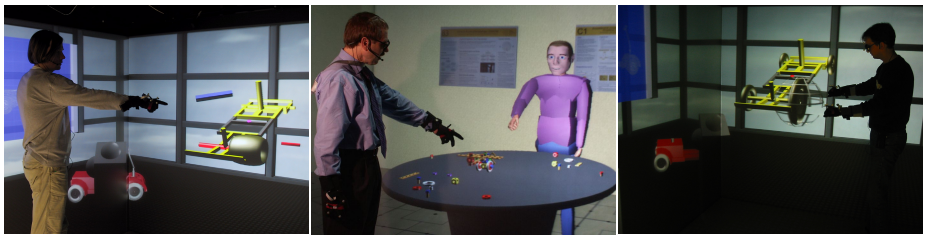

Fig. 1. Multimodal interaction in knowledge supported construction tasks. Left: A user selects a wheel and connects it to a complex chassis ("*Take* [pointing gesture] *this wheel and connect it there…*") (Latoschik, 2001). Middle: The user and the artificial communication partner MAX agree on the referenced part on the table (Kopp *et al.*, 2003). Right: The user scales a previously connected part which keeps its attributes, here the wheel's roundness (Biermann & Jung, 2004).

AI and VR concepts have been combined in a variety of research projects using customized and application specific integration methods. In the last few years, several approaches have aimed at a more general way of a conceptual and technical integration of AI techniques into simulation based systems (Cavazza & Palmer, 2000; Luck & Aylett, 2000; Peters & Shrobe, 2003; Soto & Allongue, 2002). An integration based on a common representation layer as proposed in (Cavazza & Palmer, 2000) offers several advantages regarding adaptability and reusability. Its content can be made persistent using an external file format where AI, graphics, physics and other simulation related content are coequally expressed in a common format.

A fundamental integration of AI and VR provides the potential for a wide range of possible applications including heuristic approximations of—e.g., physical—simulation features and advanced multimodal interaction setups. For example, we have enriched our VEs with multimodal instruction type interactions as well as with a humanoid communication partner called MAX (Kopp *et al.*, 2003). Here, lexical and semantic content about the simulated scene is mandatory during both, the analysis of the user's and the generation of Max's multimodal utterances.

1.1 Knowledge Representation in VR

This article presents a unified and comprehensive method of knowledge integration and access in VR-based simulation systems. This method will be illustrated in two example areas: knowledge supported virtual construction and multimodal interaction. On shallow examination the two areas seem to differ significantly. In a closer examination, we will illustrate how necessary generalizations made from both, the example applications' data representations, as well as from the VR-simulation specific data representations lead to a common knowledge layer capable of a high-level description of advanced simulation features.

We start by introducing the used terminology and by a closer analysis of the built-in semantics of existing representations of generic VR-simulation system. From a knowledge based view, objects and structures of a simulation system can be defined by *entities*, *attributes*, *relations,* and *concepts.*

Entities represent simulated objects. Attributes describe certain feature-value pairs of entity representations. For example, simple well known attributes for the graphics part are (RGBA) diffuse colors of the utilized lighting model or 9 float values for transformation specification as illustrated in the upcoming Fig. 2 and Fig. 3 by the 9DOF nodes. Each of these nodes carries 9 floats which are associated to a given entity, e.g., 2_hole_rod or hole1 in Fig. 2, to specify its transformation. All atomic data about entity features are eventually described as attributes, from simple color values, diameters of holes, entity masses up to functional features attributes like is-scalable or is-connectable.

Relations generally describe n-ary predicates of attributes, entities and concepts. Concepts represent categories, semantic circumscriptions of objects and attributes of the regarded target domain as illustrated in Fig. 2 and Fig. 4 for the base node of the 2_hole_rod entity which instantiates a ROD concept. Concepts are attribute and entity descriptions, patterns or *classes*. A typical known relation of scene graph systems is the part_of relation that defines the scene hierarchy and grouping behavior.

With respect to the example domain, the term *part* denotes non-decomposable but modifiable entities used in the construction process. Parts consist of *sub-parts* which relate to semantically self-contained sections of parts. A sub-part is defined (1) by its position relative to a part's frame of reference and (2) by a set of sub-part specific attributes which describe the sub-part's type. Sub-parts can not be instantiated without a part to which they are bound conceptually– they can not be deconstructed during user interaction.

1.2 Built-In Semantics of VR-Simulation Systems

The semantics of attributes and relations commonly used in simulation systems is defined procedurally. The interpretation of attributes and relations is defined by the underlying technical processes of the simulation modules. For example, color-attribute values are used to calculate pixel-shadings with respect to utilized lighting model, and part_of scene graph relations define the accumulative multiplication of matrices on the matrix stack. This operational and fixed semantics limits the representations available to the predefined rendering tasks and complicates or even inhibits their utilization for different application specific representations. The expressiveness of the scene graph related part_of relation as well as those of application graphs built from field route connections is strictly defined by the procedural semantics of the simulation system.

As a consequence, additional representations are necessary for reasonably complex applications since the existing features of the simulation engines are not expressive enough. This often results in purpose-built solutions which lose the declarative expressiveness, the reusability as well as the flexibility of the representation methods provided by the simulation engines. For example, a common solution in field route based systems is to define new node types which receive certain data and manipulate this data using specialized algorithms. Most VR-based simulation systems include methods for such extensions, e.g., VRML (Carey *et al.*, 1997) supports this approach using the built-in PROTO feature.

In the worst case, none of the built-in extension features are expressive and powerful enough for some applications. In that case, external modules are often loosely coupled to the simulation system and data is exchanged between them using standard interprocess communication (IPC) facilities. This in turn requires special purpose external synchronization and data-replication which complicates application development significantly or even prevents expandability and reusability of systems' components. Furthermore, in certain areas a loose coupling can in fact be insufficient. For example, continuous user interaction, e.g., dragging of an object, usually requires a high responsiveness which can not be guaranteed at all times using loose coupling without concurrently using IPC blocking behavior.

2 Simulation Knowledge Representation Layer

Our goal is a common Knowledge Representation Layer (KRL) which contains VR-simulation specific as well as application tailored knowledge. The subsequent explanations presuppose a simulation system which at least provides scene and behavior graph structures as for instance offered by the AVANGO toolkit (Tramberend, 1999). Hence, relations defined in the knowledge layer first of all have to represent the target system's representational features, and similar steps are necessary for other target systems.

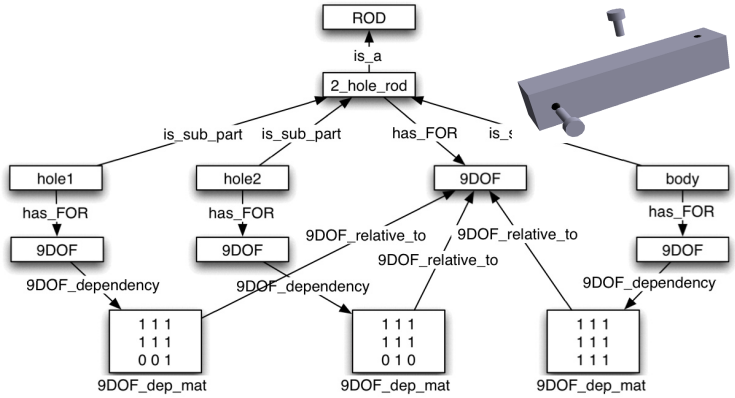

Fig. 2. Knowledge representation of an example part (a rod with two holes, see upper right) which supports intelligent scaling operations

sg_part_of (scene graph parent/child): Transitive directed relation denoting grouping behavior of scene graph nodes. Implies accumulative multiplication of existent matrix attributes found at nodes followed in the given relation direction.

fr_connect_to (field route connection): Transitive directed relation denoting value propagation of attributes in the given relation direction.

f_control_to (field reference): Directed relation denoting referential (not routed) read/write access to fields of target nodes by specialized source nodes.

Additional relations are defined to express the application specific knowledge. The following relations and their semantics support virtual construction tasks that group parts to aggregates and specify geometric dependencies between (sub-)parts:

is_sub_part: Transitive directed relation denoting the association of a sub-part to a part.

is_a: Transitive directed relation denoting a subsumption hierarchy for part concepts. is_a implies inheritance behavior of certain attributes, e.g., lexical information, of parent concepts.

has_FOR: Directed relation denoting the association of a frame of reference (FOR) to a given concept, e.g., a part or a sub-part.

9DOF_dependency: Transitive directed relation denoting a geometric dependency between two 9DOFs (9 Degrees of Freedom) as parameterized by a **9DOF_dep_mat** concept which defines the dependencies (see following sections).

9DOF_relative_to: Transitive directed relation denoting the relative position, orientation, and scaling of a 9DOF with respect to a given 9DOF frame of reference.

Fig. 2 illustrates a segment of the resulting knowledge structure which supports intelligent scaling operations for a rod with two holes as sub-parts which are defined to behave independently during a scaling operation of the main part, e.g., to maintain the holes' roundness during scaling operations. All parts and sub-parts have associated 9DOF frames of reference which define their position, orientation and scaling using the has_FOR relation. This ensures that grouping and transformation are expressed independently from each other. The sub-parts' FORs are defined to be relative to the main part's FOR via a dependent mapping defined by the 9DOF_dependency which parameterizes the 9DOF_relative_to relation using the 9DOF_dep_mats.

The semantics of this representation is as follows: The 3x3 dependency matrix of a 9DOF_dep_mat defines how the factors for position (first matrix row entries), rotation (second row) and scaling (third row) are concatenated following the algebraic semantics of the 9DOF_relative_to relation. In its plain assertion between two FORs, the 9DOF_relative_to defines well known multiplication of homogenous coordinate representations which would express common scene-graph behavior. In contrast, the chosen representation allows an extensive parameterization of the concatenation type of two linked FORs. Fig. 3 illustrates how arbitrary

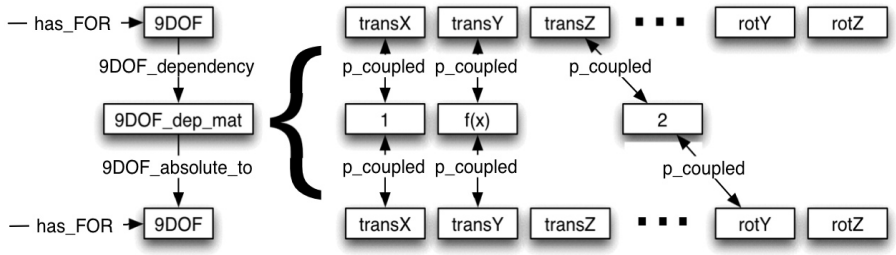

Fig. 3. Parameterized coupling of attribute values which are calculated according to the algebraic rule as defined by the embracing relation, here the 9DOF_absolute_to relation

parameters—constant values as well as functions—can be defined to modulate the algebraic effect or calculation rule of the active relation which couples two attributes.

The free interconnection of attributes even allows coupling between different geometric attributes or DOFs e.g., to have one part to rotate if another part translates or to match one part's scaling by a rotation of two other parts if two dependency matrices are used.

The two zeroes in the last row of the left and the middle 9DOF_dep_mat in Fig. 2. represent a missing p_coupled relation and hence define partial blocking of the 9DOF_relative_to semantics which defines a parent-child relation between the main part and the holes. This suppresses the consecutive impact of parent part's total scaling and only scales hole1 in the z- and hole2 in the y-direction (the principal axes of the holes' main directions).

3 Interaction Knowledge Representation Layer

The KRL is not limited to the representation of geometric dependencies as motivated for the virtual construction task. Its overall goal is to support application specific representation models as well as commonly required VR-related modeling tasks. This includes high level representations of entity data and structures for the variety of involved software modules, e.g., for the graphics, physics and the interaction components.

The idea of a semantic representation is in fact strongly inspired by the intention to utilize multimodal—gesture and speech driven—interactions in VEs. Processing of multimodal utterances can be roughly divided into several phases: Speech and gesture detection, semantic speech and gesture analysis, multimodal integration and pragmatic analysis. During processing, several of these phases frequently access semantic content from redundant data representations of other simulation modules. Here, a unified KRL partly solves the multiple database problem.

A major design goal for the knowledge representation layer is to support semantics, necessary during interaction. This includes, e.g., mappings between conceptual and lexical data for a variety of concepts and attributes. These concepts do not only represent perceivable entities and their features in the artificial world but also abstract concepts, e.g., holes in a part or actions a user can carry out.

The necessity for the representation of semantics is evident for the semantic analysis phase which has to map lexical and gestural expressions to conceptual knowledge. Fig. 4 presents another view into the semantic network which represents the example two-holed rod. The grey nodes illustrate instantiation of a two-holed rod. The conceptual definition of a two-holed rod (grey rectangular nodes) is used as a template for the actual instance (grey oval nodes). Instances represent the existing entities of the simulation. Their inheritance of concepts and attributes as defined by their conceptual templates is customizable. If they are not inherited during instantiation, their inheritance behavior is defined per relation following the connecting relations (here inst_of and is_a). Specialized negating relations (pattern <x>not_<y>, e.g., hasnot_feature) provide a method to locally override the default inheritance behavior.

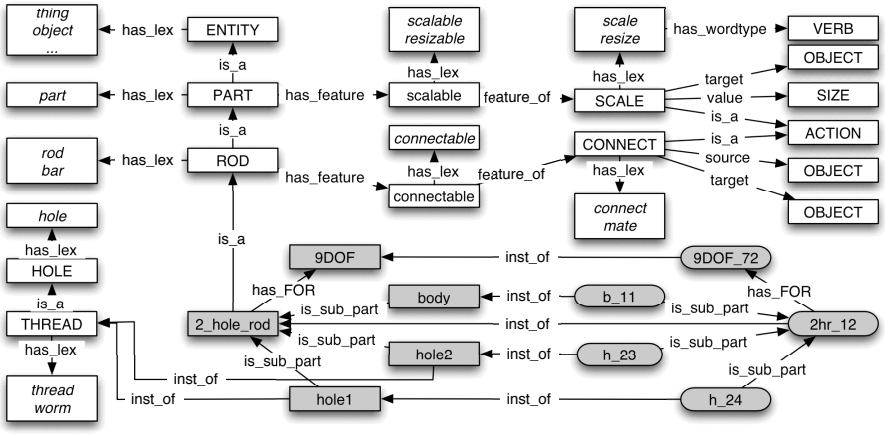

Fig. 4. Conceptual and lexical knowledge representation of the two-holed rod

Fig. 4 illustrates the interconnection between entity features which define certain simulation behavior, e.g., whether an entity is scalable, and the representation of the according user action by the SCALE, CONNECT and ACTION concepts. These interaction concepts further define their required conceptual knowledge to be fulfilled, e.g., a required target or a required interaction parameter. Where linguistic knowledge is given by linking concepts to lexical counterparts, the semantic analysis processing module is automatically enabled to map lexical information to conceptual knowledge which then can be instantiated during the processing.

For example, a connection of the screw and the two-holed rod in the virtual construction task can be accomplished in two ways. The direct manipulation way is to drag the screw. When the screw collides with the rod, the best fitting ports (here the holes) are heuristically determined and appropriate modules will be activated to simulate the connection (see subsequent sections).The second way is by using multimodal interaction, e.g., by saying: "*Put* [pointing gesture] *that screw in* [pointing gesture] *this hole.*" (see, e.g., Fig. 1. left). Focusing on the linguistic part, input is analyzed by mapping the incoming words to the lexicon which is defined by the target concepts of the has_lex relation in Fig. 4. By backward traversing these relations during the parse process (Latoschik, 2002), the matching base concepts are retrieved from the KRL.

The conceptual knowledge is now used according to its semantics. For example, verbs found in imperative speech such as "*Give me a…*"will be looked up in the lexicon. Traversing the has_lex relation will retrieve the respective concept. If this concept is then identified as an ACTION, e.g., by traversing the type hierarchy, matching interaction frames will be retrieved and instantiated which define required interaction information, e.g., the required target OBJECT instance(s). In addition to the illustration in Fig. 4, the actual existing knowledge base decomposes OBJECT concepts into their substructures, relates them to the two existing type hierarchies, and augments the linguistic structures with syntactic information.

These interaction frames, their concepts and attributes, are used as templates which are filled during input processing. For example, if the template requires one or more objects, the parse process will feed a reference resolving engine with the conceptual information necessary to identify one or more target entities (Pfeiffer & Latoschik, 2004). Since the KRL interconnects instances with the data representations of other simulation modules, this data, e.g., a quantitative RGBA value for a color attribute will be processed in the same way. Finally, completed interaction frames trigger the desired interaction.

4 Implementing and Applying the KRL

To implement the KRL, a knowledge representation tool, FESN (Functionally Extendable Semantic Network) (Latoschik & Schilling, 2003), has been developed. The FESN offers a specialized and adaptable semantic net formalism which is implemented as a C++ library and which has special features targeted at its utilization in VR simulation systems and for diverse application specific representation tasks.

Attribute augmentation of concepts: FESN nodes can carry additional attributes and values which allows a seamless transformation of a target system's representations into the FESN.

Functional extensibility: Flexible relation semantic. New relation types can be added easily. The semantics of relations is expressed by functions added to the relations.

Built-In event system: Changes of attribute values and the network's structure are monitored to enable automatic processing of changes submitted by simulation modules.

Built-In event filter: Concepts (nodes) of the FESN can be associated with multiple external attribute sources of the same attribute. A parameterized filter concept allows automatic evaluation and selection of concurrent—possibly conflicting—value changes.

External XML representation: The FESN supports SNIL, the Semantic Net Interchange Language, as an XML based external representation. This provides a convenient way to define and modify knowledge layer content (see Fig. 5).

Furthermore, the low level implementation of the FESN as a C++ library allows several performance optimizations which conserve processing resources in contrast to commonly found high-level (e.g. PROLOG or LISP based) semantic net realizations. This is particularly important for the FESN's utilization in interactive real-time simulations.

Using the FESN as the base formalism, the KRL completely defines all data and knowledge required by a simulation system in a unified representation, including graphics, physics, audio or even AI and interaction content. The KRL's content is applied to simulation systems in two ways. The first, uncoupled, method transforms knowledge bases defined by SNIL into representations compatible with the simulation

modules of the target system. Several of these modules support external representations. Some of them support proprietary XML based formats, e.g., we have previously developed an XML based format for a simulation module which handles variant parts: VPML (Variant Part Markup Language) (Biermann & Jung, 2004). The required mapping between source and target representation is conveniently achieved via XSLT processing where the mapping rules only have to be statically defined once.

In contrast to the uncoupled method, the coupled method embeds the FESN as a separate module directly into the simulation system's process space and latches knowledge access into the simulation loop(s) of all simulation modules which share this process space. In such setups, the FESN acts as a central knowledge base and monitoring instance. Its event system provides a method to control and filter value changes of attributes which might be proposed by several modules concurrently, e.g., for the 9DOF attribute.

4.1 Semantic Entities

The coupled method provides two ways of knowledge access: First, arbitrary FESN queries can be directly called on semantic net structures which are mapped into the process space of the calling modules. Second, modules can use an object centered KRL-access to dedicated entities in their own proprietary object representation. In effect, each of the simulation module specific entities has a corresponding counterpart represented in the KRL. By augmenting the module specific entity representation with an FESN-interface—for example, using object oriented multiple inheritance schemes—the entities' semantic data is directly accessible by the given simulation module. This architecture leads to a novel object or entity model we call **Semantic Entities.** Semantic Entities link proprietary entity representations with the correspondent instances of the knowledge layer and provide a standardized way of accessing the KRL.

The uniform KRL-access via Semantic Entities allows for an increased and powerful modularization of simulation systems. Typical architectures for implementing some specific functionality for a given simulation utilize so-called engines. Engines are dedicated modules in scene graph based systems which implement a certain function and which apply their results by accessing target nodes and attributes directly or by propagating attribute changes using a behavior graph. Here, target nodes and attribute-routes have to be specified by the application programmer specifically, be it for a simple interpolator engine or for an advanced collision and dynamics engine. Using Semantic Entities, information about object animation and manipulation specification is conveniently defined declaratively using the KRL. By querying the KRL via the Semantic Entities in the engines' or modules' process space, requested object features, like if an object is movable, or collidable, or any other feature a specific engine requires, can directly be accessed.

For example, we have developed several components for multimodal interaction which completely rely on Semantic Entity access. Dedicated engines preselect and sort possible targets' Semantic Entities in the scene representation according to users' gestures like view and pointing direction. This set of possible targets is then further

restricted to objects which satisfy the semantic interpretation of type or attribute specifications, e.g., during the processing of definite noun phrases like *"…the blue rod…"*: A KRL retrieval for the word *"…rod…"* will find the ROD concept by following its has_lex relation in Fig. 4. This result will restrict the preselected set of Semantic Entities to entities of the required type, e.g., to include the 2hr_12 instance in Fig. 4 which is of type ROD following the transitive inst_of and is_a relation. The semantic interpretation engine uses Semantic Entities for object centered KRL access to map the utterance's meaning to the semantic scene description which includes representations of the user and possible interaction partners as in Fig. 1.Other components evaluate the construction specific knowledge of entities and automatically instantiate required simulation modules which implement a given entity feature as will be explained in the following sections.

The Semantic Entity object model greatly simplifies the development of Virtual Environments. It promotes modularization, easy adjustment, and reuse of simulation components. Components access specific entity features via Semantic Entities. Additionally, the components themselves can be represented in the KRL to provide an automatic mutual matching of tools and target objects since they share the same representation.

4.2 Mapping Knowledge to Target Systems

Besides KRL-access, both knowledge application methods have to map the FESN representations to a target system's internal representation. This mapping transforms the modeled knowledge into structures readable and interpretable by the respective simulation modules. For the example two-holed rod, the knowledge fragment that defines the geometric dependency of hole1 (see Fig. 2.) is illustrated in Fig. 5 using the SNIL notation.

Our current AVANGO-based target system supports a scene graph metaphor with a field route concept and allows implementation of new proprietary node types. This justifies grouping of sub-parts as children of the main part. But this maps the modulated 9DOF_relative_to relations to fixed sg_part_of relations. To overcome the fixed scene graph semantics, a new node type called Constraint Mediator (CM) was developed which is parameterized by a dependency matrix for constraint definition. Instead of Geometric Constraint Solvers as in (Fernando *et al.*, 2001), which solve the constraints in an external system, the CMs in our target system implement the defined geometric dependencies as constraint nodes in the scene graph and apply them directly to the 4x4 transformation matrices. In other possible target systems, like VRML97, these constraints could be realized by using Script-Nodes, whereas the implementation of the scripting interface in VRML is often very slow (Diehl & Keller, 2002).

The CM nodes can monitor several fields to restrict and propagate their values. Unlike normal field connections (e,g, the field routes in VRML), CMs can propagate field-values in both directions and can alter the values while propagating, to establish complex constraints directly in the scene graph. These constraints implement the geometric dependencies of the knowledge based simulation as defined by the knowledge layer in the target system.

```
<semantic-net>
 <node name="2_hole_rod" type="Default"/>
 <node name="9DOF_2_hole_rod" type="Default">
   <slot name="FOR" type="9DimVect"
         inheritanceType="Attribute"
         value="0 0 0 0 0 0 1 1 1"/>
 </node>
     ...
 <node name="hole1" type="Default"/>
 <relation typeName="is_sub_part">
   <start-node nodeName="hole1"/>
   <end-node nodeName="2_hole_rod"/>
 </relation>
 <node name="9DOF_hole1" type="Default">
   <slot name="FOR" type="9DimVect"
         inheritanceType="Attribute"
         value="-.2 0 0 0 0 0 0 1 1 1"/>
 </node>
 <relation typeName="has_FOR">
   <start-node nodeName="hole1"/>
   <end-node nodeName="9DOF_hole1"/>
 </relation>
 <node name="9DOF_dep_mat_hole1" type="Default">
   <slot name="FOR" type="81DimMat"
         inheritanceType="Attribute"
         value="  … (9x6 identity)[1]
            0 0 0 0 0 0 0 0 0
            0 0 0 0 0 0 0 0 0
            0 0 0 0 0 0 0 0 1"/>
 </node>
 <relation typeName="9DOF_relative_to">
   <start-node nodeName="9DOF_dep_mat_hole1"/>
   <end-node nodeName="9DOF_2_hole_rod"/>
 </relation>
     ...
</semantic-net>
```

Fig. 5. SNIL fragment that partly describes the two-holed rod example. The extended dependency matrix that defines hole1's dependent scaling is defined as an attribute of the 9DOF_dep_mat_hole1 concept.

4.2.1 Inter-Part Constraints (Scaling / Transformations)

One application area for the dependency matrices is the simulation of building parts and sub-parts, which can have complex scaling behavior, depending on the scaling of their parent parts. Fig. 6 shows an example of a scene graph section with CMs which prevent the deformation of the holes when scaling the rod. When the user scales the rod in the X- or Y-direction, the two upper CMs in Fig. 6 set the adjust-scaling of the

[1] Only the lower three matrix rows which are relevant for scaling dependencies are depicted for readability.

holes to the inverse of the scaling of these directions to maintain the size and roundness of the holes. When scaling in the Z-direction—which is the direction of the main axis of the first hole—this hole is scaled with their parent, to fit with the thickness of the rod. The other two CMs restrict the user scaling of the holes to be equal in X- and Y-direction and to the identity scaling for the Z-direction. This allows the user to adjust the diameter of the holes with respect to the defined scaling behavior of the part.

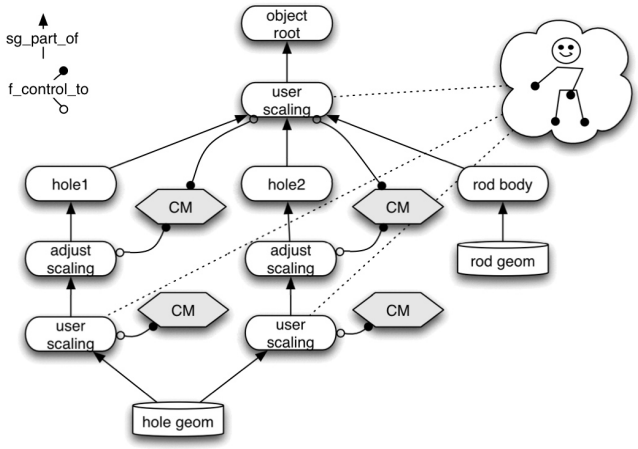

Fig. 6. Scene graph section with embedded Constraint Mediators for sub-part to part dependent scaling. CMs implement dependency matrix semantics for scene graph systems.

The coupling of other transformations as inter-part contraints is also possible. The parameters of parametrical changeable parts which are described in the KRL can be linked. This concept of linked transformations allows the simulation of gears in the virtual environment (Biermann & Wachsmuth, 2004). These gears are realized using the Constraint Mediators for the coupling of the transformation parameters. Simple rotational gears can be generated by using a coupling of the two rotations of the corresponding sub-parts with a certain transmission factor. E.g., a coupling of rotational and translational parameters can lead to a pinion gear.

4.2.2 Part-Part Constraints (Connections)

While the constraints for the scaling behaviour are normally fixed for each part, the simulation of part-part connections requires dynamic constraints. The coupling via constraints also allows the simulation of mating properties which can not be directly derived from the geometry of the parts. For example, it is possible to have plane-port connections, which restrict the movements of the parts so that the two connected planes always keep connected and in the same orientation, while they can slide until they reach the edges of the planes.

The mating geometries (so called Ports) define different degrees of freedom for the established connections. The knowledge base contains the information of the constraints for each Port-type. The restricted movements of the connected parts are

also controlled by Constraint Mediators, which are established via the semantic net, when a new connection is established. Technically, a connection is implemented by oriented mating points whose possible movements are restricted by CMs configured according to the Port types' constraints.

The example in Fig. 7 illustrates how the connection between the screw and one of the rod's holes is reflected in the target system's scene graph: A CM establishes the constraints, which simulate the connection of a screw fitted in a hole of a rod. The CM for the connection watches the positions of the two connected extrusion Ports and—in this case—alters the matrixes of the root nodes of the parts, if the positions of the Ports do not respect the constraints that are defined for this type of connection.

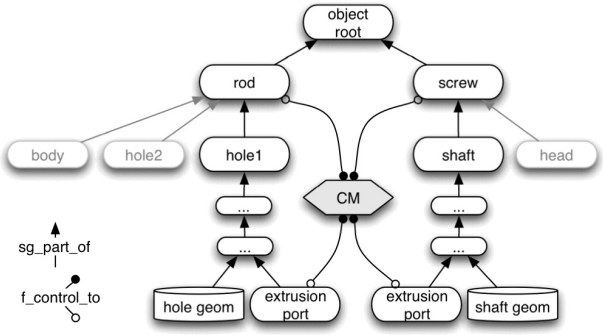

Fig. 7. Scene graph section for two parts interconnected by a constraint mediator to implement part-part geometric dependencies

5 Conclusion

We have introduced a general method for integrating semantic information into the VR simulation and interaction loop. It is based on an abstract knowledge representation layer (KRL) for high-level definition of complex application designs. The FESN, the KRL's base formalism, provides a convenient method for AI related application solutions. It interconnects the data structures of the required simulation modules and provides external representation formats to express simulation data as well as application logic in a human readable way and hence supports reusability and extensibility of once developed representations.

Semantic Entities as unified object models provide the necessary method to uniformly access the KRL during runtime. Using Semantic Entities as the central entity access facility provides several advantages: Simulation modules can be built which automatically match their functions to the respective target objects. Possible object actions and interactions can be specified in advance, using a declarative notation. Developed simulation components can be reused, adapted and modified easily. Even complex application developments can be performed by generating a few lines of XML code for the required knowledge structure.

The usefulness of a high-level knowledge representation has been demonstrated (1) for a generalized scene representation which extends the expressiveness of commonly used scene graphs and (2) for the implementation of novel multimodal interaction

methods. The illustrated method is currently applied in several projects in our lab which focus on multimodal human-computer interaction and virtual construction applications.

Future work expands the KRL to support a variety of simulation components from different graphics packages to physics libraries. The final goal is a platform which conceptually allows abstract definition of intelligent VR applications via the KRL with as minimal adaptations from the utilized simulation systems core functionality. This work has already begun with the development of an automatic data synchronization and replication framework required.

Acknowledgement

This work is partially supported by the Deutsche Forschungsgemeinschaft (DFG).

References

Biermann, P., & Jung, B. (2004). Variant design in immersive virtual reality: A markup language for scalable CSG parts, *AMDO2004*: Springer.

Biermann, P., & Wachsmuth, I. (2004). Non-physical simulation of gears and modifiable connections in virtual reality, *Proceedings Fifth Virtual Reality International Conference (VRIC 2003)* (pp. 159-164). Laval, France.

Carey, R., Bell, G., & Marrin, C. (1997). *Iso/iec 14772-1:1997 virtual reality modeling language (vrml)*: The VRML Consortium Incorporated.

Cavazza, M., & Palmer, I. (2000). High-level interpretation in dynamic virtual environments. *Applied Artificial Intelligence, 14*(1), 125-144.

Diehl, S., & Keller, J. (2002). Constraints for 3D graphics on the internet, *Proceedings of 5th International Conference on Computer Graphics and Artificial Intelligence 3IA'2002*. Limoges, France.

Fernando, T., Marcelino, L., & Wimalaratne, P. (2001). Constraint-based immersive virtual environment for supporting assembly and maintenance task, *Human Computer Interaction International 2001*. New Orleans, USA.

Kopp, S., Jung, B., Lessmann, N., & Wachsmuth, I. (2003). Max—a multimodal assistant in virtual reality construction. *KI-Künstliche Intelligenz, 03*(4), 11-17.

Latoschik, M. E. (2001). A gesture processing framework for multimodal interaction in virtual reality. In A. Chalmers & V. Lalioti (Eds.), *AFRIGRAPH 2001, 1st International Conference on Computer Graphics, Virtual Reality and Visualisation in Africa* (pp. 95-100): ACM Press.

Latoschik, M. E. (2002). Designing transition networks for multimodal VR-interactions using a markup language, *Fourth IEEE International Conference on Multimodal Interfaces ICMI'02* (pp. 411–416). Pittsburgh, Pennsylvania: IEEE Press.

Latoschik, M. E., & Schilling, M. (2003). Incorporating VR databases into AI knowledge representations: A framework for intelligent graphics applications, *Sixth IASTED International Conference on Computer Graphics and Imaging* (pp. 79-84): ACTA Press.

Luck, M., & Aylett, R. (2000). Applying artificial intelligence to virtual reality: Intelligent virtual environments. *Applied Artificial Intelligence, 14*(1), 3-32.

Peters, S., & Shrobe, H. (2003). Using semantic networks for knowledge representation in an intelligent environment, *PerCom'03: 1st Annual IEEE International Conference on Pervasive Computing and Communications*. Ft. Worth, TX, USA: IEEE.

Pfeiffer, T., & Latoschik, M. E. (2004). Resolving object references in multimodal dialogues for immersive virtual environments, *IEEE VR2004* (pp. 35-42). Chicago: IEEE.

Soto, M., & Allongue, S. (2002). Modeling methods for reusable and interoperable virtual entities in multimedia virtual worlds. *Multimedia Tools Appl., 16*(1-2), 161-177.

Tramberend, H. (1999). Avocado: A distributed virtual reality framework, *1999 IEEE Virtual Reality Conference (VR99)* (pp. 14-21). Houston, Texas: IEEE.