

A Framework for the Development of Applications Allowing Multimodal Interaction with Virtual Reality Worlds

Héctor Olmedo-Rodríguez
Universidad de Valladolid
Dpto. Informática (E.T.S.I.I.)
Campus Miguel Delibes s/n
SPAIN (47008), Valladolid
holmedo@infor.uva.es

Valentín Cardeñoso-Payo
Universidad de Valladolid
Dpto. Informática (E.T.S.I.I.)
Campus Miguel Delibes s/n
SPAIN (47008), Valladolid
valen@infor.uva.es

David Escudero-Mancebo
Universidad de Valladolid
Dpto. Informática (E.T.S.I.I.)
Campus Miguel Delibes s/n
SPAIN (47008), Valladolid
descuder@infor.uva.es

ABSTRACT

With the definition of a mark-up language for specifying scenes, behaviour and multimodal interaction based on the metaphor of interactive movie, we want to propose a framework to develop applications allowing multimodal interaction with virtual reality worlds. Reusing standardized mark-up languages for describing graphics, vocal interaction and graphical interaction, we have defined the basis of an architecture for integrating components of this kind of applications of multimodal interaction with virtual reality worlds. Using this framework we make easier the development of these applications and we provide capabilities to reuse code, graphics and dialogues.

Keywords

Virtual Reality, Behavior, Vocal Interaction, Graphical Interaction, Human-Computer Interaction, Multimodality, Spoken Dialogue Systems, Avatar

1 INTRODUCTION

It's a matter of fact that Virtual Reality Systems (VR) strongly increase the potential of Human Computer Interaction (HCI) [1]. If we consider Spoken Dialogue Systems (DS) add a complementary channel to the graphical channel as sound or vocal channel [2] is, the integration of both researching fields could be seen as a natural evolution of both technologies. For example we could navigate in a VR world using the mouse and we could select objects at the same simply by saying the name of the object. Nevertheless it has been hardly exploded in commercial systems, and despite of existing prototypes [3], there is a work scope to discover: The main reason for not existing hardly integrating solutions of VR and DS is the youth of these work areas, where most efforts have focussed on improving separately both fields despite of studying the necessities of interdependence derived from our

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Copyright UNION Agency – Science Press, Plzen, Czech Republic.

proposal. Here we present a proposal that combines VR-DS posing a platform for developing applications based on 3D virtual worlds allowing multimodal interaction driven by spoken dialogue.

VR and DS fields are characterized by a relative availability of prototypes developed at research laboratories and by some commercial system that generally has neglected the need of adjusting to any development or standard specification. VoiceXML [4] is the standard of DS, on VR there is a standard for scene definition that is X3D [5] evolved from VRML [6]. These standards are a framework of reference for developers to adapt their systems, with the consequent contributions on ease of use referring to definition of 3D scenes and dialogues and on portability of reusable modules. In this paper a framework is presented pretending to be a language for specification of 3D worlds with dialogue integration, a platform to set our worlds in motion and a repository of objects. The solution we contribute with respects the available standards for VR and DS and it's of use to link both worlds, giving argumental coherence to the definition of 3D scenes with spoken interaction.

In this article we introduce firstly mark-up languages for specifying scene and graphical and vocal interaction, and then we introduce the

XMMVR2 process and its defined languages. Next we define the needed architecture to be able to implement this kind of applications and the basis we have centred its design on as well as the used elements. After a comparison with other actual proposals, we end with the conclusions and the future work.

2 3D MULTIMODAL INTERACTION AND MARKUP LANGUAGES

Adding vocal interaction to virtual environments with graphical interaction brings clear profits. Thanks to it we can broadcast commands keeping freedom for hands and eyes. Besides, users can refer to objects outside actual sight of virtual world, making actions to be quick and its effect immediate. But there exists a difficulty for general approximation to multimodal fusion that makes necessary the definition of a reusable architecture for building new multimodal systems. This difficulty resides on the different natures of the metaphors each modality is based on.

The three components of multimodal interaction with 3D environments are: (1) three-dimensional specification that basically consist of modelling objects in the virtual world that can be static and/or dynamic, (2) graphical interaction (GUI) based on keyboard and mouse as we know nowadays and always orbiting to the event model and based on action spaces [7] that are metaphorical approximations for structuring three-dimensional user interfaces (two fundamental: theatre metaphor, locomotion metaphor; five structural: rooms, revolving stages, buildings, space stations, urban metaphors; and five navigational: elevator, train, slide, flying carpet, tele-portage); and finally (3) vocal interaction (VUI) where four metaphors are possible (proxy, divinity, telekinesis or interface agent [8]). To choose the adequate vocal interaction metaphor for our world is a difficult task and even more to specify a language including this inside of the defined framework.

Several mark-up languages exist for specifying vocal interaction, scenes and behaviour separately but there are also other mark-up languages that we will define as hybrid. Examples of mark-up languages for specifying vocal interaction are VoiceXML [4], SALT [9] and X+V [10]. As mark-up languages for specifying scenes we have VRML [6] and X3D [5]. The limitations of these two languages for specifying behaviour of integrating elements of the scene have caused the definition of mark-up languages for specifying behaviour, as for example Behavior3D [11] or VHML [12]. Into the hybrid mark-up languages we can mention MIML [13] that allows integrating speech and gesture information and the

context of a given application using an approach of parsing or syntactic/semantic processing and MPML-VR that is an extension of MPML (Multimodal Presentation Mark-up Language) a mark-up language designed for multimodal presentations using VRML 2.0 to be able to represent three-dimensional spaces through an anthropomorphic agent or human like avatar [14]. Besides of the seen mark-up languages there exist others that look for defining specifications for concrete applications [15]. According to the goal of defining a mark-up language for specifying virtual worlds with multimodal interaction, we propose the XMMVR2 language that will be described on next paragraph.

3 THE XMMVR2 PROCESS

On next figure the two steps of XMMVR2 process are shown:

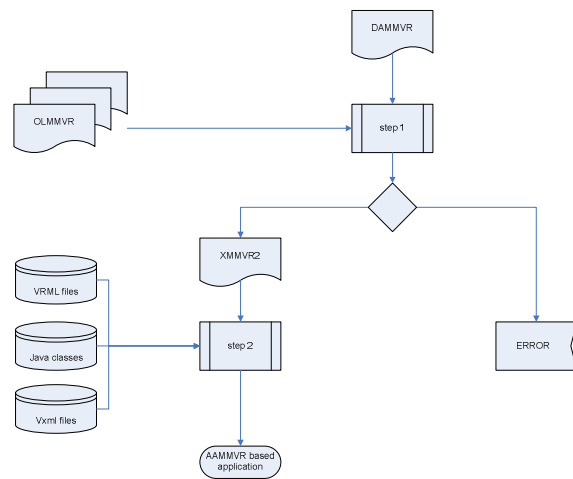


Figure 1 XMMVR2's two steps process

The general idea is developing a repository of objects for building VR worlds with multimodal interaction and reusing those objects being able to define new worlds. Next paragraphs describe each of the components.

3.1 OLMMVR

The extensible mark-up language for Objects Library for MultiModal interaction with Virtual Reality worlds is an XML file referencing to a repository of actors, localizations and dialogues. Actors have a java class inheriting puppet class and a VRML file that determines the appearance of the actor in the world. Localizations are stages represented by its corresponding VRML file. Every dialogue is composed by one or several VoiceXML files.

3.2 DAMMVR

The extensible mark-up language for Definition of Applications of MultiModal interaction with

Virtual Reality worlds is a definition file of the application to be built. In this file we select the actors, the localizations and the dialogues forming part of the defined world or interactive movie. Here we select among the possible actors of our OLMVR repository the ones conforming our cast. Likewise, we choose the stage or stages to configure the movie (see “stage” on figure 2). Finally, we select dialogues guiding our world or interactive movie (see “script” on figure 2).

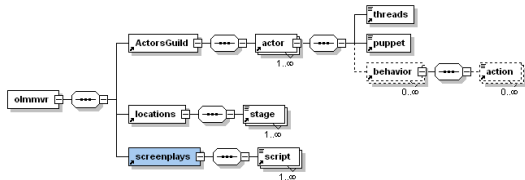


Figure 2 OLMVR's XMLSchema

3.3 XMMVR2

The eXtensible mark-up language for MultiModal interaction with Virtual Reality worlds version 2 is the result of step 1 if everything is correct (see figure 1). On this process we probe that the DAMMVR file defined is valid with respect its DTD and that it references elements existing in the OLMVR repository. If previous condition is not accomplished, the user will be advised with the corresponding error to modify the DAMMVR file.

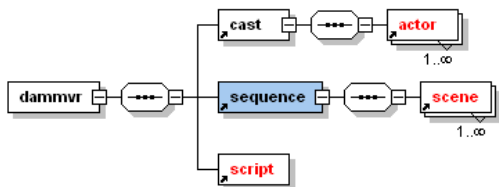


Figure 3 DAMMVR's XMLSchema

If everything is correct, the XMMVR2 file can be defined for describing our world. XMMVR2 is a proposal for definition of a mark-up language for defining scene, behaviour and interaction where we consider every world or interactive movie as an “xmmvr” element based on the cinematographic movie metaphor. We could say that XMMVR2 is a hybrid mark-up language because the idea is to use other languages such as VoiceXML or X+V for vocal interaction and X3D or VRML to describe the scene to be kept embeded on it.

This way, the interpretation of valid XML files according to the XMMVR2 DTD allows the necessary programs and files to make the specified world to work. Our XMMVR2 based system is event driven, so that a minimum list of events will have to

be defined and it does not exist a time line. An “xmmvr” element will be formed mainly by the cast of actors, element “cast” and by the sequence of scenes, element “sequence” that marks the passing of the world. Besides we reserve an element “context” for internal use.

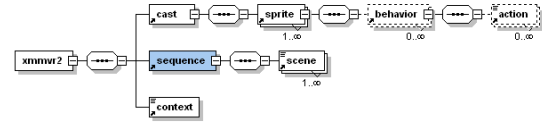


Figure 4 XMMVR2's XMLSchema

Element “cast” will be the set of sprites/actors appearing in the world or “xmmvr” movie, that is every element with a graphical appearance specified by a VRML file and a behaviour allowing interaction with the user. We will consider the user as a spectator with no presence in the world but interacting with the actors included in the cast, so that we use the proxy metaphor already explained to specify vocal interaction so that we base the system on the cinematographic movie metaphor. This could be an evolution from a fundamental metaphor for graphical interaction: the theatre metaphor. We can say an “actor/sprite” is every element taking part of the defining world and having its own behaviour to be specified with the “behaviour” mark. Every “behaviour” will be defined as a couple event and list of actions to play every actor/sprite on a determined condition. An event can be ordered by the user from a graphical interaction “GUI” or a vocal interaction “VUI”. Likewise there exist events from the system to define interaction with other actor/sprite in the world “ACT” or interaction with the world or system “SYS”. The list of actions will be one or several actions generated when an event occurs and can be of graphical character “GUI”, vocal “VUI”, of interaction with other actor in the world “ACT” or of interaction with the world or system “SYS”. Finally, we have to specify the sequence of scenes “sequence” where we have one or more scenes to be presented by default in the written order. In every “xmmvr2” world there is at least a scene and interaction between user and the scene on the world, there must exist by means an actor/sprite living in the defined world. With all those premises we’ve defined a DTD and we’ll be able to develop applications for multimodal interaction with virtual reality worlds based on XML files valid for that DTD that are indeed the specification of the scenes, behaviour and interaction in those virtual worlds.

4 XMMVR2 PLATFORM

The general objective is to dispose of a framework and a language to develop applications of

user commands or behaviour protocols. It offers the possibility of interaction with other languages (specifically languages as Java y JavaScript). To do so Script nodes or an API as EAI are used. This way it is possible to assign more elaborated behaviours as described on the successive.

4.3 Global layered vision

The raised solution to define our control flow platform consists on dividing our system on several superimposed layers where each of them gives support to the immediately upper layer. We will revise the layers conforming our platform and we will place them between two layers: one on the top and another at the bottom to be able to frame our action manager in a context and make it easier to understand. We follow a top-bottom order at the abstraction level as seen on figure:

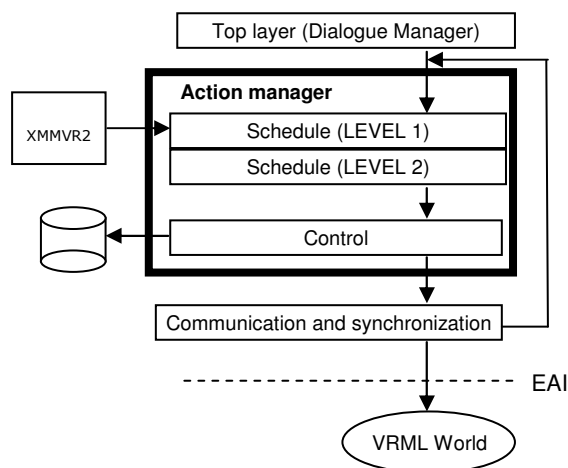


Figure 6 Global layered vision

4.3.1 Top layer

This layer corresponds to a spoken dialogue manager that generates events from vocal interaction with the user and supplies the input for the immediately inferior layer that is the input for our action manager (vocal and graphical).

4.3.2 Action manager

It is identified with a subsystem receiving events; it translates them in series of actions and plans its execution. Later it processes actions interacting with the virtual world through the supplied interface by the puppet to be detailed latter. It must be independent from the characteristics of the virtual world to be controlled. To do so, it is subdivided in two additional layers:

4.3.2.1 Schedule

It is the layer providing a mechanism of unified access for the input of events from different sources. Once inside, these events are translated into complex actions and everyone is subdivided on a series of

simple actions. Such series of actions can be executed on parallel but simple actions originated by the same complex action are executed sequentially. Besides this layer implements an architecture for the treatment of “anti-actions” orders writing off simple actions pending of processing.

4.3.2.2 Control

The input of this layer is simple actions coming from the top layer. Here these actions are executed regarding the state of the world. We call state of the world to the set formed by the states of every dynamic elements making up the virtual world. After the execution of a simple action the state of the world will be updated with the new states.

4.3.3 Communication and synchronization

This is the basis for the execution of actions on the virtual world with an upper level of abstraction [18]. It avoids us to know EAI API used by Java to communicate with the VRML world and provides us a mechanism of synchronization for the interaction with the objects in the world. This layer was redefined to admit the possibility of generating events for a virtual element when clicked with the mouse over its visual representation in the world (clickable puppet) [19]. We have here two extra sublevels:

4.3.3.1 Puppets

It is a layer of greater abstraction; it works with the bottom layer through the EAI to offer a more complex API allowing the possibility to run more simple actions but not close to the VRML referring to upper layers.

4.3.3.2 VRML nodes

It is the bottom layer. It only has the graphical representation to be displayed. It shows its interface to the top layer through the EAI.

It is considered that a puppet is composed of two separate natures: One is a three-dimensional object developed in the model language VRML, this object is just an image without any kind of behaviour or capacity for executing actions. Another is the Java class associated to the object in the virtual world and it gives functionality and capacity to perform actions for the object.

To keep this philosophy we must define VRML nodes in our applications based on a defined format. We do not only need to know VRML specification, but to keep the norms derived from the integration with XMMVR2.

4.4 Event handling

The events arriving to our system are translated into the actions associated according to the XML file valid for the DTD for XMMVR2. If these actions are

high levelled, they are decomposed according to the other two types of less abstract actions. Once the events are translated into successions of less complex actions, these actions are executed on the pertinent puppet taking into account the actual status of the implied puppets and modifying its new state subsequently. If user clicks into the visual representation of any puppet, this will produce its associated event and will feedback it into the subsystem.

Having a clear vision of the case we are dealing, we'll have a great part of the work done when deciding the software architecture to give the subsystem. Besides this way we have the problem explained in a natural way legible for anybody unfamiliar with the analysis and design standards in software engineering.

The hierarchy of events schedule is seen as a series of producers/consumers problems. If a static structure of classes is chosen, the execution of simultaneous actions into the world would be an extremely difficult task. The subsystem would be blocked waiting for the processing of an event and it had not been able to satisfy the execution demand on real time. Besides, it had not been possible the processing of "anti-actions" these are exceptional events able to inhibit the execution of simple actions waiting to be executed.

Threads and queues are the basic components to resolve problems of the kind of producers/consumers but this always implies that some norms have to be carried out to avoid blockade and being able to access to the critical resources safely. For it, instead of using any predefined structure in Java, a class EventQueue has been implemented, designed to satisfy our concrete needs and equipped with the synchronizing mechanisms to assure a correct working. This has been accomplished implementing a class Queue providing mechanisms of insertion and removal of elements over an array of elements.

5 OTHER RELATED WORKS

MMI group from the W3C has marked the lines to follow by the architecture of a multimodal interaction application at the Multimodal Architecture and Interfaces [20]. This architecture is based on a Runtime Framework, providing the Basic structure and managing communication between the other elements or Constituents. The Delivery Context Component, that is a subcomponent of the Runtime Framework, provides information about the capacities of the platform, there is no similar element in our proposal so we will have to develop it in next revisions. The Interaction Manager that is the next subcomponent of the Runtime Framework, manages the different modalities, it could be the Controller

from MVC paradigm and in the AMMVR architecture presented could be similar to the world manager. The Data Component is the last subcomponent from the Runtime Framework and is the provider of the common data model, it would represent the Model from the MVC paradigm and it could be equated to the XML file valid for the DTD of XMMVR2 in our proposal. The Modality Components, put up specific interaction capacities, it would be the Views in the MVC paradigm and the VRML and VoiceXML files for the specification of graphical interaction, scene and vocal interaction respectively in our proposal. Likewise, we could equate EAI API and the servlet we talked about presenting our architecture with the Modality Component API proposed at the Multimodal Architecture and Interfaces.

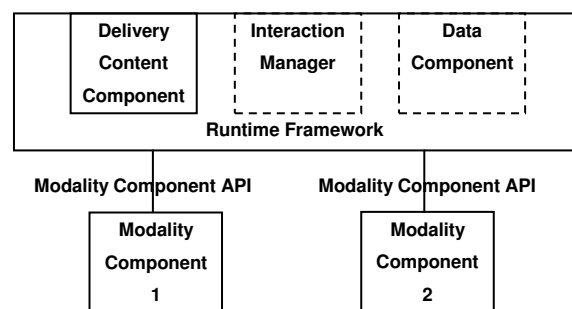


Figure 7 Multimodal Architecture and Interfaces

Besides of this architectural model, the multimodal interaction group from the W3C is developing other series of standards like for example an XML language for transporting input data from different modalities to the interaction manager, EMMA or Extensible MultiModal Annotation Language [26]. It is a specification for establishing how the input or output objects are plugged into the interaction manager, a mechanism for the unification or discrimination into different input modalities for resolving contradictory inputs cases for example, or selection into values from different modalities based on determined rules or priorities but unlike our XMMVR2 language it does not specify behaviour. On COLLADA project [29] an open standard XML Schema is defined for exchanging digital assets among various graphics software applications. Our proposal wants not only to be an exchanging medium inter platforms, we want it to be a platform.

XMMVR2 has similarities with projects as MIML and MPML before described. The last one is a mark-up language designed for multimodal presentations and originally used an agent for animating presentations PowerPoint like. The need of giving more realism to the presentations made to pose the possibility of presenting three-dimensional spaces

despite of simple two-dimensional photographs, this way it evolved to MPML-VR that is an extension of MPML using VRML 2.0 to be able to present three-dimensional spaces through an anthropomorphic agent or avatar with human aspect. It is based on the control of states in a scene, this is controlling transitions on a scene based on elements “event” as a mouse click or a voice command or “speech input”. It is on this event control that it is similar to our proposal but in its concrete object goes far from it. NICE [27] is a project for developing interactive games where a scenario with characters is presented the user can interact with the characters to achieve success in the game. Each character has a domain model corresponding to its vision of the world, ontology that can be described in a tree way and it can be represented in XML format. The world is defined by scenes with several phases on base to some “dialogue events” that generate java classes. Project CONTIGRA [28] has defined Behavior3D, it is an XML Schema automatically generated that integrates every available behaviours for widgets 3D living a repertory of definitions of behaviour and that has been part of inspiration for our proposal thought its objectives are reduced to define behaviour of widgets 3D.

6 CONCLUSIONS AND FUTURE WORK

With the presented proposal of XMMVR2 language we want to prove that it is necessary defining a meta-script to specify any virtual world allowing multimodal interaction. It puts up modularity obtaining clearness and improving possibilities of reusing and standardization. With the end of improving effectiveness of the proposed language, we have implemented the described architecture with a small example application: we have just defined an actor and one stage so we should make more complex the number of actors and stages on future developments. On the other hand, this proposal only considers the proxy metaphor for vocal interaction so we must extend it to give a solution to every presented metaphor of vocal interaction or to all of them globally. As we have used VRML language to specify the elements and graphical interaction based on an obsolete VRML browser as Cortona is, another task will be redefining our architecture to be able to work with the graphical specification language X3D using an adequate browser, this will require redeveloping the architecture based on the SAI API [22] instead of EAI API. Finally, we want our platform to be based totally on open source software for easing the work community desiring to add our proposal to join us without problems, so that we are working with other browsers as FreeWRL [25] or

XJ3D [26] that can be executed on platforms as Debian Linux. Referring to the vocal browsers, we are also posing the searching for open source solutions but nowadays it will be more difficult so by the moment we will keep on with the actual platform without leaving evaluation of other possible candidates. After the definition of a new platform based on open source tools, we will develop an evaluation phase with real users for the defined interaction capabilities to correct and improve our design trying to give an answer to every presented limitations.

Our proposal, differently from the others described has an intention of being a generic platform for development of VR applications with multimodal interaction allowing to experiment with the metaphors owned for every interaction mode, vocal and/or graphical. This way we will get a laboratory for playing with the possibilities that the use of multimodality allowed us. This will make possible the development of applications for specific areas where multimodality could be a great support to ease the interaction. An example could be the development of educational games based on VR for users with accessibility problems.

7 REFERENCES

- [1] W. R. Sherman, A. Craig. *Understanding Virtual Reality: Interface, Application, and Design*, The Morgan Kaufmann Series in Computer Graphics, 2002
- [2] D.Dahl. *Practical Spoken Dialog Systems (Text, Speech and Language Technology)*, Springer, 2004
- [3] C. González-Ferreras, A. González Escribano, D. Escudero Mancebo y V. Cardenoso Payo. *Incorporación de interacción vocal en mundos virtuales usando VoiceXML*, CEIG, 2004
- [4] VoiceXML Forum. “Voice eXtensible Markup Language”: <http://www.voicexml.org> (Revised at December 2007)
- [5] Extensible 3D (X3D): <http://www.web3d.org/x3d.html> (Revised at December 2006)
- [6] Jed Hartman, Josie Wernecke. *The VRML 2.0 Handbook*, Silicon Graphics, 1994
- [7] R. Dachselt. *Action Spaces - A metaphorical concept to support navigation and interaction in 3D interfaces*; User Guidance in Virtual Environments, Workshop "Usability Centred Design and Evaluation of Virtual 3D Environments", 2000
- [8] S. McGlashan, T. Axling. *Talking to Agents in Virtual Worlds*, UK VR-SIG Conf., 1996

- [9] SALT Technical White Paper: <http://www.saltforum.org/whitepapers/whitepapers.asp> (Revised at December 2007)
- [10] XHTML+Voice Profile 1.2: <http://www.voicexml.org/specs/multimodal/x+v/1.2/spec.html> (Revised at December 2007)
- [11] R. Dachselt. *BEHAVIOR3D: An XML-Based Framework for 3D Graphics Behaviour*; ACM Web3D, 2003
- [12] VHML Standard: <http://www.vhml.org> (Revised at December 2007)
- [13] Latoschik, M.E. *Designing transition networks for multimodal VR-interactions using a markup language*, ICMI, 2002
- [14] Naoaki Okazaki y otros. *An Extension of the Multimodal Presentation Markup Language (MPML) to a Three-Dimensional VRML Space*, Wiley-Interscience 2005
- [15] M.P Carretero y otros. *Animación Facial y Corporal de Avatares 3D a partir de la edición e interpretación de lenguajes de marcas*, CEIG, 2004
- [16] ATLAS de IBERVOX <http://www.verbio.com> (Revised at December 2007)
- [17] Rolando Quintero Téllez. *Asignación de Comportamiento Complejo a Mundos Virtuales VRML Utilizando C++*, <http://www.revista.unam.mx/vol.2/num2/art2/index.html> (Revised at December 2007)
- [18] Ignacio Fernandez-Divar Escacho, Alejandro Alcántara Zarzuela, *Herramientas para manipulación dinámica de mundos virtuales*, PFC-ECASIMM (9/2004)
- [19] Samuel García Blanco, *Modelo e implementación de un gestor de acciones en mundos virtuales*, PFC-ECASIMM (9/2005)
- [20] Multimodal Architecture and Interfaces <http://www.w3.org/TR/2006/WD-mmi-arch-20061211/> (Revised at December 2007)
- [21] Cortona <http://www.parallelgraphics.com/products/cortona/> (Revised at December 2007)
- [22] SAI, Scene Access Interface http://www.xj3d.org/tutorials/general_sai.html (Revised at December 2007)
- [23] Andrew M Phelps, Rochester Institute of Technology, Department of Information Technology, *Introduction to the External Authoring Interface, EAI*. <http://andysgi.rit.edu/andyworld10/gallery/archive/s/vrml/media/eaiclass.doc> (Revised at December 2006)
- [24] FreeWRL <http://freewrl.sourceforge.net/> (Revised at December 2007)
- [25] XJ3D <http://www.xj3d.org/> (Revised at December 2007)
- [26] EMMA <http://www.w3.org/TR/emma/> (Revised at December 2007)
- [27] NICE <http://www.niceproject.com/> (Revised at December 2007)
- [28] CONTIGRA http://www-mmt.inf.tu-dresden.de/Forschung/Projekte/CONTIGRA/index_en.xhtml (Revised at December 2007)
- [29] COLLADA <http://www.collada.org> (Revised at December 2007)