Semantic Reflection for Intelligent Virtual Environments

Marc Erich Latoschik* Christian Fröhlich†

AI & VR Lab, Bielefeld University

ABSTRACT

We introduce semantic reflection as an architectural concept for Intelligent Virtual Environments (IVEs). SCIVE, a dedicated IVE simulation core, combines modularity with close coupled integrative aspects to provide semantic reflection on multiple layers from low-level simulation core logic, specific simulation modules' application definitions, to high-level semantic environment descriptions. SCIVE's Knowledge Representation Layer provides the central organizing structure which ties together data representations of simulation modules, e.g., for graphics, physics, audio, haptics, or AI etc., while it additionally allows bidirectional knowledge driven access between the modules.

Keywords: Intelligent Virtual Environment, Virtual Reality, Simulation Core, Ubiquitous Computing, Application Framework.

Index Terms: D.2.11 [Software Engineering]: Software Architectures— [I.6.7]: Simulation and Modeling—Simulation Support SystemsEnvironments H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, Augmented, and Virtual Realities I.6.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality I.2.4 [Artificial Intelligence]: Knowledge Representation Formalisms and Methods—

1 INTRODUCTION

Rich believable worlds frequently demand the integration of special purpose software modules, e.g., for the simulation of graphics, sounds, collisions, physics or haptics. Purpose-built VR development tools often adopt design principles of the underlying graphics system like scene and application graphs while they additionally provide VR specific key features [17] like input/output device customization (AVANGO [19], Lightning [6], VR Juggler [5], CAVELib™, WorldToolKit®), network distribution (AVANGO [19], MASSIVE 3 [7], DIVE [8][2], Net Juggler), or entity centered access to world states or world logic often realized using event mechanisms (see, e.g., Lightning [6]).

While well motivated in the beginning, a close coupling between application content and graphics system is disadvantageous [3] [4] w.r.t. extensibility and portability, i.e., during the integration or replacement of additional simulation modules to render animations, sounds, physics, haptics. Furthermore, *intelligent virtual environments* [14] demand the integration of Artificial Intelligence methods. Such modules are either included on a case by case base, or they are integrated a priori into holistic architectures as found in many 3D game engines like the Doom 3 Engine, the Unreal Engine 3, the Source Engine, the C4 Engine or the CryENGINE™

Extensible architectures [1][11] follow a module based approach. The goal is a decoupling of specific application content from the internals of a simulation engine—a challenging task due to the close data and control flow coupling but distinct data representations between the various modules. To achieve this goal, we propose a concept called **semantic reflection**. Semantic reflection extends the existing object oriented programming (OOP) reflection concept to provide knowledge-driven access to objects' capabilities and interfaces. The general idea is based on a semantic representation which comprises logical scene entities [18][16][10], specific modules' data structures and control mechanisms, as well as the internals of the simulation engine itself (see figure 1) as inspired by semantic models for plain OOP [15] reflection.

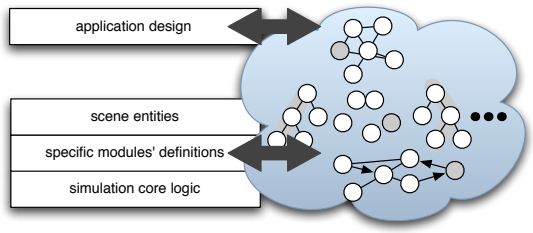

Figure 1: Semantic reflection maps the data and object representations from various simulation applications' layers to a unified semantic knowledge representation.

2 SEMANTIC REFLECTION IN SCIVE

SCIVE, a Simulation Core for Intelligent Virtual Environments implements semantic reflection using a structural knowledge representation layer (KRL) that ties the distributed world representations together. Every application object and world entity from the three layers following figure 1 is mirrored by specific representations using a custom-built semantic net base formalism [13]. Its low-level C++ implementation is linked to an event system that automatically detects read/write accesses between the connected modules and the KRL. Since this inter-module access is synchronized internally by the simulation core [12], different synchronization schemes can dynamically be applied to generate new coherent world states. These synchronization schemes ultimately represent the result of changing flows of control between the modules.

Initially, all required modules are augmented with a wrapper [9] which provides the necessary links to the simulation core and the participating modules' logic as well as the initial bindings between the modules' objects and entities and their KRL counterparts. On the KRL, these counterparts—just nodes in the semantic net—are related to the other parts of the application knowledge to grant access to that information. In contrast to (object local) OOP reflection, semantic reflection provides semantic inter-object information which can be obtained by traversing the relations starting from these specific interlink-nodes which we therefore call **semantic entities**.

SCIVE provides bidirectional application design: The backwards compatible approach still permits developers to implement a required function in a specific module: Wrappers also include mappings from the KRL's semantic entities to their own object model—usually via multiple inheritance—which then have full access to the semantic descriptions of all objects and entities.

*e-mail: marcl@techfak.uni-bielefeld.de

†e-mail: cfroehli@techfak.uni-bielefeld.de

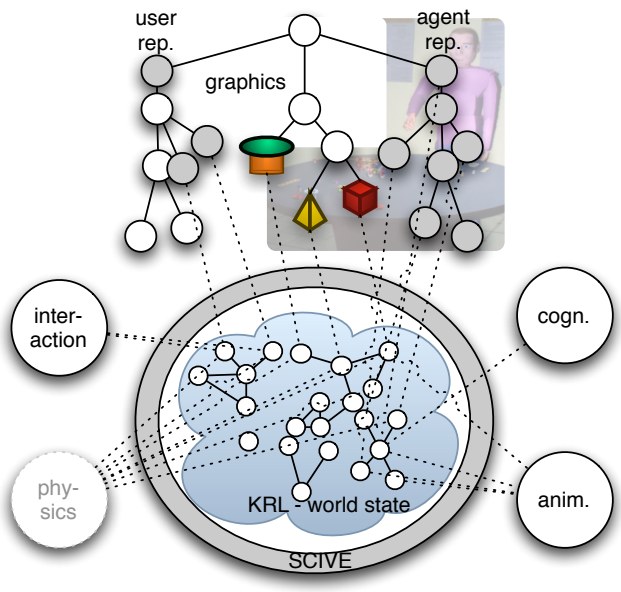

Figure 2: Application layout 2: Interconnecting modules for user interaction, agent perception and animation, KRL as well as—potentially—physics, all using SCIVE (see text).

Figure 2 illustrates the interconnection of several independent modules designed to implement an intelligent agent with perception and action capabilities in a physically simulated virtual world. Both, the artificial agent as well as the user are represented by graphics module specific **semantic entities** which provide a general interface to the KRL (depicted for the nodes with dotted lines to the KRL). Semantic entity nodes are directly placed into the scene graph at the appropriate graphics entities.

One typical solution for the implementation of an agents visual perception is the design of a sensor which automatically tracks objects in its direction and range. Typically, such specialized sensors are often implemented as graphics nodes. This is reasonable since graphics node traversal is synchronized with the application and such a sensor has automatic access to the spatial arrangement of the surrounding scene. One obvious solution to parameterize such sensors is realized by reflection, i.e., by using the runtime-type system to automatically detect compatible target nodes, i.e., nodes the agent should monitor for a given task. Using compiled programming languages like C or C++ this requires source editing and recompilation while such runtime type systems are often limited in terms of their expressiveness, i.e., they are designed to reflect the specific language object model which is often limited to plain hierarchies and—to some degree—multiple inheritance. W.r.t. the chosen KRL base formalism, this type of information can be considered rather simple and hence can easily be made accessible during runtime. The view sensor only searches for entities which have a given semantic tag, i.e., `is_agent_observable` to collect all entities it should monitor since it "knows" what it has to look for. Having collected all target objects, bidirectional semantic reflection enables the sensor to easily reach through and read the position data of the entities for its local spatial ordering.

Since semantic reflection comprises all three layers from core logic, module logic, to scene entities, application design can largely be made modular and extensible. Tools can access object and entity capabilities in a uniform way. Applications range from simple data access and exchange, inter-module synchronization, to ontology based entity access required for smart graphics and IVEs.

References

[1] J. Allard, V. Gouranton, L. Lecointre, S. Limet, E. Melin, B. Raffin, and S. Robert. Flowvr: a middleware for large scale virtual reality applications. In *Proceedings of Euro-par 2004*, Pisa, Italia, August 2004.

[2] E. F. Anthony Steed. Construction of collaborative virtual environments. In M.-I. S. Segura, editor, *Developing Future Interactive Systems*, number ISBN 1591404126, pages 235–268. Idea Group, November 2004.

[3] R. Arnaud and M. T. Jones. Innovative software architecture for real-time image generation. In *Proceedings of the I/ITSEC Conference*, 1999.

[4] W. Bethel, C. Bass, S. R. Clay, B. Hook, M. T. Jones, H. Sowizral, and A. van Dam. Scene graph apis: wired or tired? In *SIGGRAPH '99: ACM SIGGRAPH 99 Conference abstracts and applications*, pages 136–138, New York, NY, USA, 1999. ACM Press.

[5] A. D. Bierbaum, C. Just, P. Hartling, K. Meinert, A. Baker, and C. Cruz-Neira. VR Juggler: A Virtual Platform for Virtual Reality Application Development virtual platform for virtual reality application development. In *IEEE Virtual Reality 2001 conference proceedings*, pages 89–96, Yokohama, Japan, 2001. IEEE Press.

[6] R. Blach, J. Landauer, A. Rsch, and A. Simon. A Highly Flexible Virtual Reality System. In *Future Generation Computer Systems Special Issue on Virtual Environments*. Elsevier Amsterdam, 1998.

[7] C. Greenhalgh, J. Purbrick, and D. Snowdon. Inside massive-3: flexible support for data consistency and world structuring. In *Proceedings of the third international conference on Collaborative virtual environments*, pages 119–127. ACM Press, 2000.

[8] O. Hagsand. Interactive MultiUser VEs in the DIVE system. *IEEE Multimedia Magazine*, 3(1), 1996.

[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] E. Kalogerakis, S. Christodoulakis, and N. Moumoutzis. Coupling ontologies with graphics content for knowledge driven visualization. In *Proceedings of the IEEE VR2006*, pages 43–50, 2006.

[11] 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.

[12] M. E. Latoschik, C. Fröhlich, and A. Wendler. Scene Synchronization in Close Coupled World Representations using SCIVE. *The International Journal of Virtual Reality*, 5(3):47–52, 2006.

[13] 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.

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

[15] J. Meseguer and C. Talcott. Semantic models for distributed object reflection. In *ECOOP 2002 - Object-Oriented Programming: 16th European Conference Malaga*, Lecture Notes in Computer Science, pages 1–36. Springer Berlin / Heidelberg, 2002.

[16] 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*, Ft. Worth, TX, USA, March 2003. IEEE.

[17] S. M. Preddy and R. E. Nance. Key requirements for cave simulations: key requirements for cave simulations. In *WSC '02: Proceedings of the 34th conference on Winter simulation*, pages 127–135. Winter Simulation Conference, 2002.

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

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