Automatic Data Exchange and Synchronization for Knowledge-Based Intelligent Virtual Environments

Guido Heumer [1]
AI & VR Lab,
Faculty of Technology,
University of Bielefeld
P.O. Box 10 01 31
33501 Bielefeld, Germany

Malte Schilling [2]
AI & VR Lab,
Faculty of Technology,
University of Bielefeld
P.O. Box 10 01 31
33501 Bielefeld, Germany

Marc Erich Latoschik [3]
AI & VR Lab,
Faculty of Technology,
University of Bielefeld
P.O. Box 10 01 31
33501 Bielefeld, Germany

Abstract

Advanced VR simulation systems are composed of several components with independent and heterogeneously structured databases. To guarantee a closed and consistent world simulation, flexible and robust data exchange between these components has to be realized. This multiple database problem is well known in many distributed application domains, but it is central for VR setups composed of diverse simulation components. Particularly complicated is the exchange between object-centered and graph-based representation formats, where entity attributes may be distributed over the graph structure. This article presents an abstract declarative attribute representation concept, which handles different representation formats uniformly and enables automatic data exchange and synchronization between them. This mechanism is tailored to support the integration of a central knowledge component, which provides a uniform representation of the accumulated knowledge of the several simulation components involved. This component handles the incoming–possibly conflicting–world changes propagated by the diverse components. It becomes the central instance for process flow synchronization of several autonomous evaluation loops.

CR Categories: D.2.11 [Software Engineering]: Software Architectures—Data Abstraction; H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, Augmented, and Virtual Realities; I.2.4 [Artificial Intelligence]: Knowledge Representation Formalisms and Methods—Semantic Networks

Keywords: Knowledge Representation, Intelligent Virtual Environment, Semantic Network, Data Synchronization, Mediator

[1] gheumer@techfak.uni-bielefeld.de
[2] mschilli@techfak.uni-bielefeld.de
[3] marcl@techfak.uni-bielefeld.de

1 Introduction

The design and creation of believable immersive Virtual Reality (VR) applications and environments is challenging. The diverse simulation tasks involve the methodical and technical integration of several different simulation processes. A multitude of projects nowadays support application design with software tools targeted at VR development: DIVE [1], NPSNET [2], MASSIVE-3 [3], VR Juggler [4], GNU/MAVERIK [5], DEVA3 [6], Lightning [7], AVANGO [8] or commercial game engines–only to name a few–incorporate a multitude of concepts, methods and solutions for the diverse tasks of a VR simulation system.

Nevertheless, at a certain point even the most elaborate tool will lack a required feature; hence most of the VR development tools provide methods for proprietary extensions, which are widely utilized for advanced applications. Such applications are often based on heterogeneous software setups. The resulting complexity involves application-tailored data exchange, data replication and process flow synchronization methods between components. This significantly complicates system maintenance as well as reusability and exchangeability of once developed and integrated components. For this reason, in contrast to supporting as many as possible features out of the box, component-based approaches explicitly modularize the involved tasks by defining clear interfaces between the modules to be integrated [9, 10].

Work at our lab focuses at the application of Artificial Intelligence (AI) techniques to Virtual Environments (VEs). Projects thematically range from Virtual Construction to Human-Computer-Interaction (HCI), e.g. including work on multimodal interaction and on communication setups with our artificial humanoid communication counterpart called MAX. These approaches require purpose-built solutions for the technical integration of AI methods into VR and real-time graphics setups. For example, we have used several graphics packages from OpenGL to Open Inventor and SGI's Performer library as well as a variety of solutions for collision detection and physics simulation. Nowadays, we mainly utilize AVANGO [8] as the high level VR framework.

Regarding the AI side, these tools had and have to be integrated with an additional diversity of concepts, methods and associated software tools, e.g. rule-based systems, structural representations (frames, semantic networks), distributed agent based as well as connectionistic approaches (neural networks).

AI and knowledge based principles can significantly enrich VR applications. At a certain point, our VEs have to be intelligent to be believable since simulating the reality will always be an approximation and simulation accuracy is costly with respect to the deployed computing resources. Here, techniques from Artificial Intelligence and knowledge based systems have frequently proven to be useful as they support e.g. the heuristic approximation of physical simulations [11]. Furthermore, novel interaction methods, like multimodal interfaces [12-14], require a semantic representation of the virtual scene. To understand multimodal utterances and the user's intention, the system must have access to various types of knowledge, including background, general and specific task knowledge.

The required knowledge and the simulation system's internal representations are tightly correlated. For example, RGBA values of an entity correspond to a color, which under the given lighting parameters would be described in communication as "*green*"; a given 4x4 transformation that defines position and orientation of an entity would be interpreted from a humanoid's frame of reference as being e.g. *on the left side* for a given time span. The AI and VR representations of a given scene are more precisely described as different views of the same domain–the virtual scene. This motivates a common representation layer, which includes knowledge and data required by all deployed simulation modules. Such a knowledge representation layer (KRL) then becomes the central knowledge- and database and would inherently support the design of Intelligent Virtual Environments (IVEs) [15] that include semantic information [16, 17] as well as simulation specific data in a central knowledge base.

2 VR SYSTEM DESIGN

This article describes the foundation of our approach for an IVE platform. Its overall architectural design is illustrated in figure 1 and will be described in detail in subsection 2.2. This platform provides application designers with interconnection methods for required simulation modules, which are arranged around a central KRL. For compatibility and performance reasons, the diverse modules are not forced to work on a central data representation. In contrast, they are synchronized with the KRL and with each other in a predefined and determined way. Hence, as critical features such a platform has to provide data exchange and synchronization facilities.

As mentioned before, creating a Virtual Reality comprises diverse simulation tasks. If they do not support these tasks out-of-the-box, most VR frameworks at least provide several extension mechanisms. For example, the VR framework AVANGO is built on top of OpenGL Performer for rendering purposes and adopts the scene graph as a central database. The Performer scene graph is extended by a field concept that provides a dataflow network of field connections similar to VRML.

AVANGO provides built-in concepts for display abstraction, 3D user interaction and network distribution. Additional functionality (e.g. a dynamics simulation), can be realized via AVANGO's extension concept, e.g. by utilizing the scripting facility or by creating new node types, which provide the required functionality. These extension nodes are integrated into the scene graph. However, as illustrated in [18], the scene graph as a data structure has been conceived for graphics display purposes and often is inappropriate for certain other simulation tasks.

2.1 Component-based Approach

Another promising and reasonable architecture for a VR-system is the interconnection of stand-alone components for the diverse simulation tasks. Here "stand-alone" denotes the components' usability and functionality outside of the context of the whole system. Prominent simulation tasks include, e.g. graphics, audio and haptic rendering, user interaction, simulation of physical properties, distribution support as well as advanced tasks, like e.g. AI reasoning. The most significant advantages of such a component-based approach are:

1. **Reusability:** Components that are functional on their own can be used in multiple differently structured overall systems. This reduces effort for development. Results of completed projects can be integrated in new projects without the need for redesigns.
2. **Exchangeability:** In case one component does not fulfill its function in a satisfactory way or a more appropriate component becomes available, a component can be exchanged with relatively little adaptation effort.
3. **Modularity:** The functionality of the overall system can be extended in a flexible way by adding new components. On the other hand, a component can be omitted, should its functionality not be necessary. For example, an application that only presents a simple walkthrough in most cases does not need a full dynamics simulation etc. By omitting unnecessary components resources are saved, which benefits the other components.

However, this component-based approach also implies one

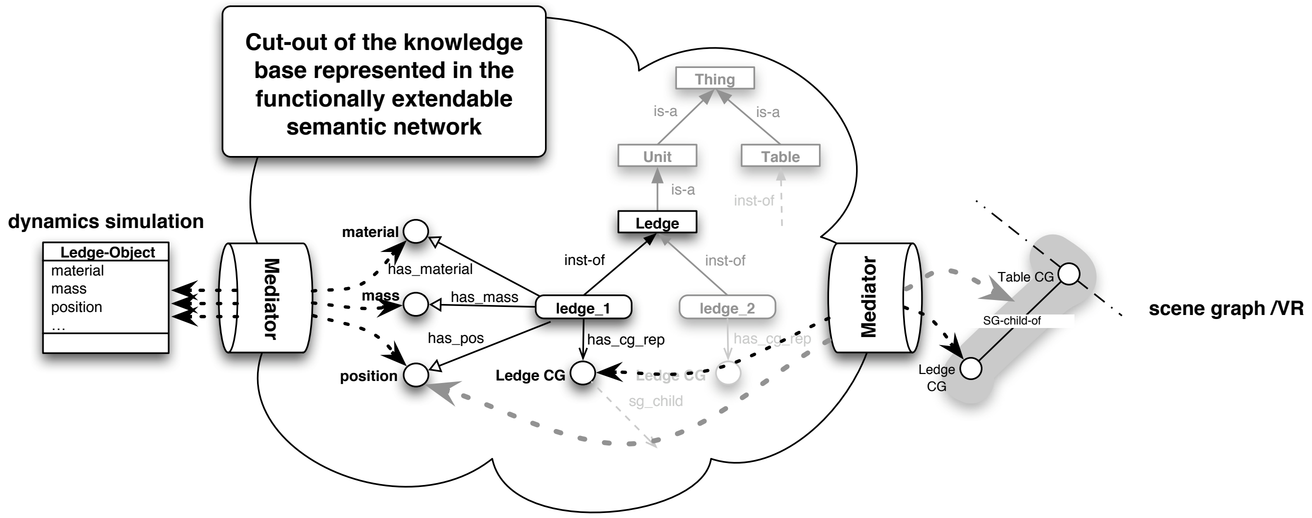

Figure 1 – Represented knowledge and interconnected domains in the FESN. Identical attributes are shared between components as illustrated with the position concept in the lower left area. Mediators control the mapping between differently structured attributes connoting the same concept or thing.

significant problem. Each of the stand-alone components has its own specialized world representation, which is customized and restricted to the particular functional aspect of the virtual environment simulated by the component (e.g. graphics, sound, dynamics etc.). The structure of this representation is optimized for the specific demands and peculiarities of the component's functional domain: For example, a hierarchical scene graph has proven to be useful for computer graphics whereas object-centered formats based on world coordinates are often favored in dynamics simulations (the different formats are illustrated in more detail in section 4.1).

Some of the data in the different components is only relevant for the specific simulation domain of the respective component such as sampled audio files for the audio output component, mass and density properties for the dynamics component etc. Other data however is relevant for several of the participating components. For example, the spatial position and orientation of an object is relevant for the graphical display as well as for dynamics simulation, for audio output and more.

Thus, the knowledge about world entities is partly kept redundantly in the different world representations of the particular components. In a dynamic world where objects undergo changes over time, this redundant data has to be constantly aligned (*synchronized*) for the jointly simulated world to be consistent. The big challenge in this database synchronization lies in the structural heterogeneity of the particular world representations, which are mostly incompatible. Our proposed approach for a unified handling of heterogeneous representation formats will be discussed in section 4.

2.2 Overall Architecture

Another critical implication of the component-based approach for a VR system is the loss of a central controlling instance or rather the possible existence of several autonomous simulation processes. Since these processes do not have knowledge about each other, conflicting world changes are prone to occur. For example, the dynamics component might move an object downward due to the force of gravity while the user interaction component reports the object to be moved upward, because the interacting user drags it. Therefore a central data representation instance is required, which resolves conflicts in world changes as generated by the participating components.

Here, we propose a central knowledge representation layer (KRL) that handles conflicting world modifications and guarantees consistency of the overall simulation. Additionally it provides a base formalism for AI content.

This base formalism has to be powerful enough to subsume the formats of all the connected simulation components. Graph representations have shown to be appropriate for modeling knowledge. They are easy to use and the representation of relations is straightforward as part of the formalism. Furthermore, commonly used representation formalisms of VEs, e.g. scene graph structures or objects with attributes, are seamlessly integrated in such graph representations. Hence, we have developed a graph-based representation as the base formalism for the KRL: A functionally extendable semantic network (FESN).

In addition to its purpose as a central facility to resolve conflicting world changes, the FESN base formalism is characterized by the following features:

1. It provides a central representation for inferences. Reasoning processes can access the full context represented in a virtual scene. This includes background knowledge provided by connected simulation components.
2. It provides one central place for the definition and design of a VE. Designers do not have to separately create, e.g. graphics content, a dynamics database, interaction rules etc. but these can be directly modelled using the FESN.
3. It provides a basis for high-level definitions of interaction patterns. This includes abstract specifications of desired direct manipulation metaphors as well as of advanced multimodal interfaces.

Technically, the component-based architecture requires a mediator layer between the particular simulation components and their respective data representation formats. From the mediator layer's point of view, the FESN is a component like any other, whose data has to be synchronized with the other components. However, the connection pattern in our approach sets the FESN as the central point of a star-like topology, so all other components are connected with it in a point-to-point manner. This enables the FESN for its role to supervise overall attribute value changes and to resolve conflicting events.

Figure 1 illustrates the overall architecture of the resulting VR platform. In a first implementation we are using the AVANGO VR framework and the commercial Vortex dynamics engine as simulation components. These are connected through the mediator layer to the central KRL, which will be explained in detail in the following section.

3 Functionally Extendable Semantic Net

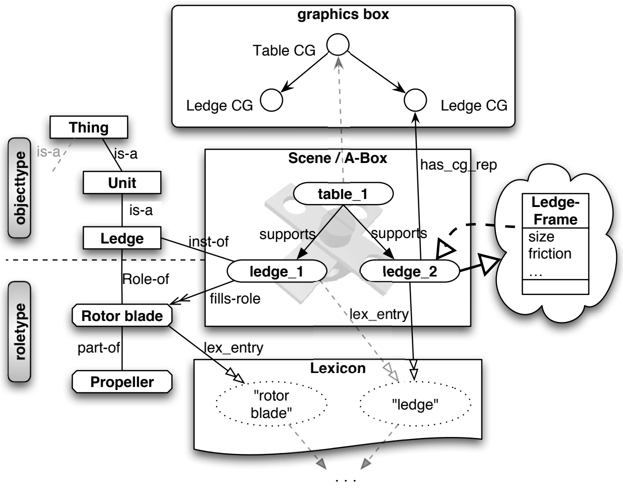

Figure 2 – The several different knowledge areas that comprise the knowledge in an IVE. The concrete instances in the current scene or A-Box (since it represents assertional knowledge) are connected to concepts of different domains (graphics, physics, lexicon etc.).

Information in the FESN is stored—as typically known from semantic nets—in nodes and relations (the base constructs). Nodes denote individuals in case of instances or concepts in case of abstract knowledge. They are explained through their relations and the related nodes. The background knowledge distinguishes several parts specific to the simulation modules. The concept knowledge also disperses into different parts corresponding to the different domains. Instead of solely distinguishing knowledge between assertional box (A-Box) and terminological box (T-Box) we further subdivide the T-Box into a physics box, a graphics box, an audio box, etc (see figure 2) [19]. Their data and knowledge representations are connected to the present assertions, where simulation specific components add assertions strongly bound to their domain and the corresponding semantics.

In the integration of the different knowledge domains—considering VR as the field of application regarding semantics—the perceivable VE is represented as a whole inside the FESN. Consequently, the FESN holds knowledge about all visible objects, their possible relations and qualities as well as (inter-) actions suitable in the simulated situation.

Since the FESN is acting as the central knowledge base for an IVE, it has to be real-time capable. Our implementation fulfills this requirement by storing information locally inside the nodes and relations as well as globally—indexed in a global lookup table. Therefore requests can be answered efficiently in two ways—querying a fact directly at a node or traversing the graph—which both have proven to work efficiently.

The FESN has its origin in classic semantic nets, but its utilization for IVEs requires the extension of the semantic network formalism. The main aspects to this extension, which will be explained in detail in the following subsections, are:

1. Frame-like node structure
2. Functional extensibility of the base constructs
3. Event system
4. Open and persistent exchange format

3.1 Frame-like Node Structure

Simulation components that have object-centered representation formats represent simulated entities as objects containing a set of attributes and values defining these entities' properties. Representing such facts in a graph can be awkward and confusing. For this reason, the FESN offers an object-centered view of objects and concepts. Corresponding to the AI concept of frames, nodes can have attributes or *slots*, so they contain sets of slot-name-value-pairs. The values can be of any type, including complex object structures, which are then wrapped inside the slot. Slots are also augmented with information about value inheritance to instances. The instantiation relation has to be functionally extended (see below) to consider these markings and to process the inheritance of values accordingly.

3.2 Functional Extensibility of the Base Constructs

Some of the defined concepts have an extended meaning in a specific application context. This direct semantics binds the meaning of a node or relation to the specific functionality in a simulation component.

For example, a "supports" relation may denote that an object is lying on another object. If the supporting object moves, the supported object should move accordingly. Regarding a scene graph based visualization component, this semantics would normally be implemented by a hierarchical scene graph grouping. To reflect this, the supports relation in the FESN can be extended by a function, which—in case of an instantiation of a supports relation between two objects—automatically generates the required scene graph structures inside the visualization component. Nodes and relations in the FESN can be functionally extended to fulfill the extended meaning or function in the corresponding application. On the one hand, these base constructs have their semantics given through the semantic net by their relations or related constructs. On the other hand, they are bound to the specific functionality in a simulation component, which is defined in a procedural way.

Other examples are the instantiation and subsumption relations which, among other things, define the inheritance of certain attributes and structures (relations to other nodes) along these relations. Does an attribute of a certain concept mean that for a concept's instance such an attribute exists—or does it mean that all instances inherit the value itself? The inheritance of attributes and structures has to be regulated and the special meaning of these relations has to be addressed. Again, the described extension mechanism is utilized to define additional functions for the instantiation and subsumption relations. These functions now implement how the regulated inheritance is applied to the involved concepts and individuals.

3.3 Event System

Different domains and applications work on the FESN concurrently. Modifications of structures or attribute values inside the central knowledge representation have to be supervised. The FESN offers a synchronized event system, allowing the propagation of modifications automatically. It collects all modifications from the different simulation components (for a detailed explanation of the synchronization process see section 5). Receiving more than one event for an attribute may cause a conflict inside the base construct. The base construct itself must decide which event to accept or how to integrate the collected events. Therefore, the concept of *event filters* has been introduced. An event filter encapsulates a heuristic how to deal with conflicting or concurrent events, collected during a simulation cycle in a base construct of the FESN. The event filter can just choose one of the events (e.g. the most recent one), can have several preferences (e.g. events from a specific simulation domain should be preferred), or can implement any other heuristic (e.g. interpolate a new result event out of all the events).

3.4 Open and Persistent Exchange Format

Graph representations generally benefit from their easy and straightforward way of representation. The meaning of objects and concepts is expressed by describing their relations and interconnections. Thus, the FESN is accessible in a suitable and human understandable way. Existing knowledge can easily be maintained and new knowledge can easily be added. For this purpose, the structure of the semantic net and attribute values are expressed in an XML-Format similar to ontology description formats as OIL or OWL—making it possible to exchange knowledge and to connect to common knowledge expressed in an ontology description.

4 Automatic Data Exchange

As has been illustrated, the overall goal is the design of a VR platform, which composes diverse stand-alone components and which aligns their respective databases through a central data exchange and synchronization mechanism. The need arises for a mediator layer between the components that automatically keeps the redundant data in the databases of the participating components synchronized. To achieve this, the mediator layer needs a formalism to represent all the different representation formats in an abstract, uniform and declarative way and to provide a means to convert attribute information from one format to another. Furthermore, the mediator layer has to be extendable to support the integration of new components or the replacement of old ones without a complete redesign of the overall application or the mediator layer. As mentioned before the main challenge lies in the structural heterogeneity of the different representation formats. In this section we introduce a concept for an abstract formalism that integrates all these formats in a uniform way and thus enables automated data exchange between them.

4.1 Structural Heterogeneity

Essentially two large groups of representation formats can be distinguished. One group comprises the *object-centered formats* and the other the *graph-based* or *structural formats*. In the first group the world is represented by a loose collection of entities or objects and all data relevant to an object is kept locally in its attributes. This kind of representation format is often found in dynamics simulation engines or computer game engines.

In graph-based representation formats the objects or world entities are arranged in an integrative overall graph structure. This group can be subdivided further, according to the type of graph-structure used. Classic scene graph structures use a hierarchical acyclic graph with a defined root node whereas more general graph structures like semantic networks may contain cycles and have no defined root or starting point. Object properties in this group of formats are also defined by attributes of the single nodes. The crucial difference to the object-centered formats however, is the possible distribution of attribute information. Attribute information does not have to be only locally defined but can be spread over parts of the graph structure. A classic example for this is the global transformation information of geometry in a scene graph structure, which usually is not only specified by a single transformation matrix in one node, but by the combination of several matrices along a certain path through the scene graph.

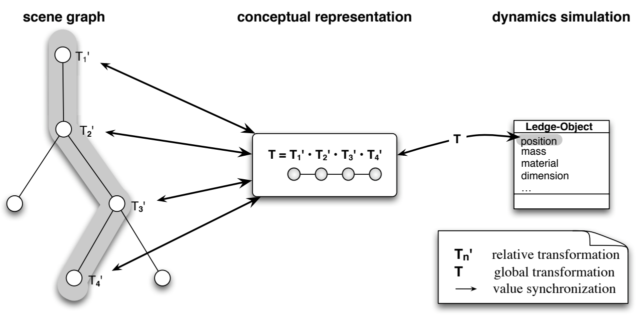

Figure 3 – Attribute value exchange between a scene graph based and an object-centered representation format. A conceptual representation is needed to define how the distributed transformation information in the SG is combined to the global transformation.

To be able to exchange such *distributed attribute information* with a representation format where this information is kept locally (or even a differently structured representation), a conceptual representation is needed (see figure 3) where the contributing attributes and rules for their combination to a total value are declared.

4.2 Composite Attributes

To unify all the representation formats mentioned above under one formalism, we propose the concept of a *composite attribute* (or short CA). Generally, a CA is a data structure that defines how several *partial attributes* are combined to calculate a total attribute value. Structurally a CA consists of a sequence of (container-) objects that are connected pair wise by relations. In each of the objects one attribute is marked respectively for being relevant for total attribute value calculation. The resulting structure is a linear chain of relations, which connects objects that contain attributes (see figure 4).

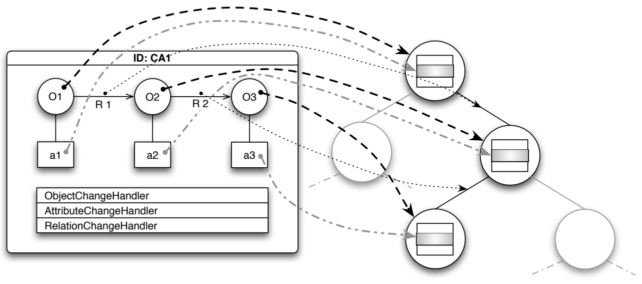

Figure 4 – Example for a CA structure corresponding to a path through a scene graph. Objects, attributes and relations are represented and have references to their counterparts in the original SG. Event handlers record changes.

The constituents of the CA structure (objects, attributes and relations) contain references to the concrete implementation instances of the respective simulation component. In the transformation example each of the objects in the CA (O1, O2 and O3) correspond to nodes in the scene graph, the attributes (a1, a2 and a3) are the transformation-fields in these nodes containing transformation matrices and the relations between the objects (R1 and R2) refer to the parent-child relation between the nodes. Thus, a CA structure essentially corresponds to a path through the structural graph of the simulation component's database.

4.2.1 Evaluation Algorithm

To calculate the total value of a composite attribute the relation chain of the CA structure is iterated. A certain *arithmetic* (or calculation rule) is determined per relation, which is defined by the types of the structural components involved - namely the connecting relation, the two connected objects and their marked attributes (see figure 5). Technically, the arithmetics are kept in a central arithmetics-table and are addressed by a key, which is generated from the five types. An arithmetic combines two attribute values of a given type to a new composite value. For the first relation the values of the attributes in the first two objects (*w(a1)* and *w(a2)* in figure 5) are used as parameters and an intermediate value is calculated. For all following relations the current intermediate value is used as the first parameter, the value of the second object's attribute is used as the second parameter, and a new intermediate value is calculated. This procedure is repeated for each relation in the chain, and finally the result of the last calculation step delivers the total value of the composite attribute: $w(CA1)$

Generally, the evaluation process is a linear chaining of freely definable calculations, which corresponds to a traversal of the partial graph described by the CA structure and the accumulation of a state value from the partial attributes.

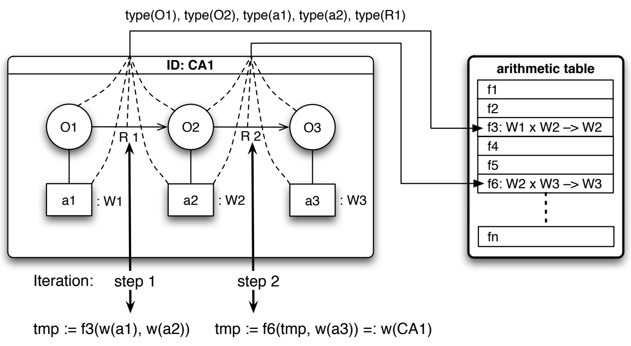

Figure 5 – Evaluation of the CA structure to its total value. Iteration over the relation chain is performed, determining an arithmetic for each relation. This arithmetic determines how the partial attribute values are combined to a total value.

It has to be regarded that the value types of the arithmetics' parameters have to be compatible to the value types of the attributes (W1 to W3 in figure 5). In the global transformation example, the corresponding arithmetic would be the matrix multiplication combining the relative transformation matrices in the fields of the scene graph nodes to the global transformation matrix.

Whenever one of the participating attribute values changes, the composite attribute's total value has to be recalculated by the mediator layer. This value can then be synchronized with the other participating simulation components (for details on this see section 5).

It is important to keep in mind that a composite attribute changes its value not only if one or more of its partial attributes change their values. The total value might also change if the constituting structure as such is changing (e.g. parts of the scene graph are regrouped). To ensure robustness against this type of value changes the CA representation in the mediator layer has to be kept informed about all structural changes in the underlying native structure as well and has to be modified accordingly.

A CA structure representing an attribute in an object-centered representation format is a special case, because in this case an attribute is not composed of several partial attributes but completely locally defined. In this case the CA structure consists of only one object with a marked attribute. The mediator layer now only has to detect value changes in this attribute and inform the other connected components about this, while converting the values between the different formats.

4.2.2 Assignment of Values

A more complicated case is the *reverse direction*, i.e. the insertion of a new total value into the CA structure (from right to left in figure 3). The problem here is to determine which of the partial attributes have to change their values so that the total value of the CA structure yields the desired new value. In most cases, where two or more partial attributes are involved, there would be an infinite number of possibilities to distribute the values that would result in the same total value. Hence, there is no single defined solution to calculate. In our approach, a certain *designated attribute* is defined during setup of the CA structure. This particular attribute and only this one is the attribute that value changes are applied to while all other partial attributes in the CA are left unchanged.

Determining the target value for the designated attribute requires two inverse functions for a given arithmetic: the *left-inverse*, which calculates the first parameter given the result and the second parameter of the original arithmetic and the *right-inverse*, which yields the second parameter given the result and the first parameter. Hence, if the arithmetic evaluates to $f(w_1, w_2) = w_R$ the left-inverse delivers $f^{-L}(w_R, w_2) = w_1$ and the right-inverse $f^{-R}(w_R, w_1) = w_2$. These inverses are also kept in the arithmetics-table and are addressed by object types, attribute types and relation type, analogous to the arithmetic itself.

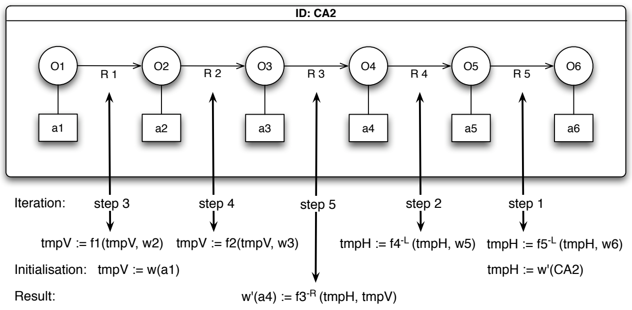

Figure 6 – Reverse direction of the evaluation algorithm of a CA structure. The attribute a4 is the designated attribute, whose new value has to be calculated. Two iterations are performed, accumulating the temporary values tmpH and tmpV. These are combined to the desired value of the designated attribute in the last step. The value of a2 w(a2) is abbreviated with w2 etc.

Figure 6 illustrates the algorithm for the reverse direction. Generally, two iterations are performed. The first iteration works backwards over the relation chain, accumulating a temporary value `tmpH` using the left-inverse. `tmpH` is initialized with the new desired value for the CA: $w'(CA2)$. This iteration terminates when the designated attribute a_{des} (a4 in this case) is reached, i.e. it is contained in the first object of the currently processed relation. The second iteration works from the beginning of the relation chain forward, using the original arithmetic, and accumulating a second temporary value `tmpV`, which is initialized with the value of the first attribute $w(a_1)$. When the second iteration reaches the designated attribute (i.e. it is contained in the second object of the currently iterated relation), the desired value of the designated attribute is calculated from the temporary values by using the right-inverse: $w'(a_{des}) = f^{-R}(tmpH, tmpV)$

4.2.3 Conclusion

With the formalism introduced here, arbitrary combinations of attribute values along a linear path through a graph can be described. Total values of such composite attributes can be calculated from the structure and new values assigned to the structure. This permits the exchange of graph-based representation formats with object-centered formats. By allowing the free definition of arithmetics in an arithmetics table, which is addressed by the types of the involved objects, extensibility towards arbitrary combinations of objects, attributes and relations is granted.

4.3 Extendable Mediator Layer

For our VR simulation platform the described data exchange formalism has been implemented as an extendable mediator layer. To facilitate the integration of future components into the system, a design was chosen that conceives the separation into a main mediator component and component-specific wrapper layers (see figure 7).

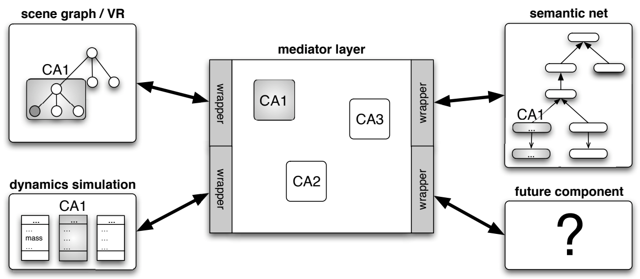

Figure 7 – Software engineering view on the mediator framework and connected components. Components are connected through specific wrapper layers, which implement a common interface, ensuring extendibility.

The main component implements the structural exchange mechanism, using the CA concept, and manages the connected wrappers. A wrapper layer realizes the part of the mediation process that is specific to its connected component, like the mapping between the CA structure and the component's implementation instances. The advantage of this wrapper-based approach is, that a wrapper layer has to be developed only once for a given component. From then on, it can exchange values with all simulation components that wrappers will be written for at a later point of time.

For each attribute that is to be synchronized between two or more simulation components, a CA structure is defined in the wrapper layers of each of the participating components. Each CA structure defines how the attribute value is calculated for this individual component. By assigning a global ID (e.g. CA1 in figure 7) for each composite attribute, the mediator layer knows which CA structures it has to synchronize in which wrappers. The wrapper layer registers the CA structures with the event system of the connected component to keep it informed about value as well as structural changes. In case no event system exists for a given component, the wrapper layer has to poll the component's database for value changes—however this only has to be performed for the relevant attributes.

Another function of the wrapper layers is the conversion of values between the formats of the simulation components. To ensure extensibility here, the wrapper has to provide methods to convert values from the native format of the simulation component to a general format the mediator layer defines and vice versa. Therefore, the mediator layer has an extensible type system to identify types of values from the different connected components uniquely.

5 Temporal Synchronization

After a solution for the structural aspect of synchronizing heterogeneous databases has been presented in the preceding section, now a concept for the temporal integration of the process flow in the overall system is introduced.

Generally, all simulation components are running in independently timed evaluation loops comprised of several update steps. For the integrated VR system, we propose to call the different update steps and loops from a central synchronization facility. This facility also triggers the propagation of value changes through the mediator layer.

It is important for data synchronization that the local database of a component is in a consistent state at the time of a global synchronization step. This normally is the case only between two update-cycles. However, the mediator layer has no knowledge about when such points in time are reached. For this reason, the wrapper layers define a synchronization method that is called by the synchronization facility when the simulation component has finished its update cycle. This call prompts the component to propagate its value changes via the mediator layer to all other components, which subscribed for the CAs in question. The mediator layer provides two mechanisms to handle the value changes in the receiving components. Either they can be collected in form of value change events in dedicated event loops for later processing or they can be applied directly to the local database. Which mechanism is used depends on the overall synchronization policy.

The temporal flow of the synchronization process in our integrated VR system is divided into three stages, which are repeated in a loop (see figure 8):

1. **Collect stage:** All value changes in the databases of connected components are determined and sent in form of value change events to dedicated event-queues in the FESN, where they are collected. To achieve this, an update cycle of the participating components is executed and the synchronization methods of the respective wrapper layers are called.
2. **Sync stage:** The collected events are processed in the FESN and an update step of the FESN is executed. *Event filters* that are attached to the nodes and relations in the FESN resolve conflicting events.
3. **Propagate stage:** Finally, the events adjusted by the event filters in the FESN are propagated to all connected components, where they are applied directly to the databases. For this purpose, the synchronization method of the FESN wrapper layer is called.

For this approach to be viable, the evaluation loops and update steps of the different components have to be controllable externally. In our case, this is given for the Vortex dynamics simulation and for the knowledge representation layer. The AVANGO VR framework originally does not allow external control of its main loop. However, due to the open-source status of the AVANGO project, we were able to integrate custom-made mechanisms, which made the integration possible. Future versions of our VR platform, however, may include components that do not allow external control or modification of their evaluation loops. In this case, several evaluation loops have to be run concurrently. For this reason, a more sophisticated synchronization mechanism that can handle real concurrency of the evaluation loops is subject of our current research. Such a scheduling system has to take the complex interrelations and interdependencies of the single components of an interactive VR system into account.

6 Conclusions and Future Work

This paper presented a method for automatic data synchronization of redundant data in interconnected autonomous simulation components. It proposed a concept of composite attributes, which allows the integration of heterogeneous representation formats in an abstract, uniform and declarative way. This concept facilitates the description of arbitrary calculation rules for attribute values along linear paths through

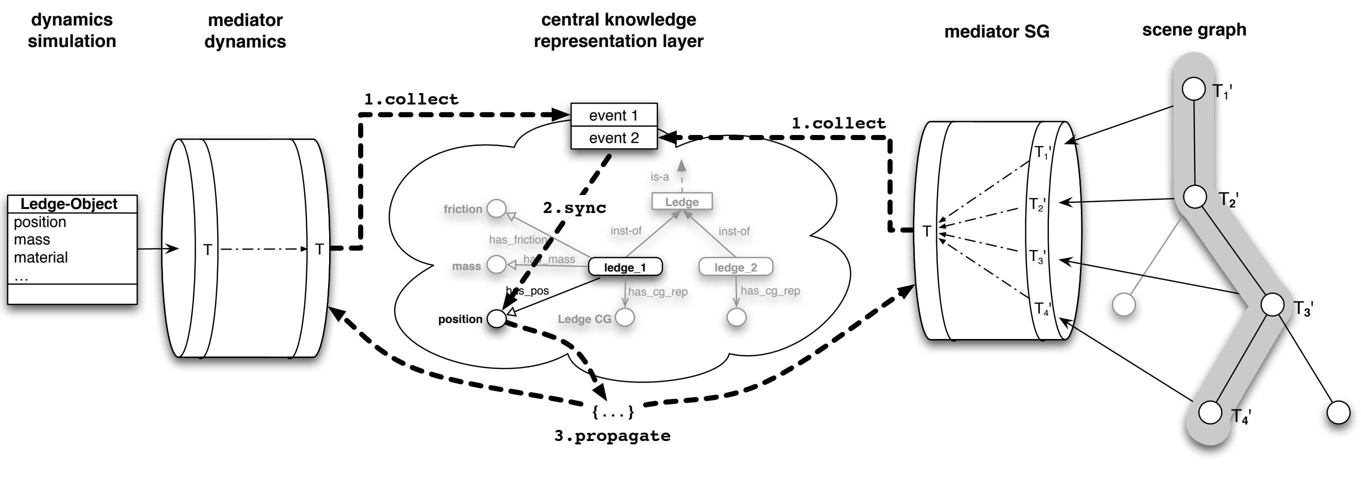

Figure 8 – The three synchronization stages of the automatic data exchange mechanism: 1. *Collect* – Value change events from the connected components are collected in the FESN. 2. *Sync* – Conflicting events are resolved through event filters. 3. *Propagate* – Resolved value changes are propagated to the connected simulation components.

graph structures (i.e. including scene graphs and semantic nets). Thus, it overcomes the structural heterogeneity of simulation components and enables data exchange between e.g. graph-based and object-centered representation formats.

The structural aspect of data synchronization is supplemented by a concept for the temporal synchronization of data flow in component-based simulation architectures. It introduces a central knowledge representation layer, which is implemented by a functionally extendable semantic net as the base formalism. This KRL integrates the various types of knowledge kept in the simulation components. It handles and resolves conflicting world changes propagated by these components through freely definable event filters.

The mediator layer and the FESN have been developed in [20] and [21] respectively. An initial prototype connects the AVANGO VR framework with the Vortex dynamics engine and realizes automatic data-exchange of object transformation information between them. The knowledge representation layer is already used to define intelligent simulation operations and advanced interaction methods in multimodal virtual construction scenarios.

The introduced synchronization framework and knowledge representation component are the foundation for a knowledge based VR platform. Its open and extendible architecture is now used to integrate additional components, e.g. to implement a stand-alone multimodal interaction component, which is currently closely attached to the AVANGO framework.

Future work will additionally provide alternative components for audio, graphics and physics simulation, e.g. by integrating OpenSG or Open Dynamics (ODE). Future work on the KRL includes research on declarative semantics for the base formalism as well as exploration and implementation of alternative AI representation methods, e.g. to incorporate neural networks as additional base formalisms. In its completion, the projected platform will support convenient development of Intelligent Virtual Environments. It will support AI-based methods not as additional features but instead as central representation facilities. These representations will provide an enhanced expressiveness of world content and behavior for the design of intelligent VR applications.

References

[1] C. Carlsson and O. Hagsand. DIVE - A Multi-User Virtual Reality System. in *VRAIS'93, IEEE Virtual Reality Annual International Symposium*. 1993.

[2] M.R. Macedonia, M.J. Zyda, D.R. Pratt, P.T. Barham, and S. Zeswitz. Npsnet: A Network Software Architecture For Large Scale Virtual Environments. Presence, 1994. 3(4): p. 265-287.

[3] C. Greenhalgh, J. Purbrick, and D. Snowdon. Inside massive-3: flexible support for data consistency and world structuring. in *Third international conference on Collaborative virtual environments*. 2000: ACM Press.

[4] A.D. Bierbaum. VR Juggler: A Virtual Platform for Virtual Reality Application Development, Master thesis, 2002.

[5] R. Hubbold, J. Cook, M. Keates, S. Gibson, T. Howard, A. Murta, A. West, and S. Pettifer. GNU/MAVERIK: a microkernel for large-scale virtual environments. in *ACM symposium on Virtual Reality software and technology*. 1999: ACM Press.

[6] S. Pettifer, J. Cook, J. Marsh, and A. West. DEVA3: Architecture for a Large-Scale Distributed Virtual Reality System. in *ACM Symposium on Virtual Reality Software and Technology VRST'00*. 2000. Seoul, Korea.

[7] R. Blach, J. Landauer, A. Rösch, and A. Simon. A highly flexible virtual reality system. Future Generation Computer Systems, 1998. 14(3-4): p. 167-178.

[8] H. Tramberend. A distributed virtual reality framework. in *IEEE Virtual Reality Conference*. 1999.

[9] K. Watsen and M. Zyda. Bamboo - A Portable System for Dynamically Extensible, Real-time, Networked, Virtual Environments. in *IEEE Virtual Reality Annual International Symposium*. 1998. Atlanta, Georgia.

[10] A. Kapolka, D. McGregor, and M. Capps. A Unified Component Framework for Dynamically Extensible Virtual Environments. in *Fourth ACM International Conference on Collaborative Virtual Environments*. 2002.

[11] P. Biermann and I. Wachsmuth. Non-Physical Simulation of Gears and Modifiable Connections in Virtual Reality. in *Proceedings Fifth Virtual Reality International Conference (VRIC 2003)*. 2004. Laval, France.

[12] M. Billinghurst. Put that where? voice and gesture at the graphics interface. ACM SIGGRAPH Computer Graphics, 1998. 32(4): p. 60-63.

[13] R. Arangarasan and G.N.J. Phillips. Modular Approach of Multimodal Integration in a Virtual Environment. in *Fourth IEEE International Conference on Multimodal Interfaces ICMI'02*. 2002. Pittsburgh, Pennsylvania: IEEE.

[14] M.E. Latoschik. Designing Transition Networks for Multimodal VR-Interactions Using a Markup Language. in *Fourth IEEE International Conference on Multimodal Interfaces ICMI'02*. 2002. Pittsburgh, Pennsylvania: IEEE Press.

[15] M. Luck and R. Aylett. Applying Artificial Intelligence to Virtual Reality: Intelligent Virtual Environments. Applied Artificial Intelligence, 2000. 14(1): p. 3-32.

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

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

[18] W. Bethel. Scene Graph APIs: Wired or Tired? in *SIGGRAPH'99 Conference Abstracts and Applications*. 1999.

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

[20] G. Heumer. Framework zum Datenaustausch zwischen heterogen strukturierten Simulationskomponenten (*A Data Exchange Framework for Heterogeneously Structured Simulation Components*), Diplom (MSc) thesis, Faculty of Technology, University of Bielefeld, 2003.

[21] M. Schilling. Ein Framework für funktional erweiterbare Semantische Netzwerke in der Virtual Reality (*A Framework for Functionally Extendable Semantic Networks in Virtual Reality*), Diplom (MSc) thesis, Faculty of Technology, University of Bielefeld, 2003.