Interaction-Centric Modelling for Interactive Virtual Worlds: the APIA Approach

Francois Bernier, Denis Poussart, Denis Laurendeau, Martin Simoneau
Computer Vision and Systems Laboratory
Dept. of ECE, Laval University, Québec, Canada, G1K 7P4
{fbernier,poussart, laurend ,simoneau}@gel.ulaval.ca

Abstract

Conceptual modelling studies the different abstraction methods of the real world. The conception and the execution of virtual worlds depend strongly of the type of conceptual models. Existing modelling methods such as object-oriented modelling are not appropriated when the main concern is the dynamic reusability and interoperability. Such a reusability and interoperability must be free of any human intervention. This paper presents a new paradigm named Interaction-Centric Modelling (ICM) that increases reusability and interoperability of virtual entities and behaviours.

1. Introduction

Virtual worlds can be augmented with highly realistic models and it would be desirable that the developers of these virtual worlds would reuse existing models or entities. Ideally, entities that have not been designed to work together should be able to interact without having to redefine the underlying physical models. Such flexibility is mainly based on the interoperability of entities and reusability of entities and models. Using a common graphical representation format like VRML[1] increases the compatibility and the portability of visual information. However, there is no standard for describing the interaction between independently created virtual entities. Since it is impossible to predict all possible combinations of interactions between entities, the mechanism for implementing the interactions must instead rely on a generic representation of an interaction. This flexibility can be described as achieving dynamic relationships among entities in the virtual worlds[2].

The interaction between entities is a topic that has been analyzed in many fields. The interactions between virtual entities themselves and between virtual entities and real-world users are a subject study in the multi-users virtual environments. The plug-and-jack method[3], the object-oriented interaction model and group objects[4] are three paradigms that increase an entity interaction capability. As another example, DEVA3[5] proposed four sources of behaviour and the solution concerning virtual entity representation is based on components. DEVA3 proposes the use of components to compose objects dynamically, to add behaviours using scripting languages, and to implement dynamic inheritance. However, inheritance restricts the reuse of entities because no universal classification tree can satisfy all designers' requirements.

All the paradigms presented above are entity-centred. The consequence is that the entity can interact with a limited set of entities, according to the compatibility connector plugged on this entity or on the predefined set of known events.

Other approaches are less entity-centred. For instance, Lee[6] presents an interesting method called interaction-based programming that aims to increase the reusability for animations. This kind of programming takes apart the development and the representation of algorithms for computation from the algorithms for coordination. However, this method is specific to animation.

This paper presents rules for designing conceptual models that increase entities and behaviours reusability and interoperability at run-time. The first part presents elements that affect the new abstraction underlying the paradigm. The second part describes a new abstraction technique and a representation of the real world that makes entities and behaviours more flexible (reusable and interoperable) both during construction and execution. This abstraction technique, the Interaction-Centric Modelling (ICM), is the basis of the framework architecture called Actor-Property-Interaction Architecture (APIA) developed in the context of the VERTEX project[7] Finally, APIA's unique features and flexibility are demonstrated through an example.

2. Factors Having an Impact on Conceptual Modeling

The granularity, the level of rigidity, the choice of the entity and the nature of the relations between the entities largely influence the choice of the conceptual models. The following paragraphs will detail each of them.

The granularity has an impact on reusability and interoperability of the conceptual models. Fine-grained entities are flexible but complex to manage and to interconnect. Coarse-grained entities are easier to manage but they are not flexible enough and cannot be reused easily. Since interoperability and reusability are mainly targeted, the resulting solution has to be more granular than existing ones. However, the increasing complexity should be dealt with by implementing an adequate structure and by a proper encapsulation of entities.

The level of rigidity is defined as the number of parameters fixed when defining a conceptual model. For example, the C++ language is undefined in regard of the simulation field, making different simulation models strictly built on this language non-compatible. On the other hand, over-defined architectures are useful for specific fields of application but too restrictive for generic simulation architectures. The proposed solution has to evaluate existing application fields and to find the largest common denominator between these applications in regard of reusability and interoperability criteria.

Finally, the representation of the real world in the virtual world largely depends on the nature of each entity and its relations with others. For instance, scene graph approaches like VRML are based on the geometrical (in opposition to physical) representation of the real world. Aggregation (e.g. a scalpel into a human body) is used without assigning a meaning to the relationship. On the other hand, the object-oriented modelling[8], (OOM), is an example of an abstraction of the real world that implies a set of specific relationships between entities but that leaves the nature of the elements flexible. Most problems encountered in OOM are mainly due to the misuse of inheritance and aggregation. Reuse of behaviours, properties, and entities of the virtual world can be achieved in a better way.

3. The Actor-Property-Interaction Architecture

The approach proposed in this paper is based on a different paradigm that departs from object-oriented modeling. The new representation of the real world into virtual components focuses principally on the interactions between the entities instead of centring the abstraction on entities themselves hence the expression “Interaction-Centric Modeling” which we use to characterize this approach. In brief, the resulting approach is also more granular than OOM. The class concept is fragmented into three elements: property, interaction and character. These basic elements are regrouped into a new container called actor. These actors are linked together with concepts of part-whole relationships borrowed from the cognitive science[9]. The following sections expose the basic concepts of the Actor-Property-Interaction Architecture (APIA) and the rules that are proposed to build more reusable and interactive virtual worlds.

3.1. Definitions

The first step consists in defining the required elements of our virtual world. The user must define or reuse existing basic elements like properties, interactions, characters and actors. Two helper managers can be redefined for specific applications but the existing ones would be sufficient for most applications.

Actor: It is the entity of the simulation. The actor is the “thing” that is simulated. Examples of actors are: submarine, tumor, avatar, atom, mouse, virtual pilot, etc. The actor is similar to a class in object-oriented modeling except that it has neither attributes nor methods. It groups properties (attributes) by dynamic aggregation and encapsulates relationships with other actors when required.

Property: They represent physical or abstract data: mass, velocity, position, Reynolds’s number, money, computer mouse coordinates, etc. Properties are used instead of attributes because they are not encapsulated into an actor. They can exist in many locations at the same time (distributed), have some default behaviors, and can be modified only through specific interactions. Of course smaller-grained properties are easier to reuse.

Interaction: It is the link between the properties belonging to the actors. The behaviors are implemented through the interactions. Examples of interactions are: gravity, heat transfer, robot control, missile launching, etc. They are sequences of code, or algorithms, that apply some modifications on properties when called by the managers. The interaction knows which properties are required. Usually, an interaction should reuse existing properties instead of creating new ones.

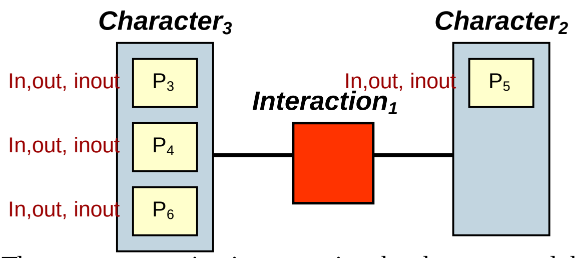
Character: A character is a group of properties defined under a specific name. A property is not specific to a character. For example, a *Collisionable* character will group all required properties like the geometry and the position of an object.

Interaction applicability manager: The first manager decides whether or not and when an interaction must be applied to an actor. It is the control part of the interaction. It can decide whether the call to the interaction must be performed periodically or episodically (event). This manager is also useful to restrict interaction applicability to spatial domains. This kind of restriction limits the domain of interest of the interaction in large environments.

Interaction call manager: The call manager decides how the properties are sent to the interactions.

3.2. Character - Interaction Connection

Figure 1. Global view of character-interaction association



The next step consists in connecting the characters and the interactions together. An interaction can depend on one or more characters, like illustrated in Figure 1. The properties attributed to characters can be used by an interaction as input, output or both. Moreover, default values can be defined at this step in order to implement default behaviors.

3.3. Character-Actor Assignment

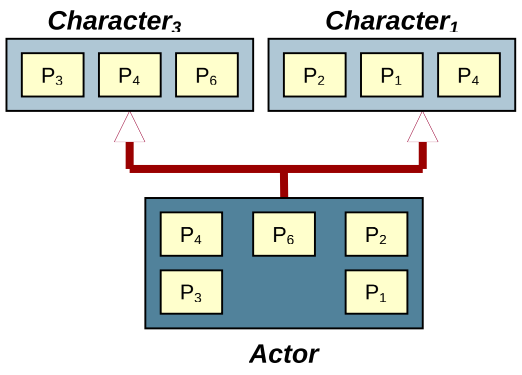


Figure 2. Character assignation to an actor

Assigning a character to an actor allows him to act and behave into an artificial world. As illustrated in Figure2, inheritance is a good way to assign character(s) to an actor. This type of inheritance, which can be compared to the class inclusion relationship described in Winston and Al., means that an actor is a kind of character. In this way, an actor knows how to interact with other actors that inherit from complementary characters. An actor can act and react with external actors in different ways because it is compliant with some interactions. It is not possible for an actor to inherit from another actor because actors form a group concept onto which interactions are applied and within which properties are stored.

Since actors inherit of the properties of the character, values for the inherited properties must be defined. At this step, an actor can be instantiated into the virtual world.

3.4. Actor "part-of" Relationships

An actor can contain other actors. Traditionally, the inclusion relationship was met in the geometrical representation (VRML) of the real world.

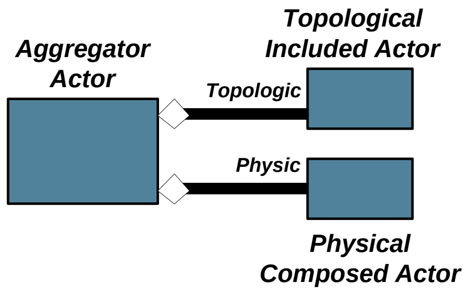


Figure 3. Aggregation of actors

The relationships between actors are useful but have many confusing meanings. For example, "the tumour is in the liver", "the liver is composed of blood" and "the liver is composed of lobes" are all examples of aggregation for which the type of aggregation would be meaningful. APIA allows all kinds of relationships as described by Winston et Al. in the taxonomy of part-whole relationships. The interaction applicability manager at run-time can use this information as will be seen in the following section. Two of them, shown in Figure 2, are described in the following paragraphs.

In the **topological inclusion**, an actor can be described as included, or located, inside a volume or on the surface of another actor. For instance, a scalpel is included into a patient body. This information is useful for the interaction manager applicability. For instance, collision detection between a surgery tool and an organ is not required as long as the tool has not entered the body.

The **physical composition** retains the idea of space or stuff aggregation described by Wilston et Al: component-object, portion-mass and stuff-object inclusion. Some interactions apply differently on actors if these actors are related to other actors by a meronymic inclusion. For instance, a liver is composed of lobes. If the liver is removed from the virtual world, the lobes must be also removed.

4. An Example using APIA

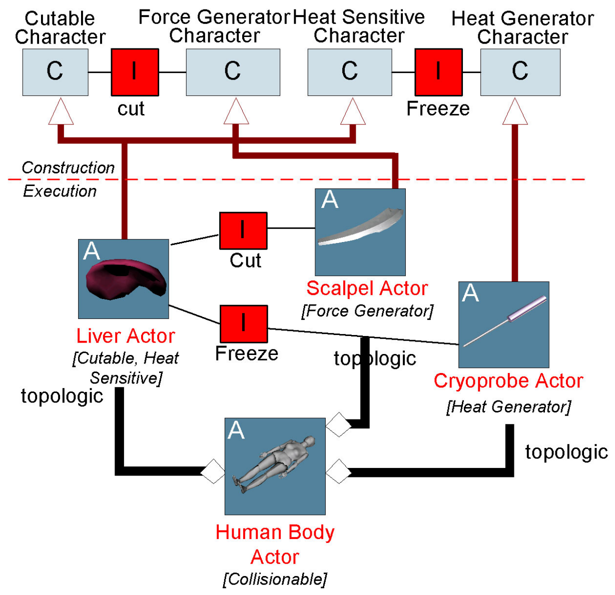


Figure 4. Representation of the construction-execution of the simulation

Figure 4 illustrates an example using APIA in a medical context. In such a virtual world, a surgeon can cut a liver with a scalpel and freeze this same liver with a cryoprobe. Instead of making entities interacting directly, APIA uses an indirection mechanism, the character. The *Cut* interaction applies between the *Cutable* character and the *Force Generator* character. The *Freeze* interaction applies between a *Heat Sensitive* character and a *Heat Generator* character. These characters group some properties required by the interaction. An actor will own all the properties of inherited characters.

Since the dependency between instantiated entities is generic, it is possible for the liver to inherit from a *Heat Generator* character and take into account the fact that the liver is itself a heat generator.

In this example, the *Cut* interaction could be called only when the scalpel is located into the body, i.e. when the scalpel is linked to the body with a topological inclusion. Such a mechanism of activation delegates the control of the applicability to the APIA applicability manager. In this way, a model designer that defines all activation and scheduling rules could execute its model into a complete simulator without requiring a software specialist.

5. Implementation and Applications

The Actor-Property-Interaction Architecture has been implemented into a distributed framework based on a real-time release of CORBA named TAO[10]. In addition to the reusability and the interoperability issues, this framework takes into account other elements such as data distribution, real-time simulation abilities and communication between the real world and the virtual world. Current implementation of this framework runs on Windows, Linux and could be ported to several other operating systems.

APIA has been used in an underwater telerobotics application conducted in cooperation with the Robotics Division of the Hydro Québec Research Institute, IREQ. More specifically, APIA has been applied in a training simulation engine for tele-operated submarines and in the real-time control of a submarine communicating with a control centre with a tether[11].

Cryosurgery planning, training, and assistance to the surgeon also benefited of the same architecture. The SKALPEL project consists in modeling tumor tissue deformations[12] and thermal heat transfer[13] caused by a cryogenic probe. Theses models have been implemented and executed in APIA. Organs, tumors and several cryogenic probes can be assigned to actors. The thermal heat transfer as well as tissue deformation are computed by many interactions. Magnetic resonance images are assigned to properties.

Both applications are being implemented using APIA's simulation engine. Accommodating these two applications has been helpful in validating the APIA paradigm and its current software implementation.

6. Conclusion and Future Work

The APIA architecture increases the possibility of interaction of a virtual entity with a virtual world in a more dynamic way. Instead of centring the communication problem between entities on the entity itself, APIA focuses on an inter-entity paradigm, the Interaction-Centric Modelling (ICM). Moreover, basic elements are finer-grained, thus allowing more dynamic connections. However, such dynamic connections imply more complex scheduling methods. In order to maintain the reusability and interoperability, more information on the dependence between the interaction and the characters will be required for the execution of these interactions.

Acknowledgements

This research was funded in part by the Institute for Robotics and Intelligent System (Networked Centers of Excellence program of Canada), the Natural Science and Engineering Council of Canada, the Canadian Foundation for Innovation, and Fonds pour la Formation de Chercheurs et l'Aide à la Recherche du Québec. This work is also supported by a post-graduate scholarship from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] G. Bell, A. Parisi, and M. Pesce, The Virtual Reality Modeling Language Specification, Available on the Internet at http://www.web3d.org/VRML2.0/FINAL/ (2001).

[2] Araki, Y., "A Model for Dynamic Interaction between 3D-Object in Virtual Environment". *In Proc. of Interaction'98*, 1998, Tokyo, Japan, pp. 73-80.

[3] Araki, Y., "An Interaction Model for Avatar-Habilis, with Capabilities to Use Tools in 3D-MUVEs", *In Proc. of the Virtual Worlds and Simulation Conference*, 1999, San Francisco, CA, pp. 39-44.

[4] Broll, W., "Interaction and Behavior Support for Multi-User Virtual Environments", *In Proc. of the ACM SIVE'95 - First Workshop on Simulation and Interaction in Virtual Environments*, 1995, Iowa City, IA, pp. 246-264.

[5] Pettifer, S., Cook, J., Marsh, J. and A. West, "DEVA3: Architecture for a Large Scale Virtual Reality System", *In Proc. of the ACM Symposium in Virtual Reality Software and Technology*, 2000, Seoul, Korea, pp. 33-39.

[6] LEE, G. S., "Reusable Interactions for Animation", *In Proc. of the 5th International Conference of Software Reuse*, 1998, Victoria, Canada, pp.320-329.

[7] Poussart, D. Laurendeau, D., Bernier, F., Simoneau, M., Harrison, N., Ouellet, D. and C. Moisan, "Designing Virtual Environments for Critical Transactions and Collaborative Interventions: the VERTEX Framework for Networked, Physics-Compliant Object", *SSGRR 2000 Computer & eBusiness Conference*, 2000, L'Aquila, Italy.

[8] Booch, G., *Object Oriented Design with Applications*, Redwood City, CA: Benjamin Cummings, 1991.

[9] Winston, M. E., Chaffin, R., and D. Herrmann, "A Taxonomy of Part-Whole Relations", *Cognitive Science*, 1987, 11, pp. 417-444.

[10] Schmidt, D. C., Gokhale, A., Harrison, T. H., and G. A. Parulkar, "High-Performance Endsystem Architecture for Real-Time CORBA", *IEEE Communications Magazine*, 1997, 14(2), pp. 72-77.

[11] Buckham, B., Nahon, M., and M. Seto, "Three-Dimensional Dynamics Simulation of a Towed Underwater Vehicle", *In Proc. of the 18th International Conference on Offshore Mechanics and Artic Engineering*, 1999, St. John's, Canada.

[12] Schwartz, J. M., Langelier, E., Moisan, C. and D. Laurendeau, "Non-Linear Soft Deformations for the Simulation of Percutaneous Surgeries", *In Proc. of the 4th International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2001, Utrecht, Netherlands, pp. 1271-1272.

[13] Harrison, N., Larose, F., Laurendeau, D. & C. Moisan, "Thermal Mapping with a Neural Network Approach for the Planning and Conduct of MR Guided Cryosurgeries", *In Proc. of the 8th Conference of International Society for Magnetic Resonance in Medicine*, 2000, Denver, CO, pp 1351.