Building Dynamic User Interfaces of Virtual Reality Applications with X-VRML

Krzysztof Walczak, Wojciech Wiza
Department of Information Technology, Poznań University of Economics, Poznań, Poland
{walczak,wiza}@kti.ae.poznan.pl

Abstract: A new approach to dynamic creation of user interfaces in virtual reality environments is presented. In the proposed approach, templates based on a new virtual reality language X-VRML are used. An X-VRML template consists of the user interface specification and the 3D environment model. Both parts are specified in X-VRML and interpreted by the X-VRML interpreter. A visualized instance of the user interface is used to supply the 3D environment model with initial parameter values that permits to instantiate a 3D scene. In the proposed approach, the specification of the user interface is abstract and does not rely on any particular implementation of visual elements, allowing creation of different user interface implementations according to an actual system architecture, availability of visual component libraries, as well as user needs and preferences.

Key words: Virtual reality, user interfaces

1. INTRODUCTION

During the last couple of years, we have witnessed a significant progress in computing and networking technologies. The broad accessibility and widespread acceptance of Internet, promotes the World Wide Web to be a universal platform for building network applications instead of a medium for sharing static documents. Potential application areas were highly enhanced by the use of such technologies as: dynamic HTML, cascaded style sheets (CSS), scripting languages (e.g., JavaScript, PHP), and ActiveX objects. One

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35616-7_23
W. Cellary et al. (eds.), *Internet Technologies, Applications and Societal Impact*

of the most important milestones in the Internet application development was introduction of the Java programming language. Other programming technologies like servlets, JSP, ASP, and ColdFusion, permit to create a complex and content-rich applications.

The progress in Web based applications also affected their user interfaces. While first Internet applications could only benefit from simple forms implemented in HTML, today's systems interact with a user by HTML forms, Java graphical user interfaces (GUI) including a rich set of visual elements developed in the Swing library, or interfaces exploiting powerful capabilities of Flash technology. With the introduction of 3D technology to Web applications and emerging visualization standards like VRML and X3D, it is also possible to create a 3D interfaces. Such interfaces are used to attract user with a near-physical feeling of presence in a particular place [RealImation], visualization of resources [Antarctica], or virtual interaction with other users [Blaxxun].

The main obstacle to create virtual reality applications by the use of VRML was declarative method of 3D environment specification with only minimal support for algorithmic processing (by the use of *Script* node). The VRML language has also limited possibilities of programming user interactions. Creation of advanced applications became possible with the introduction of the X-VRML language [X-VRML, Walczak02A, Walczak02B].

The X-VRML language has been designed to enable dynamic modeling of virtual reality. The X-VRML language permits building of active applications that use parameterized models of virtual scenes and dynamic generation of their instances basing on: models, current values of model parameters, data provided by a user, user preferences, and the current system state. The X-VRML language provides convenient access to databases. Data retrieved from a database can affect all aspects of the dynamically generated virtual scenes – the contents, the visualization methods, and the structure. Furthermore, X-VRML offers parameterization methods and programming concepts known from procedural languages, like: loops, conditions and variables, which – combined with the declarative approach of VRML – result in a powerful programming tool. Moreover, X-VRML supports object-oriented programming style by enabling inheritance hierarchies of classes that can be used in the virtual scene models. The X-VRML language is based on XML, which is currently the de-facto standard for creating new languages in this domain.

First experiences with virtual reality applications revealed that a non-intuitive navigation within the 3D environment and a limited interaction with its objects make building effective 3D user interfaces difficult. To solve this problem, 3D environments are equipped with an additional 2D user interface

containing standard visual elements like menus, buttons, text fields, etc. Such a 2D user interface can partially or entirely assure user interaction with the system. Since such a 2D interface is specific for a particular 3D environment, its visualization has to be performed by an application built especially for this 3D environment. This implies that such application cannot be reused for another 3D environment, even if differences in user interfaces are small. Moreover, these applications are usually platform-specific, because they use a particular set of visual elements. Another important difficulty requiring a proper solution is communication between 2D user interface elements (e.g., push button) and visual components of user interface realized in 3D technology (e.g., TouchSensor bind to a VRML node).

In this paper, a new approach to building user interfaces for virtual reality applications is proposed. In this approach, the concept of an X-VRML template is used. An X-VRML template consists of the user interface specification and the 3D environment model. Both parts are specified in the X-VRML language and interpreted by the X-VRML interpreter. The specification of the user interface is abstract and does not rely on any particular implementation of visual elements, so for every application a designer can dynamically create adequate user interface basing on the same specification. The use of the X-VRML language for specification of both 2D interface and 3D environment, followed by the use of the same X-VRML interpreter for the user interface and the model instantiation, assures their seamless integration and permits easy exchange of events and messages between a 3D scene and a user interface.

The reminder of this paper is organized as follows. In Section 2 an overview of the X-VRML language is presented. In Section 3 the concept of dynamic creation of user interfaces is described. In Section 4 and 5 different implementation architectures with application examples are presented. Finally, Section 6 concludes the paper.

2. THE X-VRML LANGUAGE

2.1. The X-VRML language overview

The X-VRML language is based on XML [XML]. An X-VRML model is an XML representation of an algorithm that generates the final content. The model is a program composed of X-VRML commands. The program can read data from a database and use values of model parameters as the input. Examples of commands are: `set` to assign a value to a variable, `for`

implementing a numerical loop, `if/then/else` implementing conditional statements, and `class/instance` representing classes and their instances. The X-VRML commands are encoded in XML and placed inside the text of the content description written in VRML, MPEG-4 BIFS-Text, X3D, or XMT. The fact that the target language (e.g., XMT) contains XML tags is not an obstacle due to special processing of the source X-VRML file: all XML tags that are not known to the X-VRML processing unit are simply ignored and included in the outcome as fragments of the content description.

In X-VRML, empty XML elements represent single-line commands such as `<set>` or `<insert>`. Non-empty elements are composed of start-tags, end-tags, and element body. The non-empty elements represent block-statements like loops, conditions, database queries, iterations, etc. Examples of non-empty elements are `<for> ... </for>` and `<db_query> ... </db_query>`. Non-empty elements can contain fragments of content description and X-VRML code. The code located inside a repeating element is interpreted multiple times. Parameters for the command execution are provided in the form of values of element attributes. In X-VRML all parameters are expressions and are evaluated prior to command interpretation. Valid nesting of X-VRML elements is defined in the Document Type Definition – DTD.

Below, an example of a numerical loop is presented. The loop uses an "i" control variable, varying from 0 to the value of 2k+10 (k is an X-VRML variable) and incremented by 1:

```
<for name="'i'" from="0" to="2*$k+10" step="1">
...
</for>
```

A comparison of Java and X-VRML representations of the same simple algorithm is shown in Table 1.

Table 1. Comparison of the same algorithm represented in Java and X-VRML

Java version	X-VRML version

`for (int i = 0; i <= 10; i++)`
`{`
`  int x = i*i;`
`  if (i < 5)`
`  {`
`  ...`
`  }`
`  else`
`  {`
`  ...`
`  }`
`}`
	`<for name="'i'" from="0" to="10"`
	`  step="1">`
	`  <set name="'x'" value="$i*$i"/>`
	`  <if condition="$i<5">`
	`  <then>`
	`  ...`
	`  </then>`
	`  <else>`
	`  ...`
	`  </else>`
	`  </if>`

2.2. Modularization of X-VRML

Functionality of the X-VRML language is divided into modules. Four different modules of X-VRML are commonly used: Core Module, Object-Oriented Module, Dynamic Module, and Database Module.

The X-VRML Core Module provides basic language functionality that can be used in most application domains. The basic functionality allows assigning values to variables, inserting calculated values of expressions to the output scene description, conditional interpretation of fragments of the X-VRML code, loops, and nesting X-VRML models. More details on the X-VRML Core Module can be found in [Walczak99].

The X-VRML Object-Oriented Module provides methods of implementing classes, building class inheritance hierarchies, and using instances of classes. The Object-Oriented Module has been described in details in [Walczak02A].

The X-VRML Database Module contains language constructs that allow access to miscellaneous data sources including relational and object-oriented database systems [Walczak02B].

The X-VRML Dynamic Module is a set of X-VRML commands allowing implementation of dynamic virtual scene models, i.e. models of scenes whose structure changes during the virtual scene run-time. The change in the scene structure can be a result of the dynamism coded in the scene model, the user interaction, or new data read from a database. More information about the Dynamic Module can be found in [Walczak01].

One of the fundamental concepts of X-VRML is extensibility. New elements – performing more advanced tasks – can be easily added to the language. Adding new elements can be beneficial in applications that use complex processing or are time-critical. Usually interpretation of a single complex element will be faster than interpretation a long X-VRML program composed of simple elements. Extensibility allows to create additional, specialized X-VRML modules with functionality specific for particular domain or application.

3. X-VRML Dynamic User Interfaces

3.1. User Interface Instantiation

To create a 3D scene from an X-VRML model, the set of input parameters has to be supplied. Values of these parameters are provided from the associated user interface, in interaction with a user or by the use of

default values. Using a traditional approach, where a user interface is hard-coded in the application visualizing the model, for each model an application implementing its user interface has to be developed.

In the approach proposed in this paper, the abstract representation of the user interface is included in the X-VRML document. Together with an X-VRML model, it forms an X-VRML template. This representation does not rely on any particular implementation of the user interface (like, for instance, in a persistence model for JavaBeans [JavaBeans]). The same user interface can be instantiated as an HTML form using standard set of HTML tags presented in a web browser. Another way of user interface presentation may be a Java application running on a server or a Java applet embedded in the web browser. In such case, the Swing library of visual elements or standard Java GUI can be used. The user interface can be also created by a C program using TCL/TK visual elements. In some cases, parts of the user interface can be included in the 3D scene, e.g., as a set of 3-dimensional buttons.

3.2. X-VRML templates

```
<interface background_color="255,255,0"
background_texture="http://somehost.com/bg.jpg">
       <component    type=""
                     name=""
              label=""
              value=""
              action=""
     />
     <listener       name="listener_name"
              node="nodeId"
              event="event_out"
              action="action_launched"
       />
       <initial_action action=""/>
       <initial_action action=""/>
</interface>
<model>
       #.... X-VRML code
       <namednode name="nodeId">
              # VRML node, event source for listener
       </namednode>
</model>
```

Figure 1. Sample X-VRML template

A template is an X-VRML document composed of two parts: the interface definition, which describes a user interface, and the model, which contains the X-VRML model of a 3D scene (see Figure 1). The interface part

is used to instantiate the user interface. The X-VRML model instantiation requires a set of parameters that are supplied by the user interface.

The interface part contains an abstract representation of visual components that are required to enable instantiation of the X-VRML user interface. The definition of the component specifies: the type of the control element (e.g., button, label, menu), visual properties of the control (e.g., color, font, position in the interface), default value returned by the component, and possible specification of actions associated with the component. The number of component attributes varies depending on its type. For instance, the *label* component has no *action* attribute, because it is non-interactive element of a user interface.

A particular type of the user interface component is a listener, which is not visualized as an independent visual control, but ties an action with a 3D object (VRML node) of the 3D scene. The listener specification contains the name of a VRML node, the name of an event, which should be listened, and the specification of actions, which should be launched when the event is sent. Such a solution permits to create integrated 2D and 3D interfaces, where some input parameters can be supplied by the buttons and text fields, while others by interaction with the 3D scene. These interactions include selection or manipulation of the 3D object or navigation within the scene (e.g., approaching a node can launch a particular action).

3.3. Actions

The use of interface controls may trigger execution of specific operations. For instance, pressing a button may start processing of the scene model. Another example is clicking on a 3D object, which causes modification of its color. In the traditional approach, the assignment of user interface controls to operations is hard-coded (see Figure 2). In the proposed approach, elements of the user interface can be flexibly assigned to actions. The action is an application function exposed for use in the user interface. The assignment is specified in the *action* attribute of the component definition. The assignment can be done dynamically during the interpretation of the X-VRML template and can be parameterized using X-VRML expressions (see Figure 3).

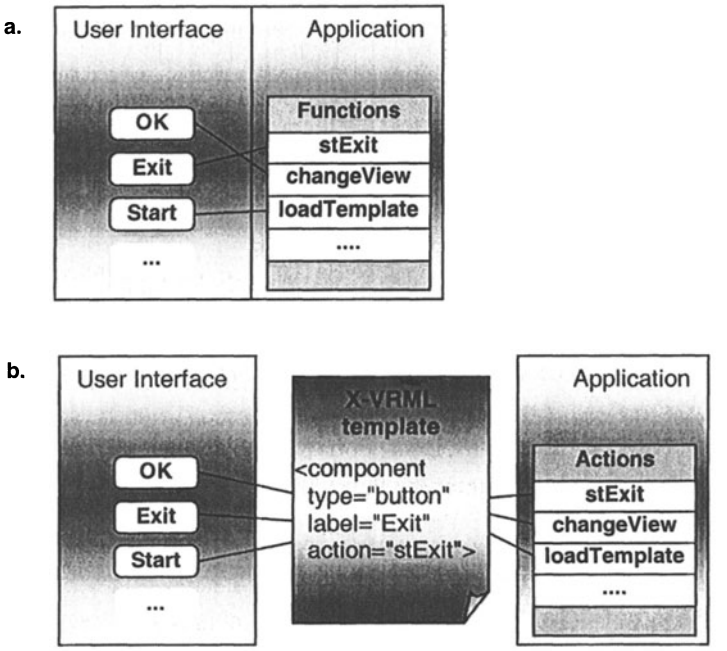

Figure 2. Relationships between user interface elements and actions:
a. traditional approach,
b. proposed approach.

```
<interface background_color="255,255,255">
       <component     type="button"
                      label="OK"
                     action="loadTemplate(@template_name)"
#loading of the template specified in
#the template_name variable
     />
     <if condition="@chkbtn_tour==true">
        <then>
              <component     type="button"
                             label="Start tour"
                             action="changeViewPoint"
#this control will be visible only
#if chkbtn_tour variable has TRUE value
                    />
          </then>
       </if>
   </interface>
```

Figure 3. An example of dynamic action assignment and action parameterization

In some cases, one user interface control can launch a chain of actions, where results of execution of one action influence execution of the next one in the chain.

Actions can be connected with: user interface (e.g., addition or removal of user interface components, component enabling, setting of component values); 3D scene (e.g., cleaning the scene, adding or removal of 3D objects, changing attributes of 3D objects); or the system itself (e.g., loading another template, reprocessing the current template, exiting).

In the interface specification, a set of initial actions can be specified. Such actions are launched once during the processing stage of the template (see Figure 1).

4. Implementation Architectures

4.1. Server-side architecture

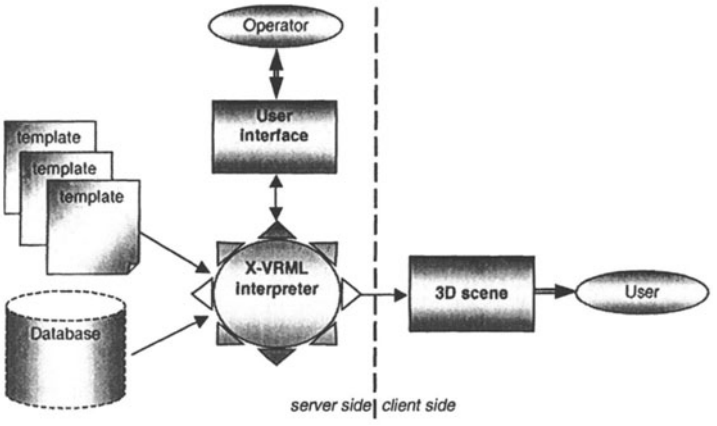

Figure 4. The general architecture of a server-side application

In Figure 4, the overall architecture of a server-side application, which performs X-VRML template interpretation, is presented. The X-VRML interpreter loads templates stored locally on the server and creates the user interface. Entire interaction with a user is performed on the server side. The instantiated model, completed with parameters supplied in the user interface by a system operator and/or the data from the database, is sent to the client.

In this case, a rendered 3D scene is delivered to the end user as a static 3D environment. No returning interactions are sent back to the server.

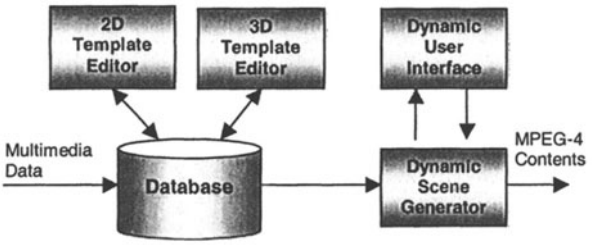

Figure 5. The architecture of the PISTE DUI application

The architecture of an application, which utilizes the servers-side concept, is presented in Figure 5. The application permits to dynamically generate MPEG-4 content [Koenen] for use in TV broadcast. The operator of the application is supplied with a Dynamic User Interface (DUI), which permits to fill a predefined set of templates with multimedia objects like video, audio, still images, 3D environments, etc. The DUI (see Figure 6) is generated by a Dynamic Scene Generator according to definitions retrieved from templates.

The application presented in Figure 5, written in Java and using Oracle RDBMS as a data store, is a part of the larger system – PISTE – designed to produce and deliver sport-related interactive multimedia content [PISTE, Boyle, Demiris01, Demiris02].

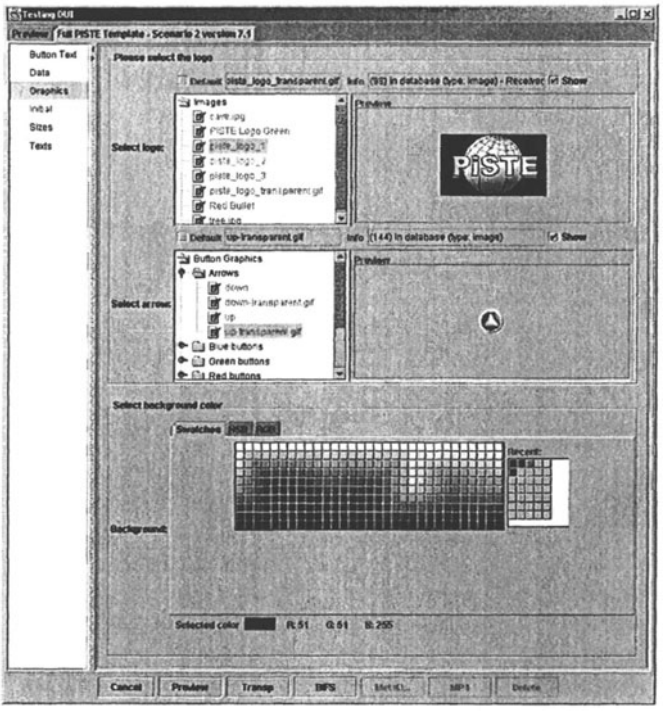

Figure 6. Dynamic User Interface in the PISTE application

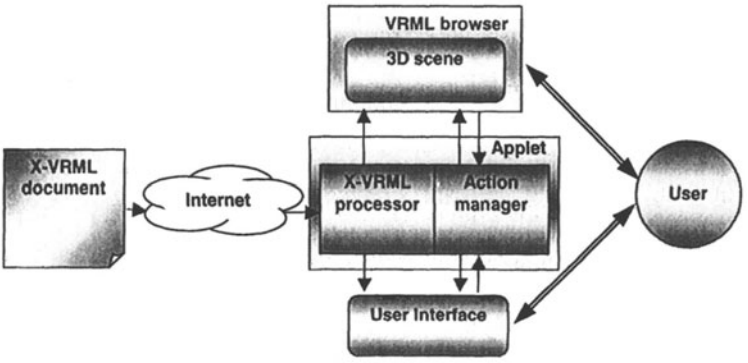

Figure 7. The architecture of the client-side application

4.2. Client-side architecture

The architecture of the system, which performs X-VRML template interpretation on the client side, is presented in Figure 7. In this case, the X-VRML processor is embedded in a Java applet running in the user browser. The applet retrieves X-VRML template from the remote web server and processes it. According to the user interface definition found in the template, the applet dynamically creates a user interface. Applying values retrieved from the user interface to model parameters, the applet creates the 3D scene and sends it to a VRML browser supporting external authoring standard like EAI or SAI (e.g., Cosmo Player, ParallelGraphics Cortona). Events and messages from the user interface and the 3D scene are processed by the action manager. This module is responsible for applying actions specified in the X-VRML template.

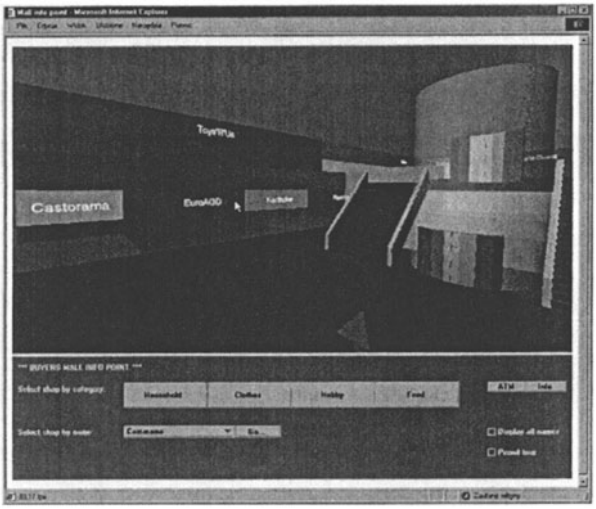

Figure 8. Mall Guide application visualized in the browser

```
<?xml version="1.0"?> <xvrml_document type="XVRML"> <vrml_header/>
<interface background_color="58,110,165">
        <component type="label" value="Select shop by name:"  ...../>
        <query NAMES="'shop_name'"
               SQL="'SELECT NAME FROM MALL_LOCATION'">
                <set NAME="'shop_list'" VALUE="@shop_list+@shop_name"/>
        </query>
        <component type="combo" value="@shop_list" ..../>

#[.....  Other component definitions  .....]

        <listener node="'TS1'" event="'touchTime_changed'
         action="startTour(1)"/>
</interface>
<model>

#[.....  X-VRML code of the model  .....]

                DEF info_point Group {
                        children [
                                <namednode name="'TS1'">
                                        DEF TS1 TouchSensor {} </namednode>
                    <inline URL="infopt.xvrml'" IN="TRUE" OUT="TRUE"/>
        ]}

#[.....  X-VRML code of the model  .....]

</model>
</xvrml_document>
```

Figure 9. A sample code of the Mall Guide application

In Figure 8 an example of the client-side application is presented. The application is a virtual mall guide that permits a user to locate shops of his/her interest. The user interface is visualized in the applet pane as a set of Swing visual controls. This set contains buttons, menu, and checkboxes. Using this interface a user can, for instance, select a particular shop from the menu; select a group of shops sharing the same category of products (e.g., food, clothes); switch on and off shop names to be displayed. The user interface has also 3D interaction element in the 3D scene defined by a listener (see Figure 9). The listener is connected with a *TouchSensor* and listens to its *touchTime_changed* event. When the event is sent by the *TouchSensor*, the *startTour* actions is launched. The *TouchSensor* is bound by a *Group* node with a set of objects, forming a virtual information stand. This virtual stand is a 3-dimensional element of the dynamic user interface allowing a user to start a virtual tour round the mall.

5. CONCLUSIONS

The approach to dynamic creation of user interfaces presented in this paper was used in several different applications of virtual reality, which vary in architectures, application areas, and used technologies. In all these systems, the use of the presented approach enhances system capabilities and makes them flexible and adaptable. The use of the X-VRML language to both user interface specification and 3D environment model, followed by the use of the same X-VRML processor, reduced application complexity.

X-VRML templates can be used to create application-independent user interfaces. Moreover, the same X-VRML template may be applied to creation of entirely 2D user interface in one case, and entirely 3D user interface in another case. An application may flexibly change the method of user interface component visualization basing on accessibility of visual component libraries. In the created system, the applet renders visual components using standard Java GUI, instead of unreachable Swing library. However, if the Swing library is accessible, the applet uses its visual elements for user interface creation.

The concept of actions and specification of connections between user interface elements and actions in the X-VRML template permits to create flexible but easy to design systems.

REFERENCES

[Antarctica] Antarctica project website, http://maps.map.net/start

[Blaxxun] Blaxxun corp. website, http://www.blaxxun.com/

[Boyle] Boyle E., W. Cellary, O. Huminiecki, W. Picard, M. Stawniak, K. Walczak, R. Wojciechowski, *Dynamic Creation of MPEG-4 Content with X-VRML*, Workshop on Business Applications of Virtual Reality, BIS-2002, Poznan, Poland, April 25, 2002

[Demiris01] Demiris, A. M., M. Traka, E. Reusens, K. Walczak, C. Garcia, K. Klein, C. Malerczyk, P. Kerbiriou, C. Bouville, E. Boyle, N. Ioannidis, *Enhanced sports broadcasting by means of augmented reality in MPEG-4*, The International Conference on Augmented, Virtual Environments and Three-Dimensional Imaging, Ornos, Mykonos, Greece, May 30 - June 1, 2001

[Demiris02] Demiris, A. M., K. Walczak, J. Wingbermühle, C. Malerczyk, M. Traka, P. Kerbiriou, D. Curet, K. Klein, E. Reusens, E. Boyle, C. Garcia, N. Ioannidis, *Sprinting along with the Olympic Champions: Personalized, Interactive Broadcasting using Mixed Reality Techniques and MPEG-4*, Workshop on Business Applications of Virtual Reality, BIS-2002, Poznan, Poland, April 25, 2002

[JavaBeans] Persistence model specification for JavaBeans, http://java.sun.com/products/jfc/tsc/articles/persistence/

[Koenen] Koenen, R., *Overview of the MPEG-4 Standard*, V.18, ISO/IEC JTC1/SC29/WG11, Coding of Moving Pictures and Audio, March 2001; http://mpeg.telecomitalialab.com/standards/mpeg-4/mpeg-4.htm

[PISTE] *PISTE - Personalized, Immersive Sports TV Experience*, Fifth European Union RTD Framework Programme, IST-1999-11172, http://piste.intranet.gr/

[RealImation] RealImation corp. website, http://www.realimation.com/product/custportfolio/cityvis.htm

[Walczak99] Walczak K., W. Cellary, *XML-based Dynamic Generation of Virtual Scenes*, Proc. of the 5th International Conference on Virtual Systems and Multimedia VSMM'99, pp. 335-348, Dundee, Scotland, UK, September 1-3, 1999

[Walczak01] Walczak K., *Database Modeling of Virtual Reality*, Doctoral Dissertation, Technical University of Gdansk, Poland, 2001 (in English)

[Walczak02A] Walczak K., W. Cellary, *X-VRML - XML Based Modeling of Virtual Reality*, Proc. of the 2002 International Symposium on Applications and the Internet (SAINT-2002), Nara, Japan, Jan. 28-Feb. 1, 2002, pp. 204-211

[Walczak02B] Walczak, K., W. Cellary, *Building Database Applications of Virtual Reality with X-VRML*, Proc. of the 7th International Conference on 3D Web Technology (Web3D-2002), Tempe, Arizona, USA, Feb. 24-28, 2002, pp. 111-120

[XML] Extensible Markup Language, http://www.w3.org/XML/

[X-VRML] X-VRML website, http://xvrml.kti.ae.poznan.pl/