Technical University of Gdańsk

Faculty of Electronics, Telecommunications and Informatics

Metadata Schema of Interactions for Multimedia Objects

Jacek Chmielewski

Doctoral dissertation

prepared under supervision of

Prof. Wojciech Cellary

Poznań, September 2007

Contents

Streszczenie

Zagadnienie opracowania metadanych dla interaktywnych obiektów trójwymiarowych jest nowym problemem, którego znaczenie wynika z rosnącego zapotrzebowania na interaktywne aplikacje wirtualnej rzeczywistości. Szybkość przygotowywania takich aplikacji zależy w dużym stopniu od możliwości wielokrotnego stosowania istniejących obiektów trójwymiarowych, które cechują się pewnym zachowaniem. Efektywne wyszukiwanie interaktywnych obiektów trójwymiarowych odpowiednich dla nowoprzygotowywanych aplikacji, wymaga opisania tych obiektów za pomocą metadanych obejmujących nie tylko geometrię i semantykę, ale również zachowanie.

Istniejące standardy metadanych umożliwiają zapis wielu informacji dotyczących obiektu. Standardy ogólne, jak na przykład DublinCore, czy XMP, pozwalają na przekazanie informacji takich jak tytuł, autor, czy data utworzenia danego obiektu. Standardy specjalistyczne, takie jak MPEG-7, czy MXF DMS-1, umożliwiają zawarcie w metadanych informacji na temat technicznych cech obiektu i jego semantyki. Brakującym fragmentem metadanych, nieuwzględnionym w istniejących standardach, jest opis zachowania obiektów trójwymiarowych w sposób umożliwiający efektywne wyszukiwanie obiektów o określonych zachowaniach.

W tej rozprawie przedstawiono sposób opisu zachowania obiektów przez modelowanie ich potencjalnych interakcji z innymi obiektami. Na zaproponowane rozwiązanie składają się dwa zasadnicze elementy: model interakcji obiektów trójwymiarowych i koncepcja interfejsu interakcji. Na ich podstawie opracowano w rozprawie specjalistyczny podjęzyk zapytań, który umożliwia efektywne wykorzystanie metadanych interakcji.

Model opisu interakcji obiektów trójwymiarowych, nazwany *Multimedia Interaction Model* – MIM, pozwala na dekompozycję ogólnej interakcji dowolnego obiektu na komponenty, które są następnie opisywane w postaci odrębnych metadanych. Konstrukcja tych metadanych umożliwia stosowanie zarówno słownych opisów semantycznych, jak i sformalizowanych opisów matematycznych. Słowne opisy semantyczne mogą być stosowane na różnych poziomach struktury metadanych. Dzięki temu jest możliwe równoległe stosowanie opisów ogólnych, dotyczących interakcji jako całości, oraz szczegółowych, dotyczących wybranych aspektów interakcji. Ponadto, takie opisy mogą być powiązane z ontologią specyficzną dla danej dziedziny, co dodatkowo zwiększa potencjał informacyjny metadanych interakcji i wypływa na jakość wyników wyszukiwania. Natomiast opisy matematyczne służą do

konstrukcji metadanych umożliwiających wyznaczenie stanu obiektu po interakcji i wykorzystanie go jako punktu wyjścia do kolejnych interakcji. Takie podejście pozwala wyszukiwać obiekty nie tylko na podstawie ich obecnego stanu, ale również stanów, w których obiekty mogą znaleźć się w wyniku interakcji.

Model MIM i wynikające z niego metadane interakcji są ściśle powiązane z nowym podejściem do konstrukcji metadanych. W zaproponowanym podejściu obiekty interaktywne są traktowane w podobny sposób jak elementy programu komputerowego, a nie jak statyczne przedmioty. Podstawą tego podejścia jest koncepcja interfejsu interakcji o nazwie *Interaction Interface* – II, która obejmuje definicję struktury interfejsu interakcji oraz zasady tworzenia nowych interfejsów interakcji. Zastosowanie interfejsów interakcji pozwala na precyzyjne określenie semantyki parametrów biorących udział w interakcji, co znacząco upraszcza wyszukiwanie obiektów o określonych cechach interakcji.

Koncepcja przedstawiona w rozprawie i zaimplementowana w postaci schematów metadanych w formacie XML Schema została zweryfikowana przez jej praktyczne zastosowanie do opisu szeregu złożonych obiektów interaktywnych. Każdy opis został przeanalizowany pod kątem możliwości zastosowania go w różnych scenariuszach wyszukiwania obiektów na podstawie metadanych interakcji.

Abstract

The problem of metadata for interactive 3D objects appeared very recently. Its significance is due to growing demand for interactive applications of virtual reality. Time required to prepare such applications depends heavily on availability of reusable 3D objects with embedded behavior. Effective search for interactive 3D objects needed for new applications requires object metadata that cover not only object geometry and semantics, but also behavior.

Existing metadata standards define different information describing an object. General metadata standards, like for example Dublin Core or XMP, allow storing information about object title, creator, or its creation date. Specific standards, like MPEG-7 or MXF DMS-1, allow storing technical and semantic information about an object. The missing metadata fragment, not included in existing metadata standards, is metadata of object behavior. Such information should be stored in a format which enables efficient search for objects with specific interaction properties.

In this dissertation, a new approach to describe interactive 3D object behavior is presented. In this approach object behavior is described by models of potential object interactions with other objects. The proposed approach is based on two main elements: a model of interactive object interactions and a concept of interaction interface. Based on these two elements a special query sub-language was implemented. The sub-language enables efficient usage of interaction metadata.

Model of 3D object interactions, called *Multimedia Interaction Model* – MIM, allows decomposition of an interaction of any object into components. These components are described by distinct metadata structures tailored for a particular component. Structure of interaction metadata allows using textual semantic descriptions and formal mathematical descriptions. Semantic descriptions can be used on different levels of metadata structure, providing both general and specific descriptions. General descriptions concern the interaction as a whole, while specific descriptions concern particular details of an interaction. Moreover, semantic descriptions can be linked with an ontology specific for a given domain. This possibility increases the information value of interaction metadata and quality of search results. Mathematical descriptions are used for metadata that allow calculating new state of an object after the interaction. The new state can be used as a starting point for next interactions. Therefore, mathematical descriptions enable search engines to run searches for interactive

objects based not only on current object state, but also on object states resulting from an interaction.

The MIM model and interaction metadata based on this model are the basis for a new approach to metadata design. In the proposed approach interactive objects are treated like elements of a computer program, instead of static items. Foundation of such approach is a concept called *Interaction Interface* – II. According to this concept the structure of an interaction interface is defined, as well as specifies rules for defining new interaction interfaces. Interaction interfaces, as a part of interaction metadata, provide detailed semantics of parameters related with an interaction. Extended semantic information contained in the interaction metadata facilitates searches for objects with specific interaction properties.

The approach presented in this dissertation was implemented as metadata schemas in a form of XML Schemas. It was verified by practical application to a number of descriptions of complex interactive 3D objects. Usefulness of such descriptions was analyzed in different search scenarios.

1. Introduction

In the last decades outstanding progress is observed in the domains of electronics, information technology, and the Internet. According to Internet World Stats, the Internet penetration reaches almost 70% in USA and around 40% in Europe. Globally, more than one billion people are regularly using Internet. The reach of the Internet is increasing not only in numbers but also in form. It evolved from slow dial-up connections via fixed lines to broadband and wireless connections. The number of mobile communication devices is also fast increasing. In the most advanced countries the penetration of mobile phones is reaching or even exceeding 100%. With the current trends of evolution of mobile devices, notably the success of Nokia smart phones and Apple iPhone, it can be assumed that soon several billions people around the world will have small, networked computer that is always with them.

Development of communication and information technologies is followed by development of more and more advanced applications of these technologies for professional and common use. These technologies are used in almost any domain, from professional computer aided design systems, simulations, and telemedicine, to education [92], personal and entertainment multimedia solutions [87] like virtual museums [90][95], networked home appliances, on-line multi-user 3D games, and interactive Internet applications. Due to expanding technology capabilities, the new applications can be more powerful and can provide more features. However, usually complex applications with many functionalities are more difficult to use, while the ease of use of new products is a key factor of its market acceptance and therefore its business success.

Product ease of use is affected by product design and employed user interface. All devices deal with the problem of information overload and need to provide a user with a convenient interface to manage this information. A user interface issue is extremely important for small devices, where the ease of use have to compensate limits imposed by device size. The shrinking size of devices and growing information overload motivates investments and progress in the field of user interfaces. In the last years, one could observe how user interfaces of personal computers and communication devices moved from passive two-dimensional interfaces to interactive and multimedia solutions.

In near future user interfaces of most personal devices and computers will try to mimic a real three-dimensional world to provide a user with the most natural experience possible. Some operating systems [63][15] and applications [29] go a step further offering full 3D interfaces.

The use of 3D user interfaces adds a new level of freedom, which allows treating the electronic world as a natural extension of the real world and opens a way for easier information management and immersive 3D applications.

Interactive 3D interfaces and applications [88][89] may be constructed as simple 3D scenes with some objects inside, or as dynamically generated 3D worlds. Efficient creation of both requires the use of interactive 3D objects. An interactive object recognizes a number of external interactions and reacts according to predefined algorithms. A complete set of such algorithms composes object behavior. Behavior of an interactive 3D object is stored within the object itself instead of being programmed externally in an application. Such approach is similar in concept to external, reusable programming libraries and follows the object-oriented paradigm. This approach allows to easy exchange of application components without the need to modify the application code. As a result, time and effort required to build an interactive 3D scene or compose a 3D world decreases.

The benefits of interactive 3D objects increase with the growth of shared, searchable libraries of reusable 3D objects. Simple libraries or collections of 3D objects already exist, but exchange of objects is limited by specific technologies used to develop these objects. The increasing use of 3D on the Internet favors universal and open standards. As a consequence the majority of new 3D objects are developed in one of two technologies that satisfy above requirements, namely X3D (Extensible 3D [96]) with its predecessor VRML (Virtual Reality Modeling Language [86]) and MPEG-4 (a standard developed by Moving Pictures Experts Group [55]). X3D standard is focused on 3D geometry and 3D scene composition, while MPEG-4 comes from telecommunications industry and covers more 3D related areas, including binary file formats and communication protocols. Despite different origins both standards use similar 3D geometry format. It is yet unknown either one of these standards will overtake the whole 3D market, or they will exist in parallel. Nevertheless the openness of the above standards, similarities in 3D geometry representation and already existing automatic conversion tools, suggest that in near future 3D object exchange and reusability will be common.

With the increasing numbers of reusable interactive 3D objects, collections of such objects may become so immense that finding a right object will be very time consuming. Moreover, new interactive 3D products are often targeted at global markets and clients from many different countries and cultures. To properly address different needs of users from various countries, development teams include specialists from all those countries. Such teams are often distributed across the world and they produce and reuse interactive 3D objects in different cultural and language contexts. The resulting multi-national collections are even more difficult to search and locate a specific object. Solution to this problem is to build search engines specialized in finding 3D objects. Prototype of a search engine that uses geometric characteristics of 3D objects has been created by a team at Princeton [27] and is available on the Internet [1].

However, to achieve highly relevant search results a search engine needs to address all components of an interactive 3D object – its geometry [9][96], semantics [28] and behavior. While geometry information can be extracted from the object, its semantics and especially behavior are difficult and sometimes impossible to gather automatically. Therefore, there is a need for metadata. According to ISO/IEC 11179 the metadata is defined as data that describes other data [36]. It is information about the object, distinct from the object itself. There exist many metadata standards tailored for different domains and applications. The most basic use of metadata is for general cataloguing and indexing [20][19][52]. Each domain has its own set of metadata solutions. Cultural heritage domain uses Spectrum [83], CIMI [14] and others [6][53][61]. Movie production industry has MXF [75][76], TV-Anytime [25], and p-meta [24] among others. Also multimedia objects have wide spectrum of available metadata solutions. The most important are Dublin Core [20], XMP [103], EXIF [41], DIG35 [19], NISO Z39.87 [59], MusicBrainz [57], ID3 [32], AAF [2], MXF DMS-1 [76], P/Meta [24], MPEG-7 [56]. All listed solutions are described in Chapter 2. Existing metadata standards are mostly focused on object semantics, though there are also tools for describing object technical characteristics [73][31]. Nevertheless none of existing metadata standards is capable of addressing the behavior of an object. Therefore, there is a need for a metadata solution able to describe not only object semantics but also its behavior. The complete metadata of a 3D interactive object should include tools for:

- describing general semantics and production, technical, and management information;
- describing geometry;
- spatio-temporal decomposition;
- three dimensional decomposition;
- describing semantics of interactions;
- building chains of interactions;
- defining parameters related with interactions;
- extending the list of available types of parameters related with interactions;

Currently there are a few widely used metadata standards useful for describing multimedia objects [74], like: MPEG-7 [56], Material Exchange Format (MXF) Descriptive Metadata Scheme (DMS) 1 [76], Advanced Authoring Format (AAF) [2], or more general Dublin Core Metadata Element Set [20]. However, from the above standards all but one deal only with the first item listed above – general semantics and information. The one more advanced is the MPEG-7 standard. It addresses not only general semantics and information but also geometry and spatio-temporal decomposition. Nevertheless, it was formulated with audio and video contents in mind and its handling of 3D objects is limited. Especially, in terms of interactive 3D objects. In the past few years, some 3D metadata solutions were developed [48][62][9][10]. However, there is no universal and extensible approach capable of describing interactive 3D objects.

In this dissertation an interaction metadata model is presented, called Multimedia Interaction model (MIM). This model is designed to complement existing metadata standards and to cope with descriptions of any interactions of 3D object. In this model two new ideas are introduced: a universal solution for modeling object interactions and a concept of interaction interfaces of 3D objects.

The MIM model is based on the *event – condition – action* concept used in active database systems. The model describes interaction in terms of its source, required context, and the result. Each part of this description, especially the context and the result, can be described not only with semantic tools but also with mathematical expressions that approximate object behavior algorithms. Such approach allows calculating the actual result of an interaction, which creates a new object state. The new object state can be used as a starting point of subsequent interactions. In effect, metadata composed of mathematical expressions can be used to calculate subsequent states of an object, which allows analyzing complex chains of interactions. The interaction model is complemented by Interaction Interface concept, and interaction metadata querying solution. The Interaction Interface concept provides tools for describing object interaction properties in a way similar to describing application programming interfaces (API). This enables search engines to easily match objects compatible in terms of interactivity, just by comparing implemented interaction interface. The Interaction Metadata Query sub-Language (IMQL) allows exploiting metadata descriptions created with the Multimedia Interaction Model and Interaction Interface concepts.

The thesis of this dissertation can be formulated as follows:

Multimedia Interaction Model (MIM) allows search engines to find 3D objects with specific interaction properties.

This dissertation is composed of 10 chapters:

Chapter 2 contains an overview of the current state of the art in the domain of metadata for multimedia and 3D objects. The chapter presents various metadata standards, from general solutions like DublinCore, to standards designed especially for multimedia contents like MPEG-7. Description of each metadata standard focuses on its primary purpose.

Chapter 3 presents state of the art technologies that can be applied for metadata of multimedia 3D interactive objects. This concerns not only metadata standards but also other solutions that can be used for describing 3D objects (3DSEAM) and their interactions (MathML). This chapter contains also descriptions of available metadata querying solutions.

In Chapter 4, first, the limitations of the current metadata solutions are analyzed in detail. Then, the concepts of the interaction model and interaction interfaces are briefly presented.

Chapter 5 defines all basic terms and describes the concept of Multimedia Interaction Model. The overall model overview is followed by descriptions of main model components.

In Chapter 6, the concept of Interaction Interface is described. This chapter contains definition of the Interaction Interface, specifies rules for defining new interaction interfaces, and provides definition of the fundamental Visual Interaction Interface.

Chapter 7 presents detailed information about the structure of Multimedia Interaction Model and Interaction Interface concepts introduced in Chapter 5 and Chapter 6. In this chapter the XML Schema of Multimedia Interaction Model and Interaction Interface are illustrated and described in detail.

Chapter 8 provides information about the Interaction Metadata Query sub-Language (IMQL). The description includes definition and syntax of the proposed querying language and specification of rules for embedding IMQL within existing and more general querying languages.

Chapter 9 contains 5 examples of applications of all proposed solutions. Examples demonstrate the structure of interactions metadata, usage of mathematical expressions, custom interaction interfaces, semantic descriptions linked with an ontology, and complex descriptions of interaction results.

Chapter 10 concludes the dissertation.

Appendices A-C contain XML Schema of the interaction metadata description. This includes Multimedia Interaction Model XML Schema, Interaction Interface Schema and overall Interactions XML Schema. The last one ties the first two into complete description of object interactions.

Appendix D presents an XML document with definition of the Visual Interaction Interface.

Appendix E includes XML documents with full metadata of example objects presented in Chapter 9.

2. Metadata for multimedia objects

2.1. Overview

Based on recent publications [74][30] existing solutions for multimedia metadata can be divided into classes based on their domain and purpose:

- Metadata for general purpose descriptions.
- Metadata for describing still images.
- Metadata for describing audio content.
- Metadata for describing audio-visual content.

The following sections describe existing metadata solutions from the above classes, with an emphasis on solutions applicable for 3D objects.

2.2. Metadata for general purpose descriptions

2.2.1. Dublin Core

The Dublin Core (DC) metadata is a set of fifteen elements designed to foster consensus across disciplines for the discovery-oriented description of diverse resources in an electronic environment. The first element set [94] – Dublin Core Metadata Element Set (DCMES) – was created during Dublin Core workshop [22], held in Dublin, Ohio USA. Originally, the DCMES was created to enhance searching of document-like objects on the web. However, it can be applied to any electronic resources: from graphics, to sound and video files. Due to its general nature and portability, DCMES quickly became widely used metadata standard for a number of digital library solutions around the world. Currently the work on DCMES specification and extensions is lead by Dublin Core Metadata Initiative (DCMI) [21].

The Dublin Core Metadata Element Set [20] has been formally endorsed in two standards: ISO Standard 15836-2003 [38] and NISO Standard Z39.85-2001 [58]. DCMES is composed of fifteen elements referring creation and technical information. The specification defines element URI, name and definition. It does not provide information about structure of an element value, nor about its syntax.

<u>Dublin Core Metadata Element Set reference (version 1.1):</u>

1. Contributor – An entity responsible for making contributions to the resource.

2. Coverage – The spatial or temporal topic of the resource, the spatial applicability of the resource, or the jurisdiction under which the resource is relevant.
3. Creator – An entity primarily responsible for making the resource.
4. Date – A point or period of time associated with an event in the lifecycle of the resource.
5. Description – An account of the resource.
6. Format – The file format, physical medium, or dimensions of the resource.
7. Identifier – An unambiguous reference to the resource within a given context.
8. Language – A language of the resource.
9. Publisher – An entity responsible for making the resource available.
10. Relation – A related resource.
11. Rights – Information about rights held in and over the resource.
12. Source – The resource from which the described resource is derived.
13. Subject – The topic of the resource.
14. Title – A name given to the resource.
15. Type – The nature or genre of the resource.

Since DCMES is intended to support cross-discipline resource discovery, it does not satisfy every possible declarative need in every discipline. Thus, in most applications it is used in a combination with more advanced, domain-specific metadata standard. DC elements are usually a subset of resource metadata and are used as a minimal metadata for data exchange and discovery.

2.2.2. XMP

The XMP metadata specification [103] was created by Adobe systems Incorporated [3]. It was designed to improve management, search and retrieval, reuse, and consumption of multimedia assets. The XMP specification consists of three main components: data model, storage model and formal schema definitions. Data model is derived from RDF and it was designed as a subset of the RDF data model. The data model provides support for metadata properties attached to a resource. Properties have property values, which can be simple or complex (structure or array). Properties also may have properties offering information about the property value. Storage model describes serialization of the metadata to the XML format. Metadata serialized in such way can be attached to various types of digital resources and easily processed across different systems and applications. Schema is a collection of predefined sets of metadata property definitions. There are some predefined schemas like a

Dublin Core Schema, a basic rights schema or a media management schema. However, the collection of schemas is extensible. It is possible to extend existing schemas and to define new ones.

Due to wide adoption of XMP, not only in Adobe products, The International Press and Telecommunications Council (IPTC) [33] has integrated XMP in its Image Metadata specifications [34].

2.3. Metadata for describing still images

2.3.1. EXIF

The Exchangeable Image File Format (Exif) [41] standard specifies the formats to be used for images, sound and tags in digital still cameras and in other systems handling the image and sound files recorded by digital still cameras. It is one of today's commonly used metadata format for digital images.

The Exif metadata covers metadata related to the capturing of the image and the context situation of the capturing. This includes metadata related to the image data structure (e.g., height, width, orientation, etc.), capturing information (e.g., rotation, exposure time, flash), recording offset (e.g., image data location, bytes per compressed strip, etc.), image data characteristics (e.g., transfer function, color space transformation, etc.), as well as general tags (e.g., image title, copyright holder, manufacturer, etc.), or even GPS information.

2.3.2. DIG35

The DIG35 specification [19] goal is „To provide a mechanism to allow end-users to see digital image use as being equally as easy, as convenient, and as flexible as the traditional photographic methods while allowing benefits that are possible only with the digital format." To reach this goal DIG35 metadata was designed to "improve semantic interoperability of metadata between various digital imaging devices".

Metadata elements included in the DIG35 specification cover following topics:

- Basic Image Parameter – a general-purpose metadata;
- Image Creation – mostly technical camera characteristics;
- Content Description – semantic description of the image;
- History – records steps of image processing;
- Intellectual Property Rights;

- Fundamental Metadata Types and Fields – definition of metadata types and fields used in all other components.

The DIG35specification defines also metadata encoding with XML Schema.

2.3.3. NISO Z39.87

NISO Z39.87 [59] is a set of metadata elements designed to facilitate interoperability. It enables users to develop, exchange and interpret raster digital image files in a way that supports image management and creation of digital image collections. The NISO Z39.87 dictionary applies only to raster images created thorough digital photography or scanning, and altered through editing or image transformation. It is not intended to be used with other image formats such as animated bitmap, vector, or motion picture.

Elements of the NISO Z39.87 metadata set cover wide range of topics related with large-scale digital repositories and digital asset management systems. This includes: basic image parameters, image creation, imaging performance assessment and image history.

2.4. Metadata for describing audio content

2.4.1. MusicBrainz Metadata Initiative

MusicBrainz project [84][57] is a project of the US non-profit MetaBrainz Foundation [50]. The project aims to create large database of music metadata and facilitate storing and exchanging information about digital audio tracks. Part of the project was to develop a metadata specification – MusicBrainz Metadata Initiative. They chose RDF/XML as a base for their metadata solution and defined three namespaces, each defining RDF classes and properties to cover different areas of MusicBrainz audio metadata. First namespace includes classes and properties for expressing basic music related metadata, as artist, album, track, etc. Second namespace covers metadata lookups between an RDF enabled client and a MusicBrainz metadata server. Third namespace, called MusicBrainz Extended Metadata Namespace, is reserved for future use and will focus on the more detailed metadata such as contributors, roles, lyrics, release dates, remix/cover information, etc.

2.4.2. ID3

The ID3 metadata specification [32] is similar in nature to the basic namespace of MusicBrainz Metadata Initiative. It defines a set of elements that allows to state information about the title, artist, album, etc. of the music file. It is encoded as a container in the MP3 audio file format.

2.5. Metadata for describing audio-visual content

2.5.1. AAF

The Advanced Authoring Format (AAF) [2] is a cross-platform file format of multimedia resources that supports encapsulation of metadata. AAF is an effort of Advanced Media Workflow Association (AMWA) (formerly AAF Association) [4]. Its main goals are to provide description of authoring information and to allow the interchange of data between multimedia authoring tools. AAF is designed to support motion picture post-production/authoring environments. The object-oriented AAF model allows for extensive timeline-based modeling of motion picture montages, including transitions between clips and the application of effects (e.g., dissolves, wipes, flipping). Apart from describing characteristics and transitions of a video clip, AAF also allows to track the entire derivation chain of a clip, from its current state to the original storage medium.

The AAF object model is extensible. Extensions can be defined as subclasses of existing metadata objects.

2.5.2. MXF DMS-1

The Material Exchange Format (MXF) [75] is another file format intended to encapsulate audio-visual data and their descriptions. The MXF is developed by Society of Motion Picture and Television Engineers (SMPTE) [77]. MXF operates in the same domain as AAF and thus it was designed to complement the AAF. The MXF format does not include layering and transition information and uses streamable encoding. This makes it favorable for creation and distribution, while AAF is used in post-production. To facilitate format interchange, all areas where these two formats overlap are technologically identical.

The MXF format contains three types of metadata: structural, ‘dark’, and descriptive metadata. Structural metadata is derived from AAF data model and describes technical characteristics and relations of the audio-visual clips carried by an MXF file. Dark metadata is the term given to metadata that are unknown by an application. They may contain vendor specific or private elements that are meant to be read only by specific applications and are ignored by other systems. Descriptive metadata is a set of semantic descriptions about the data. Among others it comprises information about the production (e.g. rights, broadcaster and delivery type), the clip (e.g. which type of camera was used) or a scene (e.g. scene director and actors). DMS-1 (Descriptive Metadata Scheme 1) [76] is an attempt to standardize such information within the MXF format. DMS-1 enables interoperability of MXF metadata with different metadata schemes such as MPEG-7, P/meta and Dublin Core.

2.5.3. P/Meta

The P/Meta Metadata Scheme [24] is an effort of European Broadcasting Union (EBU) [23] aiming at developing common metadata exchange framework for broadcasters, publishers and archives. P/Meta is not an exchange file format and it is not concerned about the transfer and storage of the metadata and the media itself.

P/Meta is a flat list of attributes complete with semantic descriptions. Attributes are grouped in transaction sets related to different workflow processes. One of P/Meta objectives is to maintain language independence. For this reason it defines code values for appropriate attributes and syntax and notation for set construction. Attributes and sets included in the P/Meta Metadata Scheme cover the areas of logical content and meaning of information, and technicalities of coding and protocols. This includes:

- support for identification and recognition of multimedia material;
- editorial and descriptive information;
- rights to use multimedia material;
- information on how properly open and/or play back the multimedia material.

While MXF format is intended to allow the interchange of finished material. AAF role is to be an authoring format, capable of supporting full storage requirements. The P/Meta has a broader scope of metadata exchange between content producer, distributor and archive. It supports all business transactions concerning multimedia material – even those required before the material exists. Additionally the P/Meta Metadata Scheme permits to include of new metadata attributes and sets.

2.5.4. MPEG-7

The MPEG-7 [56] is an ISO standard [39] developed by Moving Picture Experts Group (MPEG) [54], which is a working group of the Joint Technical Committee for Information Technology (ISO/IEC JTC 1). The MPEG-7 standard, formally named “Multimedia Content Description Interface” is a little bit different than all metadata solutions presented above. It does not target a particular type of multimedia. It is intended for describing any multimedia content. MPEG-7 standardizes tools that are used to create metadata of multimedia content. These tools include: descriptors, description schemes, a Description Definition Language (DDL) and a number of Systems tools.

Descriptor is a basic element of the MPEG-7 metadata. It is used to represent specific features of the content. Generally descriptors represent low-level features such as visual (e.g. texture, camera motion) or audio (e.g. melody). Descriptor specification defines the syntax and semantics of the feature representation. Examples: a time code for representing duration of a movie, color histogram for representing a color of an image and a character string for representing general purpose information like title.

Description scheme is a structural component of MPEG-7. It defines the structure and semantics of the relationships between its components, which may be both descriptors and description schemes. With description schemes it is possible to describe multimedia content structure and semantics. For example, it is possible to describe a movie, temporally divided into shots, including semantic textual descriptors at the movie level and technical color, motion and audio amplitude descriptors at the shot level.

Both description tools presented above are represented using Description Definition Language (DDL). DDL enables the creation of new description schemes and descriptors, It permits to extend and modify existing description tools. MPEG-7 DLL is based on W3C XML Schema [101] recommendation. It uses all XML Schema tools and constructs, and adds a few more, like array or matrix datatypes.

Systems tools are not related directly to metadata descriptions. They cover the areas of storage and transfer of descriptions, as well as issues related with intellectual property rights.

3DSEAM

The MPEG-7 standard was designed for any multimedia contents and with its vast base of descriptors and description schemes, is largely accepted as a solution for the characterization of audio, image or video objects. However, it is not sufficient for 3D multimedia objects.

To cope with this problem a group from Laboratoire Informatique de Grenoble (LIG), lead by Ioan Marius Bilasco, proposed a MPEG-7 extension called 3D SEmantics Annotation Model (3DSEAM) [9][10]. The main assumption done by LIG group is that MPEG-7 description tools are sufficient and it is only the localization descriptors that need to be extended. The standard localization descriptors provided by MPEG-7 include the Media Locator and the

Region Locator. Media Locator is a base for any localization tools that refer to the media file or its fragments (e.g. shots in a movie). Alas, it allows only for a single URI, which is not sufficient for 3D objects scattered through many files. To overcome this limitation 3DSEAM defines an extension of Media Locator – Structural Locator. Region Locator tool is a base of descriptors used to represent a specific region within a multimedia content (e.g. a see on a picture of a tropical beach). Although the Region Locator provides some geometric localization, the 3DSEAM extends it to new tool called 3D Region Locator. This new tool provides 3D specific localization: 3D surfaces and volumes.

The extensions proposed in 3DSEAM complement the MPEG-7 standard and make MPEG-7 the best available solution for metadata of 3D multimedia objects.

3. Interaction descriptions

3.1. Overview

In the 3D virtual environments interactions can be classified as: human to human, human to object and object to object interactions. The first type – human to human interactions – concern new 3D communication solutions that allow for communication between two or more persons conducted via the 3D virtual environment. The area of human to object interactions refer to 3D design and simulation systems, where actions taken by humans on 3D objects are followed by pre-programmed reactions of those objects. The last type of interactions – object to object interactions – is related mostly to 3D virtual worlds, where objects have behavior defined and can act independently of human interventions. This dissertation focuses on metadata for describing the last type of interactions.

Research on autonomous 3D objects with embedded behavior and object to object interactions, is recent and fast growing. The metadata related with this domain: interaction metadata, constitute a particularly new topic. Therefore, at present there are no solutions that cover all aspects of interactions metadata. This section describes existing solutions that can be used to construct interactions metadata.

Object to object interactions are an effect of execution of object behavior, which in most cases is predefined in a form of computer program written in some programming language. There is no easy way of extracting a behavior algorithm for the executable code or even from the source code, especially object oriented code. Thus, to enable searching of objects based on their interaction attributes, there is a need for interaction metadata descriptions eligible for automatic computer processing and query languages capable of addressing such interaction metadata. Since 3D object behavior is a computer program, metadata description of object behavior and resulting interactions, can be constructed in two ways: as a semantic description of the result of behavior execution (i.e. an interaction) and as an algorithm representing the logic of the computer program. Currently there are no universal metadata solutions that cover both forms of describing interactions. Nevertheless, if taken separately, some state-of-the-art solutions can be applied. The semantic description can be covered by MPEG-7 text annotations that allow for various forms of annotations, while the algorithmic description can be written using mathematical markup notation of MathML. Both listed solutions are described in Sections 3.2 and 3.3, respectively.

The interaction metadata describes interactions of 3D objects in some 3D space. Therefore, some 3D geometrical properties and relations have to be described as well. Some of 3D properties of the interaction can be expressed using 3D locators from 3DSEAM. However, the set of description tools available in 3DSEAM is limited to basic geometrical shapes. In some cases, interaction description has to describe a viewpoint of the object. For example: the interaction occurs when the object 'sees' another object. For such interactions 3DSEAM 3D locator tools are not sufficient. Such functionality can be covered by Extensible 3D standard (X3D). It is not a metadata standard, but some of its elements and concepts can be applied to descriptions of 3D properties of an interaction. The X3D standard is described in Section 3.4.

There are no metadata query languages designed with interaction metadata in mind. Thus, to be able to use interaction metadata in a query, the query languages have to rely on metadata syntax, not its semantics. Most of the metadata in today's world are written in XML format [26]. All three: MPEG-7, MathML and X3D have XML representations. Therefore, the most obvious choice for metadata query language is XQuery [105]. Query language designed for XML documents. Another common data format used for metadata descriptions is RDF [68]. The RDF is often used to represent semantic descriptions, knowledge about an object or a feature. RDF documents are also written in XML, so the XQuery language can be used also in this case. However, there is better, more specialized language – SPARQL [79]. Apart from the XML format, metadata are often stored in relational databases. In some solutions even XML based metadata are kept in relations in a database [91]. Hence, also the universal SQL [35] language can be used for querying metadata, including interaction metadata. The first two metadata query languages mentioned above will be described in more detail in Section 3.5.

3.2. MPEG-7 text annotations

Apart from being the best known metadata standard for multimedia objects, the MPEG-7 standard provides a number of tools for creating semantic descriptions [71][49]. Most o them are used to describe semantic entities of the multimedia content and semantic relations of these entities. In case of an interaction of a 3D multimedia object the description can be implemented as a simple or complex textual annotation. The MPEG-7 provides four solutions for text annotations: free text, keyword, structured and dependency structure annotations.

Free text annotation is a text written in a natural language. It can be any language or the same text can have many language versions.

Example 3-1

Consider a simple interaction: "A green car moving towards an intersection. A traffic light changes to red and the car brakes abruptly." In the presented scene the car is interacting with a traffic light. The free text annotation of such interaction looks as follows:

```
<TextAnnotation>
        <FreeTextAnnotation xml:lang="en">
                A green car brakes on the red light just before an intersection.
        </FreeTextAnnotation>
</TextAnnotation>
```

Listing 3-1. MPEG-7 TextAnnotation example

Keyword annotation is a list of keywords that represent main elements of the described scene. This type of annotation makes a step towards computer processing. The free text annotation has to be analyzed with a help of natural language processing (NLP) techniques which can be difficult and time consuming, while keywords from keyword annotation can be used almost directly. On the other hand, a consequence of such simplicity is lost of the structure of description. Like in the text annotations, also here keywords can be written in different languages. However, there is only a very limited way of relating same keywords written in different languages.

Example 3-2

Consider Example 3-1. Keyword annotation of the same interaction is the following:

```
<TextAnnotation>
        <KeywordAnnotation>
                <Keyword>car</Keyword>
                <Keyword>brakes</Keyword>
                <Keyword>red traffic light</Keyword>
                <Keyword>intersection</Keyword>
        </KeywordAnnotation>
</TextAnnotation>
```

Listing 3-2. MPEG-7 KeywordAnnotation example

Structured annotation is an extension of the concept of keywords. It is intended to maintain the simplicity of analyzing keywords, while keeping the basic semantic structure of the description. In structured annotation, keywords are treated as answers to simple questions: *Who*, *WhatObject*, *WhatAction*, *Where*, *Why* and *How*. The answer to each question is represented by a single keyword – single annotation element. These annotation elements are not simple text elements. Instead of MPEG-7 *Textual* type, the MPEG-7 *TermUse* type is used here. The *TermUse* type enables the annotation element to be either a free text description (a keyword) or a dictionary-based term.

Dictionary terms are preferred, since they allow more precise annotations. A dictionary, in MPEG-7 called 'classification scheme', is defined by different qualifiers, which allow associating a dictionary with a set of genres and a given semantic concept (*Who*, *Where*, etc). A hierarchical relationship between dictionaries enables automatic establishment of parent-child relationships between dictionary entries. The whole dictionary structure could be considered as the starting point to further develop an ontology.

Example 3-3

Consider Example 3-1. Structured annotation of the same interaction is the following:

```
<TextAnnotation>
        <StructuredAnnotation>
                <WhatObject href="urn:examples:vehicles:car" />
                <WhatAction href="urn:examples:vehicleActions:deaccelerate">
                        <Name>brake</Name>
                </WhatAction>
                <Why><Name>red traffic light</Name></Keyword>
                <Where><Name>intersection</Name></Keyword>
        </StructuredAnnotation>
</TextAnnotation>
```

Listing 3-3. MPEG-7 StructuredAnnotation example

The above description presents three solutions for representing an annotation element: term reference, inline term and free term. The first element *car* is represented by an URI of a particular term in a predefined dictionary. The second element *brake* uses inline term representation to include both, the relation to a dictionary and the term inline definition. Third and fourth elements use simple free term representation and just define a new term without linking it with any existing dictionary.

Dependency structure annotation is the most powerful from text annotations available in the MPEG-7 standard. It enables creation of descriptions that represent linguistic structure of an annotation. Dependency structure annotation concept is based on a linguistic theory called 'dependency grammar' [17][18][43][44], which explains a grammatical structure of a sentence in terms of dependencies between its elements. The structure is represented as a tree with one root element and a set of leafs, where each leaf can be a subtree organized in the same way as the main tree. Such tree like structure of an annotation is represented by nested

XML elements. Types and attributes of these elements represent the linguistic relations. Like in the structured annotation, elements of the annotation may reference a dictionary, which enables the use of an ontology.

Example 3-4

Consider Example 3-1. Dependency structure annotation of the same interaction is the following:

```
<TextAnnotation>
        <DependencyStructure>
                <Sentence>
                        <Phrase operator=”subject”>
                                <Head type=”noun”>car</Head>
                                <Phrase>
                                        <Head type=”adjective”>green</Head>
                                </Phrase>
                        </Phrase>
                        <Head type=”verb” baseForm=”break”>breaks</Head>
                        <Phrase>
                                <Head type=”adverb”>abruptly</Head>
                        </Phrase>
                        <Phrase>
                                <Head type=”preposition”>before</Head>
                                <Phrase>
                                        <Head type=”noun”>intersection</Head>
                                </Phrase>
                        </Phrase>
                </Sentence>
        </DependencyStructure>
</TextAnnotation>
```

Listing 3-4. MPEG-7 DependencyStructure example

Using dependency structure annotations it is possible to query annotations using its syntactic structure. For example, it is possible to search for interactions in which the subject was *car* and the verb was *brake*. Such query permits to find all interactions where a car brakes for whatever reason, which could be difficult using only unstructured text matching.

3.3. MathML

The MathML [51][7][72] is an XML based markup language capable of encoding the structure of mathematical expressions, so that they can be rendered in a browser, manipulated in an editor, and evaluated in a computer algebra system. Handling proper display and meaning of a mathematical expression requires a little bit different approaches. The MathML language provides separate notations for presentation and content encoding.

The presentation notation has 30 elements with about 50 attributes. Most elements represent templates or patterns for laying out subexpressions. The MathML presentation notation has the same application as for example mathematical LaTeX commands [42][45][46][47]. Those commands are used to describe a way mathematical expressions are displayed.

The most important presentation elements are *mi*, *mn* and *mo* for representing identifiers, numbers and operators respectively. Other elements include for example *mfrac*, which is used for forming a fraction from two expressions by putting one over the other with a line in between, or *mfenced* element used to display parentheses surrounding contents of the element.

Example 3-5

Consider a simple mathematical expression $(a + b)^2$. The presentation notation of this expression is the following:

```
<mrow>
  <msup>
    <mfenced>
      <mrow>
        <mi>a</mi>
        <mo>+</mo>
        <mi>b</mi>
      </mrow>
    </mfenced>
    <mn>2</mn>
  </msup>
</mrow>
```

Listing 3-5. MathML presentation notation example

The top-level element: *mrow*, specifies that its content is a single row expression. The first subelement: *msup*, is a superscript element. It has two arguments: the base and the superscript. The base of the superscript element is a *mfenced* element which will render parentheses around an expression contained in another *mrow* element. The expression inside parentheses is composed of three elements: two identifiers *a* and *b* and an operator +.

Using presentation elements it is possible to precisely control how an expression will look when displayed in a browser, or printed on paper. However, it is possible that properly presented mathematical expression cannot be evaluated in a computer algebra system. To avoid any misinterpretations possible with the presentation notation MathML provides the content notation.

The content notation is focused on expressing the meaning of the mathematical expression. The content notation is intended for automatic processing and evaluation, where presentation nuances are not important. The content notation does not follow infix mathematical notation used for display or print. Instead, similar to LISP computer language, it uses prefix notation. The prefix notation corresponds to natural language constructions like “f of x” and “subtract 5 from 8”. A side effect of using the prefix notation is that parentheses are no longer necessary, which makes computer processing and evaluation even easier.

The content notation has around 100 elements, with more than 10 attributes. Most of these elements represent mathematical operations and functions and are grouped in classes like: n-ary arithmetic (e.g. *plus*, *times*), unary elementary classical functions (e.g. *sin*, *log*). Other elements represent mathematical containers like *set* and *vector* or variables (*ci*), numbers (*cn*) or external symbols (*csymbol*).

Example 3-6

Consider the expression provided in Example 3-5. The content notation of this expression is the following:

```
<mrow>
  <apply>
    <power/>
    <apply>
      <plus/>
      <ci>a</ci>
      <ci>b</ci>
    </apply>
    <cn>2</cn>
  </apply>
 </mrow>
```

Listing 3-6. MathML content notation example

The top-level element: *mrow*, specifies that its content is a single row expression. The *apply* content element means apply an operation to an expression. In this case, the *power* element is applied to a subexpression. The second *apply* element applies *plus* operation to *a* and *b* variables.

The order of child elements is crucial. The *apply* element requires the first element to be an operator and interprets the remaining children as the arguments of that operator. Also some operators depend on the order of arguments, like the *power* operator in the above example.

The MathML content notation is not intended to systematically codify all of mathematics. Instead, MathML creators carefully chose a relatively small number of commonplace mathematical constructs that should be sufficient in large number of applications. The base set of content elements includes:

- arithmetic, algebra, logic and relations;
- calculus and vector calculus;
- set theory;
- sequences and series;
- elementary classical functions;
- statistics;
- linear algebra.

Set of elements in the above areas is not complete. However, MathML provides a mechanism for associating semantics with new notational constructs. This allows defining and encoding mathematical concepts that are not in the base collection.

3.4. Extensible 3D

Extensible 3D (X3D) [96] is an ISO open standard [40] for 3D content integrated with multimedia. It is a universal interchange format for integrated 3D graphics and multimedia. X3D was developed by the Web3D Consortium [93] as a successor of the Virtual Reality Modeling Language (VRML) [86], the original ISO standard for web-based 3D graphics (ISO/IEC 14772) [37]. X3D has a rich set of features to support various applications:

- 3D graphics – Polygonal geometry, parametric geometry, hierarchical transformations, lighting, materials and multi-pass/multi-stage texture mapping;
- 2D graphics – Text, 2D vector and planar shapes displayed within the 3D transformation hierarchy;
- Animation – Timers and interpolators to drive continuous animations; humanoid animation and morphing;
- Spatialized audio and video – Audiovisual sources mapped onto geometry in the scene;
- User interaction – Mouse-based picking and dragging; keyboard input;
- Navigation – Cameras; user movement within the 3D scene; collision, proximity and visibility detection;
- User-defined objects – Ability to extend built-in browser functionality by creating user-defined data types;

- Scripting – Ability to dynamically change the scene via programming and scripting languages;
- Networking – Ability to compose a single X3D scene out of assets located on a network; hyperlinking of objects to other scenes or assets located on the World Wide Web;
- Physical simulation – Humanoid animation; geospatial datasets; integration with Distributed Interactive Simulation (DIS) protocols.

From the wide set of features provided by X3D, interaction description needs only three nodes: Geometry, Viewpoint and Transform. The *Geometry* node is a base node for all X3D geometry definition tools, like: primitives (Box, Sphere), indexed face set, and others. The Geometry node provides a functionality of defining arbitrary 3D volumes, which are particularly useful for interaction descriptions.

Example 3-7

Consider a simple 3D cone. The X3D representation of this geometry is the following:

```
<X3D profile="Immersive" version="3.1">
        <Scene>
                <Shape>
                        <Cone height="5" bottomRadius="2.5"/>
                        <Appearance>
                                <Material/>
                        </Appearance>
                </Shape>
        </Scene>
</X3D>
```

Listing 3-7. X3D geometry representation example

Default rendering of the above X3D code is the following. A cone centered in the local coordinate system, whose central axis is aligned with the local Y-axis. The cone has height of 5.0 and radius of the cone base of 2.5. The cone center is located on the Y-axis in a half of cone height.

The *Viewpoint* node in X3D is used to define a specific location in the local coordinate system from which the user may view the scene. Among others, it contains *position*, *fieldOfView* and *orientation* fields. In interaction descriptions the Viewpoint node can be used to define viewpoints of the object, not the user.

Example 3-8

Consider a 3D viewpoint with the following X3D representation:

```
<X3D profile="Immersive" version="3.1">
        <Scene>
                <Viewpoint fieldOfView="0.785" orientation="0 0 1 0" position="0 2.5 8.1"/>
                ...
        </Scene>
</X3D>
```

Listing 3-8. X3D Viewpoint node example

A viewpoint can be illustrated as a view of a camera positioned in a 3D space. The above X3D code represents a viewpoint of a camera placed in the local coordinate system at 3D point {0, 2.5, 8.1}. The camera is directed down the −Z-axis toward the origin with +X to the right and +Y straight up. The field of view of the camera is 0.785 radians.

The last node, *Transform* node, is a tool for transforming coordinate system of the Geometry node. In X3D, geometry primitives, like Box or Cone, are always centered in the local coordinate system and aligned with the local coordinate axes. The Transform node contains translation and rotation fields, which provide a way to move and orientate geometry elements in space.

Example 3-9

Consider a cone presented in Example 3-7. The Transform element that moves and rotates the cone is the following:

```
<X3D profile="Immersive" version="3.1">
        <Scene>
                <Transform rotation="1 0 0 3.14" translation="0 2.5 0">
                        <Shape>
                                <Cone height="5" bottomRadius="2.5"/>
                                <Appearance>
                                        <Material/>
                                </Appearance>
                        </Shape>
                </Transform>
        </Scene>
</X3D>
```

Listing 3-9. X3D Transform node example

The transformation applied to the cone object is a cone moved up along Y-axis by 2.5 and rotated by 3.14 radians along the X-axis. The effect is a cone turned upside-down with its apex located at the origin of the local coordinate system.

3.5. Metadata query languages

3.5.1. XQuery

Increasing amount of information stored, exchanged and presented in the XML format caused a need for a solution to efficiently query XML data sources. To fulfill this need, after almost ten years of research, the W3C XML Query Working Group [100] formulated XQuery language [105].

The XQuery had its origins in a Query Languages Workshop held by W3C in 1998, in Boston, USA [64]. During this workshop, representatives of a number of organizations ranging from industry, academia to research community, started a discussion on the features and requirements for an XML query language. The 66 presentations came mainly from the members of two distinct constituencies: those working primarily in the domain of XML 'as--document', and those working with XML 'as-data'. The first reflected XML's original roots in SGML, the latter related to increasing presence of XML in the middleware domain. Both points of view are represented in the W3C Query Language Working Group. In effect, the resulting XML query language is representing needs and perspectives of both communities.

The works organized in W3C Query Language Working Group focused initially on Quilt [13]. An XML query language based on XPath [97], XQL [104], XML-QL [98], SQL [35] and OQL [5]. The Quilt was built from the collaborative efforts of the working group in defining requirements, use cases and an underlying data model and algebra [107]. The Quilt was a foundation for XQuery formulated in mid-2001. The main requirements of XQuery [99] include:

- More than one syntax binding.
- Convenient for humans for read and write.
- Declarative. Do not enforce a particular evaluation strategy.
- Independence of any protocols with which it is used.
- Ability to add update capabilities in future versions.

All these requirements have been met and recently the XQuery specification has got the status of official W3C Recommendation [105].

The XQuery language has three clearly distinguishable forms. First, in a formal algebraic language [108] and the data model [107], which described the inner workings of an XQuery processor. Second, it is an XML-based syntax, called XQueryX [102]. It allows articulating

XQuery expressions with syntax more appropriate for machine processing. The third form is the 'surface' syntax, which is a user-friendly and the most commonly used XQuery notation. The term 'XQuery' is used as the name of the 'surface' syntax.

The XQuery is for XML like SQL for relational databases. It allows extracting and manipulating data from XML documents or any data source that can be viewed as being in the XML format. XQuery is based on a tree-structured model of the information content of an XML document, containing seven kinds of nodes: document nodes, elements, attributes, text nodes, comments, processing instructions, and namespaces. Simple data types used in XQuery are the same as in XML Schema. Basic syntactic rules of XQuery are the following:

- XQuery is case-sensitive.
- Elements, attributes and variables must be valid XML names.
- A string value must be in single or double quotes.
- Variable is defined with a *$* followed by a name, e.g. *$library*
- Comments are delimited by *(:* and *:)*, e.g. *(: XQuery comment :)*

XQuery uses XPath expression syntax to address specific parts of an XML document. It allows also conditional expressions used often in scripting languages: 'if-then-else'. The XQuery can be structured similar to an SQL query, with the use of FLWOR expression [106]. A FLWOR expression supports iteration and binding of variables to intermediate results. It can be used for computing joins between to or more XML documents or for restructuring data. The FLWOR expression is constructed from five clauses: FOR, LET, WHERE, ORDER BY, and RETURN. The FOR and LET clauses in a FLWOR expression generate an ordered sequence of tuples of bound variables, called the tuple stream. The optional WHERE clause serves to filter the tuple stream, retaining some tuples and discarding others. The optional ORDER BY clause can be used to reorder the tuple stream. The RETURN clause constructs the result of the FLWOR expression. The RETURN clause is evaluated once for every tuple in the tuple stream, after filtering by the WHERE clause, using the variable bindings in the respective tuples. The result of the FLWOR expression is an ordered sequence containing the results of these evaluations, concatenated as if by the comma operator. Additionally to the comma separated results, XQuery provides features allowing new XML documents to be constructed. The resulting XML elements known in advance can be mixed with the XQuery expression forming an XML structure for the results. When, the returned XML nodes are based on the results, XQuery dynamic node constructors can be used.

Example 3-10

Consider an XML document which describes books available in a bookstore. Each book is described by its title, publisher, authors, publication year and price.

```
<bookstore>
 <book year="2002">
        <title>Introduction to MPEG 7: Multimedia Content Description Language</title>
        <author><last>Manjunath</last><first>B. S.</first></author>
        <author><last>Salembier</last><first>Philippe</first></author>
        <author><last>Sikora</last><first>Thomas</first></author>
        <publisher>Wiley</publisher>
        <price>150.00</price>
 </book>
 <book year="2007">
        <title>XQuery</title>
        <author><last>Walmsley</last><first>Priscilla</first></author>
        <publisher>O'Reilly Media, Inc.</publisher>
        <price>49.99</price>
 </book>
 <book year="2006">
        <title>MPEG-7 Audio and Beyond: Audio Content Indexing and Retrieval</title>
        <author><last>Kim</last><first>Hyoung-Gook</first></author>
        <author><last>Moreau</last><first>Nicolas</first></author>
        <author><last>Sikora</last><first>Thomas</first></author>
        <publisher>Wiley</publisher>
        <price>130.00</price>
 </book>
</bookstore>
```

Listing 3-10. Sample XML document

The above document is queried for all books co-authored by Thomas Sikora. The list of books should be returned as HTML unordered list. Such query written with the XQuery language is the following:

```
<ul>
{
let $store := doc("bookstore.xml")/bookstore
for $book in $store/book
let $authors := $book/author
where $authors/last = "Sikora"
return
        <li>
        {
                $book/title/text()
        }
        </li>
}
</ul>
```

Listing 3-11. XQuery query example

The first LET clause binds the root *bookstore* element to *$store* variable. The FOR clause iterates through the *$store* and retrieves each *book* element. The *book* element is bound to the *$book* variable used in second LET clause, which binds all book *author* elements to the *$authors* variable. The WHERE clause verifies if the last name of any book author is "Sikora". If so the *$book* is passed to the RETURN clause, which prints the book title enclosed in <li> – HTML list item tags. The resulting XML is the following:

```
<ul>
	<li>Introduction to MPEG 7: Multimedia Content Description Language</li>
	<li>MPEG-7 Audio and Beyond: Audio Content Indexing and Retrieval</li>
</ul>
```

Listing 3-12. XQuery results example

3.5.2. SPARQL

Another form of metadata representation is RDF [68]. RDF is a directed, labeled graph data format for representing information. In the metadata field it is used for representing semantic knowledge about an object or about a feature. The query language designed for querying RDF documents is SPARQL [79]. SPARQL name is a recursive acronym that stands for SPARQL Protocol and RDF Query Language. The SPARQL language is developed by W3C RDF Data Access Working Group [66]. SPARQL is based on previous RDF query languages as rdfDB[65], RDQL [67], and SeRQL [11]. The SPARQL was released as W3C Candidate Recommendation in June 2007.

The SPARQL is both, a protocol and a query language. It consists of three separate specifications: first is the data access protocol [78] which uses WSDL to define simple HTTP and SOAP interfaces for remotely querying RDF sources. The responses of SPARQL queries are generated according to second specification: query results XML data format [82]. The third specification is the query language specification, which makes up the core of SPARQL.

The SPARQL is a syntactically very similar to the SQL query language. Its main structure is built with SELECT, FROM and WHERE clauses that work in the same way as in SQL. Additionally, SPARQL provides a way to bind URIs with labels, which is used to simplify the query and improve readability. This mechanism works like namespaces in XML. SPARQL queries an RDF graph via pattern matching. It includes conjunctive patterns, value filters, optional parameters, and pattern disjunction. RDF is built on the triple, a 3-tuple consisting of subject, predicate, and object. Likewise SPARQL is built on the 'triple pattern', which also consists of a subject, predicate and object. The triple pattern is used in WHERE clause to formulate a pattern that is matched against the triples in the RDF graph. SPARQL provides a number of abbreviations for writing complex triple patterns. Both basic syntax and abbreviations are based mostly on the Turtle [8], an RDF serialization alternative to RDF/XML. Triple patterns are like RDF triples, but it can include variables. Any of the subject, predicate, and object values in a triple pattern may be replaced by a variable. Variables are used to indicate data items of interest that will be returned by a query. The WHERE clause may contain multiple triple patterns, combining them gives a basic 'graph pattern', where an exact match to a graph is needed. The result of SPARQL query is a sequence of results that, conceptually, form a table or a result set. Each row of the table corresponds to one query solution. And each column corresponds to a variable declared in the SELECT clause. The results can be returned also in XML format according to Query Results XML Format.

Example 3-11

Consider bookstore inventory description presented in Example 3-10. The same data can be expressed in RDF. Listing 3-13 presents a slightly modified RDF description of bookstore inventory.

```
@prefix     dc:          <http://purl.org/dc/elements/1.1/> .
@prefix     :            <http://mybookstore.com/book/> .

:b1         dc:title     "Introduction to MPEG 7: Multimedia Content Description Language" .
:b1         dc:creator   "Thomas Sikora" .
:b1         dc:publisher "Wiley" .
:b1         dc:date      "2002" .

:b2         dc:title     "XQuery" .
:b2         dc:creator   "Priscilla Walmsley" .
:b2         dc:publisher "O'Reilly Media, Inc." .
:b2         dc:date      "2007" .

:b3         dc:title     "MPEG-7 Audio and Beyond: Audio Content Indexing and Retrieval" .
:b3         dc:creator   "Thomas Sikora" .
:b3         dc:publisher "Wiley" .
:b3         dc:date      "2006" .
```

Listing 3-13. Sample RDF document

The SPARQL query for retrieving titles of books written by "Thomas Sikora" is the following:

```
PREFIX      dc:     <http://purl.org/dc/elements/1.1/>
SELECT      ?title ?year
```

```
FROM        <http://mybookstore.com/inventory.rdf>
WHERE       {
        ?x      dc:creator      “Thomas Sikora”
        ?x      dc:title        ?title
        ?x      dc:date         ?year
}
```

Listing 3-14. SPARQL query example

The first line of the query above associates a *dc:* prefix with the elements definition. The *SELECT* clause specifies that the query should return book title and publication year. Third query line contains the *FROM* clause which indicates the RDF document to be evaluated. Finally the *WHERE* clause contains triple patterns that filter the RDF data and associate data fields with appropriate query variables. The filtering is done by the first triple pattern. It matches all elements whose *creator* property is equal “Thomas Sikora”. The other two triple patterns match all RDF triples and therefore act only as variable bindings.

The result of the above SPARQL query is the following result set:

title	year
Introduction to MPEG 7: Multimedia Content Description Language	2002
MPEG-7 Audio and Beyond: Audio Content Indexing and Retrieval	2006

Table 3-1. SPARQL results example

The XML format for the above result is the following:

```
<sparql
        xmlns="http://www.w3.org/2005/sparql-results#"
>
        <head>
                <variable name="title"/>
                <variable name="year"/>
        </head>
        <results ordered="false" distinct="false">
                <result>
                        <binding name="title">
                                <literal
                                        datatype="http://www.w3.org/2001/XMLSchema#string"
                                >
                                        Introduction to MPEG 7: Multimedia Content Description
                                        Language
                                </literal>
                        </binding>
                        <binding name="year">
                                <literal
                                        datatype="http://www.w3.org/2001/XMLSchema#integer"
                                >
                                        2002
                                </literal>
                        </binding>
                </result>
                <result>
                        <binding name="title">
                                <literal
                                        datatype="http://www.w3.org/2001/XMLSchema#string"
                                >
                                        MPEG-7 Audio and Beyond: Audio Content Indexing and
                                        Retrieval
                                </literal>
                        </binding>
                        <binding name="year">
                                <literal
                                        datatype="http://www.w3.org/2001/XMLSchema#integer"
                                >
                                        2006
                                </literal>
                        </binding>
                </result>
        </results>
</sparql>
```

Listing 3-15. SPARQL results example in XML format

4. Metadata for interactions of 3D objects

4.1. Introduction

As follows from Section 3, MPEG-7 Textual Annotations and MathML content notation can contribute to the new problem of interaction descriptions. However, they may cover only fragments of interactions metadata. Without a complete solution it is impossible to build useful metadata descriptions of object interactions that can be used by search engines and object matching algorithms.

Problems of new metadata applications are usually covered by adaptations and simple modifications of existing metadata solutions. However, existing metadata solutions were designed to describe a static or linear content, i.e. still images, documents or movies. In case of interaction descriptions the problem is more sophisticated. An interaction is not a simple 'content'. Interaction influences modification of the content, i.e. the object, but interaction itself is an action based on object behavior. The behavior is represented as an algorithm expressed by a computer program. The interaction metadata does not have to describe the whole algorithm, which may be very complex or confidential. To enable analyzing chains of subsequent interactions it is enough to describe only the result of execution of such computer program. However, existing metadata solutions are not applicable, since they are not capable of describing an object state that depends on many variables. There is a need for a new metadata solution able to mix descriptions of both: the object (the content) and results of object behavior (results of computer program execution).

The new metadata solution has to employ new approach to metadata construction and to ensure widest possible interoperability, particularly in the area of content description. The content description area and interoperability issues are described in Section 4.2, where applications of possible state-of-the-art metadata solutions for 3D objects are reviewed. While the new concept of interactions metadata is presented in Section 4.3.

4.2. Application of existing metadata solutions for 3D objects

Current metadata solutions presented in Section 2 were designed for multimedia in a form of a still image, audio or video. They are also tailored for different domains of purposes and can only be used as a base or as a part of 3D multimedia object metadata.

General purpose metadata solutions like Dublin Core and XMP are applicable for any multimedia assets. They do not cover any topics specific for 3D multimedia objects and therefore cannot be used as the only metadata solution. They may be used as a base of object metadata and facilitate object search queries that use only basic object information.

Metadata designed for audio and still images included some general purpose elements, technical elements and some management elements. They do not have any direct relation to 3D multimedia objects. However, they can be used as a part of 3D multimedia object metadata. 3D objects may contain textures and sounds which can be described by audio and image metadata solutions. 3D object textures may also be movies, so audio-video metadata may be also created according to existing audio-video metadata solutions. They can be a part of 3D multimedia object metadata covering the area of texture description. Moreover, metadata solutions like P/Meta or AAF also support descriptions related to digital asset management processes, which can be applied for 3D multimedia objects.

Though all the above metadata solutions are partially applicable for 3D multimedia objects, it is not possible to formulate a single, uniform metadata covering all aspects of a 3D object. As mentioned in Section 2.5.4, there is only one solution that is fully applicable for 3D multimedia objects – the MPEG-7 standard, with 3DSEAM extension. Such combination covers four areas:

- descriptions of general semantics and production, technical, and management information;
- descriptions of geometry;
- spatio-temporal decomposition;
- three dimensional decomposition.

MPEG-7 with 3DSEAM extension is capable to address a specific fragment of a 3D scene or a 3D object. Hence, it can be used not only as a content description solution, but also as a tool to attach interaction descriptions to a specific object or a fragment of an object. This feature makes it possible to construct interactions metadata of complex objects and makes MPEG-7 with 3DSEAM a good solution for implementation of interactions metadata concept.

4.3. Concept of interactions metadata

Problems listed in Section 4.1 show that limitations of currently available metadata solutions are not capable of describing object interaction properties. To deal with this situation, we propose a new approach to interaction metadata. The proposed approach is based on following assumptions. To be interactive, objects need to have a behavior. Behavior is encoded in a form of a computer program. Computer program of an interactive object has at least one communication interface – an API, which allows exchanging information with the 3D virtual world or other interactive objects. Such API includes a number of functions and attributes that reflect the state of an object. Knowing the API and the computer program it is possible to precisely describe object interaction characteristics and possible object interactions, as well as match compatible objects.

Following assumptions stated above, we introduce two new concepts: *Interaction Interfaces* and *Multimedia Interaction Model*. The *Interaction Interfaces* concept is used to describe the API of an object in terms of functions and attributes related to object interactions. Both elements of the API are represented by object parameter types of the Interactive Interface and are accompanied with semantic descriptions. The Interaction Interfaces are used in metadata as object interaction characteristic and as a dictionary of parameter types referenced in the Multimedia Interaction Model.

The *Multimedia Interaction Model* is a conceptual model of an interaction described from the point of view of an object. It means that the interaction description is build around object reaction and a context that triggers the reaction. This approach comes from areas of computer science dealing with active environments, especially active databases. Research on the active databases resulted in paradigm called *Event-Condition-Action* (ECA) [16]. The Event-Condition-Action rules paradigm was introduced in late eighties by the members of the HiPAC project. The semantics of ECA rules are straightforward: when an event occurs, evaluate the condition associated with the event; and, if the condition is satisfied, trigger the action. The ECA rules are used up to date to express active DMBS features and event-reaction oriented features in other systems [12][60][85][109]. The way the ECA paradigm is used in the Multimedia Interaction Model is the following. The *Event* element corresponds to the trigger of an interaction. The trigger is a change of a selected property of some object or a virtual environment in which the interaction occurs. The *Condition* element describes the context of an interaction. The context represents a state of selected objects or the virtual environment that is required to allow the action. The *Action* element matches the object reaction. The reaction describes how the object changes itself in a reply to the event in a given context.

Example 4-1

Consider a three dimensional virtual space with 3D objects scattered throughout that space. A 3D sphere travels through that space and bounces off 3D objects. Each 3D object has some color. The sphere changes its color when bouncing off green and blue objects. The ECA rules of such interaction are the following:

- Event – collision of a ball and some object.

  The trigger is a change of the collision property of the sphere.

- Condition – the color of the object has to be either green or blue.

  The context is a color of the object, with which the sphere collided.

- Action – sphere color is altered to red and yellow alternately.

  The reaction is a change of the sphere color.

The Multimedia Interaction Model and Interaction Interfaces do not replace existing interaction metadata standards. The new concepts complement existing solutions providing the structure for interaction descriptions and a dictionary of interaction parameter types, which enables building more precise interaction descriptions. While both proposed concepts have to be used in conjunction with some 3D metadata solution, they are not tied to any particular metadata standard. Although the best results can be obtained when used with the MPEG-7 standard extended by 3DSEAM, because at present those standards are the most appropriate to describe 3D objects.

The Multimedia Interaction Model is described in detail in Section 5, followed by the description of Interaction Interfaces presented in Section 6.

5. Multimedia Interaction Model

5.1. Basic definitions

Object is an entity composed of a unique identifier (O-ID), non-empty set of parameters, three dimensional (3D) geometry, and behavior logic. 3D geometry and behavior logic may be empty. Empty 3D geometry means that it does not contain any points. Empty behavior logic means that it does not contain any instructions. Some object parameters may concern object 3D geometry.

Object parameter is a function which represents a specific object characteristic. Evaluation of the function may require some arguments.

Object parameter value is a result of an evaluation of the object parameter function for given arguments.

Object parameter type is a class of object parameters with the same data type of their values and the same semantics. Each object contains only one parameter of a specific type.

We distinguish four kinds of objects:

1. **Interactive object**: an object with non empty 3D geometry and non empty behavior logic.
2. **Non-interactive object**: an object with non empty 3D geometry and empty behavior logic.
3. **Interactive environment**: an object with empty 3D geometry and empty behavior logic.
4. **Non-interactive environment**: an object with empty 3D geometry and non empty behavior logic.

In the reminder of this thesis saying "object" we mean all four kinds of objects listed above. Saying "environment" we mean objects from the third and the fourth category.

3D Virtual World is a continuous three dimensional space and a set of objects existing inside this 3D space. The set of objects is composed of at least one interactive object, an arbitrary number of non-interactive objects, and exactly one environment. All objects except of the environment are positioned in the 3D space, i.e. there are geometrical relations among them.

3D interaction space is a non-empty set of continuous 3D subspaces of the 3D Virtual World. Each subspace contains a set of objects positioned inside this subspace. The set of objects can be empty.

Object state is a set of parameter values of this object. As some parameters may concern object 3D geometry, the 3D geometry is indirectly reflected by an object state.

Object state change is a modification of the value of at least one object parameter.

Objects may mutually interact. An **interaction** occurs when a change of the state of one object causes a change of the state of another object.

Given a set of objects of the 3D Virtual World and a state change of an object composing this set, than the set of objects of the 3D Virtual World is split into three subsets:

1. a subset of **trigger objects**, i.e. objects whose state change initiated an interaction;
2. a subset of **interacting objects**, i.e. objects whose state changed as a result of an interaction;
3. a subset of **context objects**, i.e. objects not taking part in the interaction.

A part of the interacting object behavior logic that handles an interaction is called **interaction logic**.

5.2. Concept

Searching and matching 3D objects based on object interaction properties require appropriate 3D object metadata describing these properties. Metadata description of object interactions is a set of separate descriptions of all possible object interactions. A single interaction of an object is described according to the Multimedia Interaction Model (MIM) proposed in this thesis.

MIM is an abstract model of an interaction of one trigger object with a particular interacting object. It can be applied to any interaction that occurs in a 3D Virtual World. The MIM model describes an interaction from the point of view of an interacting object.

The MIM model represents the interaction logic of an interacting object. This logic is simultaneously represented in two ways: as a mathematical description and as a semantic description.

The mathematical description represents the result of execution of the interaction logic in a formal way. It is used to calculate new state of an interacting object after an interaction. In a mathematical description predefined constants may be used, as well as values of parameters of a trigger object, the environment and an interacting object. In case of parameters of a trigger object, a mathematical description may refer to its values either before or after the change. Constants and values may be used either unchanged or transformed according to a given mathematical function. Due to mathematical calculations of a new state of an object after an interaction, it is possible to analyze chains of interactions. Such knowledge can be explored by systems of automatic object matching.

The semantic description provides meaning of an interaction. Interaction meaning can be expressed in a number of different forms, from simple set of keywords to a graph of concepts related to an ontology. Due to such approach to semantic description it is possible to analyze meaning of an interaction on different levels of abstraction, which can be used in keyword or text based search systems.

Example 5-1

Consider a 3D advertisement sign object. The sign object interacts with avatar object by changing its height depending on the change of distance and height of the avatar. Figure 5-1 depicts a schema of such interaction.

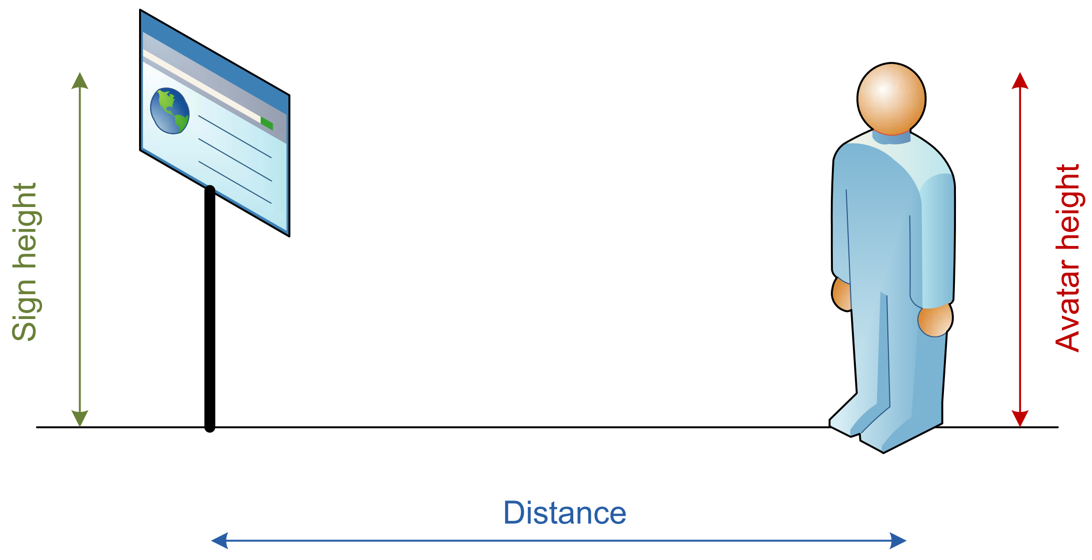

Figure 5-1. Interaction example

Mathematical description of such interaction is the following:

Definition of symbols:

AH_1 – avatar height before change

AH_2 – avatar height after change

AD_1 – avatar distance before change

AD_2 – avatar distance after change

SH_1 – sign height

SH_2 – new value of sigh height

Calculations of new height of the sign object:

$$SH_2 = AH_2 + AH_2 * \log_{10}(AD_2)$$

A free-text semantic description is the following:

```
The height of the sign object increases if an avatar grows or moves closer, and decreases if an avatar
shrinks or moves away
```

Listing 5-1. Interaction free-text semantic description example

Another form of semantic description: structured annotation, is the following:

```
What: sign height
        WhatAction: increase
                Cause: growing avatar
                Cause: avatar moving closer
        WhatAction: decrease
                Cause: shrinking avatar
                Cause: avatar moving away
```

Listing 5-2. Interaction structured semantic description example

As stated in Section 4.3 the MIM model is based on the ECA rules paradigm [16]. The Event, Condition and Action components of the MIM model are described in detail in Sections 5.3, 5.4 and 0 respectively.

5.3. Event

An event is a change of the state of an object. The object is called: trigger object. In the MIM model, the Event component contains a set of preconditions that need to be satisfied to trigger a particular interaction. The preconditions include a specification of a 3D interaction space and a non-empty set of object parameter types. To satisfy preconditions, the event has to occur inside a specified 3D interaction space and object parameters, which values are modified, have to belong to one of specified object parameter types.

The 3D interaction space is represented by a set of 3D spaces and 3D viewpoints. In MIM implementation both elements are taken from X3D language. A 3D space is defined by an X3DGeometry. By default, the geometry is positioned in the center of the interacting object. If no geometry is specified the 3D interaction space is equal to the whole 3D Virtual World. The geometry can be repositioned in space with X3D transformation tools (translation and rotation). The transformation is always done relatively to interacting object coordinate system. The 3D viewpoint definition is based on the X3D Viewpoint node. A 3D viewpoint is always positioned in the center of the interacting object and orientated according to interacting object axes.

Example 5-2

Consider again Example 5-1. In this example, event is defined as a change of height of an avatar and a change of distance between the avatar and an interacting object. The avatar has to be positioned inside a specific 3D interaction space. Event description is the following:

3D interaction space: a sphere of diameter 1000 and the center in the center of the sign object, excluding a sphere of diameter 1 and the center in the center of the sign object.

Parameter types:

Height type – height of an object in relation to the interacting object.

Distance type – distance between the center of an object and the center of the interacting object.

5.4. Condition

A condition is a set of requirements set on the states of trigger object, the environment and interacting object. A condition needs to be satisfied to trigger an action. In the Condition component of the MIM model, a mathematical description has a form of logical expression. The logical expression represents all condition requirements. Semantic description of a condition may be used as a substitute or an enhancement of the mathematical description. The Condition element of MIM metadata is optional and can be omitted. In such case an interaction will be executed at each occurrence of the event.

In MIM implementation the Condition logical expression is written in MathML content notation according to *mathml:condition.type* type. The logical expression may use symbols representing variables and constants. Each variable represents a value of a specified parameter type of the trigger object, the environment or the interacting object. A constant is a value defined according to a data type of a particular parameter type.

Example 5-3

Consider Example 5-1 extended with a condition. Condition consists of requirements regarding avatar Sex parameter and environment Ambient Light parameter. Mathematical description of the condition is the following:

<u>Description of symbols:</u>

AS – avatar Sex parameter value

Sex.FEMALE and Sex.MALE – predefined constants

EAL – environment Ambient Light parameter value

<u>Logical expression of the condition:</u>

(AS = Sex.FEMALE) & (EAL > 0.4)

A free-text semantic description of the condition is the following:

```
The avatar object needs to represent a female and the environment must be bright enough.
```

Listing 5-3. Interaction free-text semantic description example

Another form of semantic description: keyword annotation, is the following:

```
female avatar; bright light
```

Listing 5-4. Interaction keyword semantic description example

5.5. Action

An action is a change of the state of an interacting object. In the MIM model, the Action component describes the outcome of an action, a new object state, expressed as a semantic description optionally accompanied by mathematical description represented by mathematical expressions used to calculate new values of object parameters. An action can be composed of an arbitrary number of sub-actions that provide more detailed descriptions. Such approach allows building action descriptions that can be evaluated on different levels of details.

The implementation of the Action component of the MIM model uses MathML content notation for mathematical expressions. Expressions are written according to *mathml:apply.type* type. Expressions use variables and constants in the same way as condition expressions.

Description of an action is illustrated in Example 5-1.

6. Interaction Interfaces

6.1. Concept

Automatic matching of interactive objects requires definitions of objects parameter types. Optimization of search algorithms that use object interactive characteristics is more effective when object parameter types are grouped into logical subsets. Applications of the MIM model in some specific systems require definition of new parameter types. To make it all possible the MIM concept includes two additional components: Interaction Interfaces (II) and Interaction Interface Definition Rules (IIDR).

Interaction Interface Definition Rules describe how to define new object parameter types and how to group them in new interaction interfaces. IIDR are specified in Section 6.2.

An interaction interface groups object parameter types that have similar semantics, e.g. object parameter types related to object visual characteristics or object parameter types related to object internal inventory. An object implements a given II if the set of object parameter types is a superset of the set of parameter types of this II. Each object of the 3D Virtual World, except for the environment, implements at least one interaction interface – the Visual Interaction Interface defined in Section 6.3.

6.2. Interaction Interface Definition Rules

6.2.1. Interaction Interface definition rules

1. II definition consists of an interface name, ID and a set of object parameter types.
2. Name of an II has to be a unique free-text description depicting semantics of object parameter types contained in this II.
   Examples:
   - Visual Interaction Interface
   - Aural Interaction Interface
   - Inventory Interaction Interface
3. ID of an II has to be a unique and short acronym of the II name. If the name uses a term "Interaction Interface", than the acronym of that term has to be II and it has to be prefixed with a '-' character.
   Examples:
   - V-II

    - A-II
    - I-II
4. Set of object parameter types has to group all object parameter types that are logically related, i.e. they have similar semantics. The decision on exactly which object parameter types should be grouped together depends on the given application and domain and is left to the author of the interface.
5. New II may be defined as an extension of an existing II. The extending II has new name, new identifier and a set of object parameter types which is a superset of the set of object parameter types of the extended II.

6.2.2. Object parameter type definition rules

1. Object parameter type (OPT) definition consists of an identifier, a semantic description, a specification of a data type of values of object parameters of this object parameter type and a specification of arguments required to evaluate the object parameter function.
2. OPT identifier has the following syntax:
   [interaction interface ID].[object parameter type name]
   Object parameter type name has represent its semantics and has to written in camel-capped notation with a leading upper-case character.
   Examples:
    - V-II.SizeBoundingBox
    - A-II.Waveform
    - I-II.NumberOfItems
3. Semantic description of an OPT has to depict its meaning. The semantic description has to be expressed in one of two forms:
    - Text annotation based on MPEG-7 TextAnnotation data type, which allows free text, keyword, structured and dependency structure annotations.
    - Relation to an ontology concept formulated as a URI.
4. Data type specification of an OPT has to be written using tools defined in XML Schema specification [101].
5. OPT arguments specification has to list all arguments required to evaluate an object parameter function. An argument description has to contain argument name and argument data type.
6. Argument name has to depict its meaning and has to be unique in the scope of the OPT.
7. Data type of an argument has to be specified in the same way as a data type of the OPT.

6.3. Visual Interaction Interface

Visual Interaction Interface is the only predefined interaction interface. It represents object parameter types related with object visual characteristics: geometry, position and appearance and is used to build metadata of visual and geometrical interactions.

Visual Interaction Interface definition

Name:

Visual Interaction Interface

Identifier:

V-II

Object parameter types:

SizeBoundigBox	Description: Object size represented by a bounding box. Data type: x3d:BoundingBoxSize Arguments: None
SizeBoundingSphere	Description: Object size represented by a bounding sphere. Data type: x3d:SFFloat Arguments: None
SizeDimension	Description: Distance between two furthest points of an object measured along a given vector. Data type: x3d:SFFloat Argument:

	3DVectorType
ShapePrimitive	Description: Name of the shape of an object taken from a predefined dictionary, e.g. box, cone, sphere, etc. Data type: shapePrimitiveType Arguments: none
ShapeFaceSet	Description: Shape of an object described by a set of points and faces connecting these points. Data type: x3d:IndexedFaceSet Arguments: none
ShapeCustom	Description: Shape of an object described by an MPEG-7 3D Shape descriptor. Data type: mpeg7:Shape3DType Arguments: none
ShapeContour	Description: Shape of a contour of a 3D object projection onto a 3D plane described by MPEG-7 Contour-Based Shape descriptor. Data type: mpeg7:ContourShapeType Arguments:

	3DPlaneType
ShapeRegion	Description: Shape of a contour of a 3D object projection onto a 3D plane described by MPEG-7 Region-Based Shape descriptor. Data type: mpeg7:RegionShapeType Arguments: 3DPlaneType
PositionCenter	Description: Coordinates of the center of object bounding sphere or object bounding box. Data type: x3d:SFVec3f Arguments: xs:string [box | sphere]
PositionCollision	Description: Collision flag indicating collision of described object and interacting object. Data type: xs:boolean Arguments: none
Orientation	Description: Object orientation relative to 3D Virtual World axes. Data type: x3d:MFRotation Arguments:

	none
AppearanceTexture	Description: Object texture described by MPEG-7 Texture Browsing descriptor. Data type: mpeg7:TextureBrowsingType Arguments: none
AppearanceTextureHomogeneous	Description: Object texture described by MPEG-7 Homogeneous Texture descriptor. Data type: mpeg7: HomogeneousTextureType Arguments: none
AppearanceColor	Description: Object uniform color. Data type: x3d:SFColor Arguments: none
AppearanceColorDominant	Description: Object color described by MPEG-7 Dominant Color descriptor. Data type: mpeg7:DominantColorType Arguments: none
AppearanceColorLayout	Description:

	Object color described by MPEG-7 Color Layout descriptor. Data type: mpeg7:ColorLayoutType Arguments: none
AppearanceColorStructure	Description: Object color described by MPEG-7 Color Structure descriptor. Data type: mpeg7:ColorStructureType Arguments: none
AppearanceTransparency	Description: Object transparency. Data type: x3d:SFFloat Arguments: none

Full XML schema of the Visual Interaction Interface is presented in Appendix D.

7. XML Schema of Interaction Metadata

7.1. Interactions schema

Interactions schema defines a structure of a final metadata document. The structure of the final document is composed of references to required Interaction Interfaces and a list of interaction descriptions.

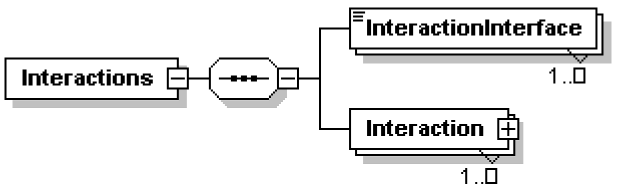

Figure 7-1. Interactions schema structure

The *Interactions* element has an optional *objectURI* attribute, which identifies the object described by this metadata document. The attribute is optional, because it is not needed if the MIM metadata is embedded in other object metadata, like MPEG-7. Each *InteractionInterface* element contains an URI of an Interaction Interface definition. Single interactions metadata may reference one or many Interaction Interfaces. The *Interaction* element contains description of a single interaction constructed in accordance with the Multimedia Interaction Model schema presented in Section 7.2. Interactions metadata may be composed of one or many Interaction elements.

The Interaction schema is described by an XML Schema presented in Appendix A.

7.2. Multimedia Interaction Model schema

7.2.1. Overall structure

A single interaction is composed of three main MIM model elements *Event*, *Condition* and *Action*. This triplet is complemented by a fourth element called *Semantics*. The interaction description contains one or more *Event* elements, zero or more *Condition* elements, one *Action* element and zero or one *Semantics* element.

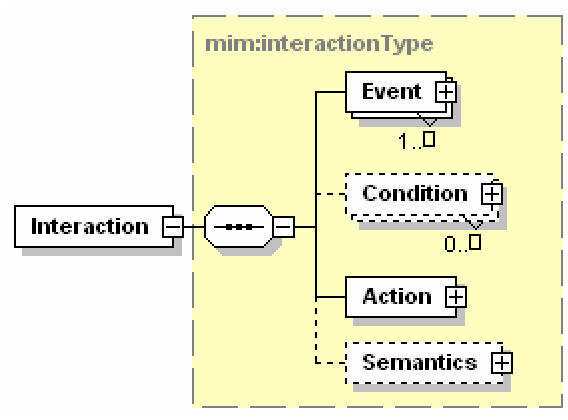

Figure 7-2. Single interaction schema structure

7.2.2. Event element schema

Event element describes an event, i.e. a change of a single parameter of a specified type, which can trigger an interaction. The *Event* element contains *InteractionSpace*, *Parameter*, *Argument* and *Semantics* elements. The *InteractionSpace* element is optional. The *Parameter* element is mandatory and may be used only once. The *Event* element contains also zero or more *Argument* elements and zero or one *Semantics* element.

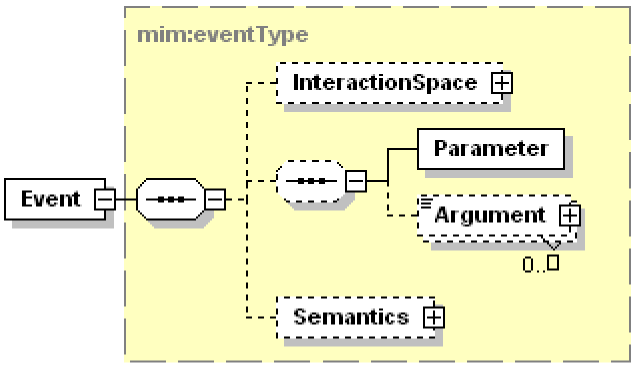

Figure 7-3. Event schema structure

The optional *InteractionSpace* element describes a 3D interaction space of an event. It can be expressed as a set of 3D volumes and 3D viewpoints, or with a semantic description. 3D volumes are defined with the use of *Space* element, which is based on X3D X3DGeometry and Transform elements. 3D viewpoints are represented by *Viewpoint* element, also based on the Viewpoint element from the X3D language. Semantics of a 3D interaction space can be described with the use of *Semantics* element, which has the mpeg7:TextAnnotationType type and provides MPEG-7 annotation tools.

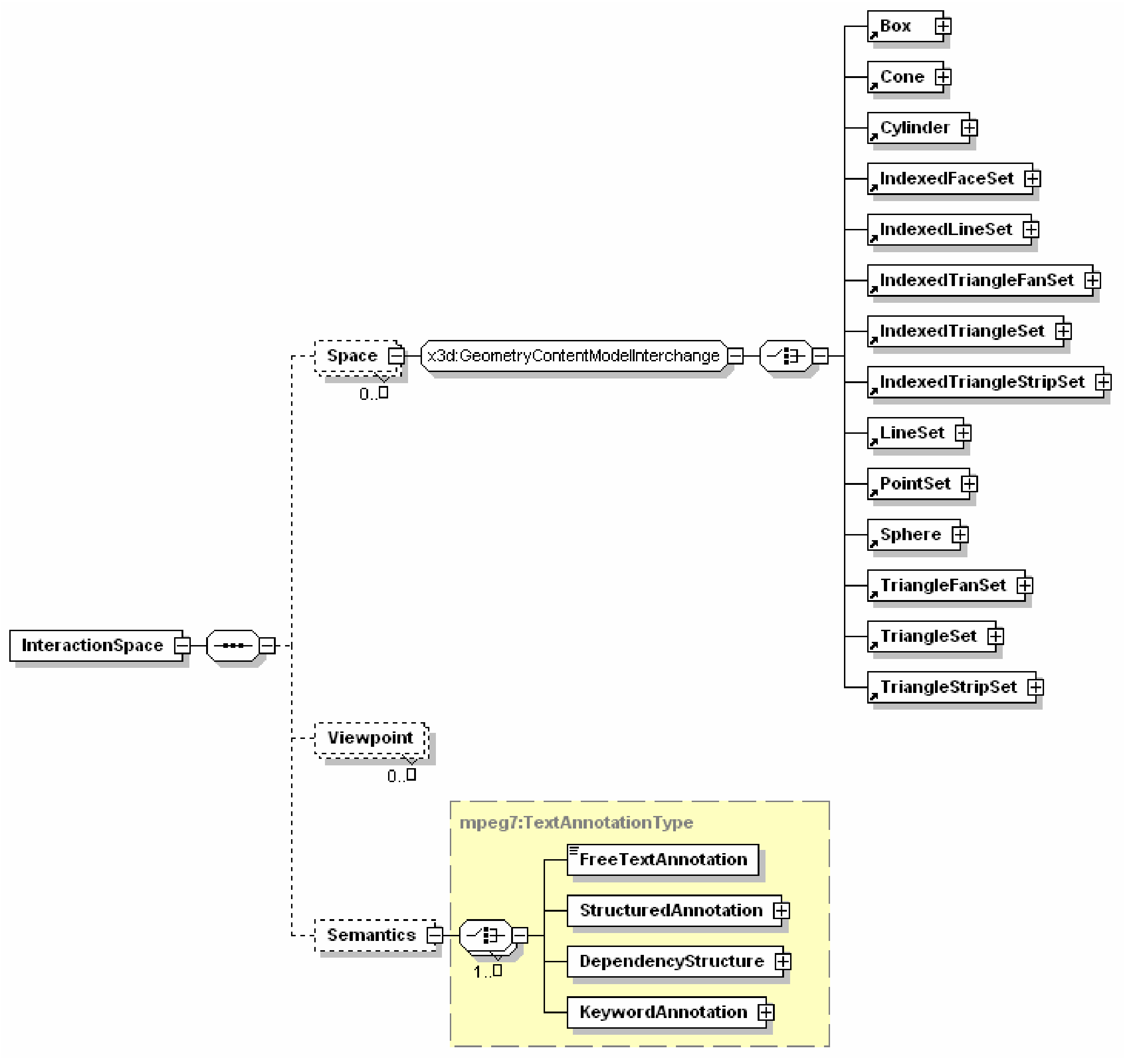

Figure 7-4. InteractionSpace element schema structure

The second child element of the Event element is the *Parameter* element. It contains a *typeID* attribute, which is an URI of the parameter type. Third child element is the *Argument* element that provides a tool for describing parameter arguments. Semantics of an event can be described with the use of fourth element: *Semantics*. The *Semantics* element has the mpeg7:TextAnnotationType type.

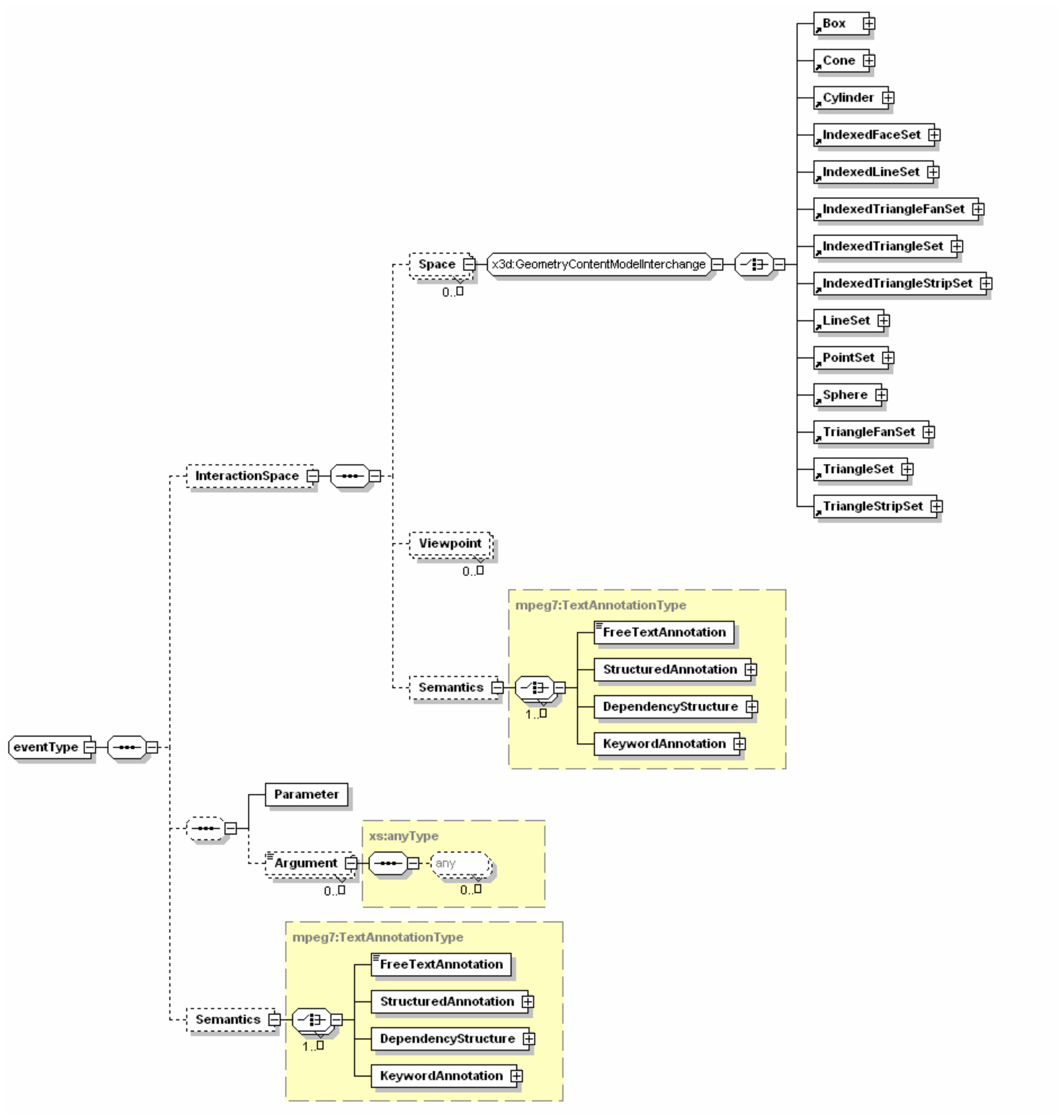

Figure 7-5. Full structure of the Event element schema

7.2.3. Condition element schema

Second element of interaction description, is the optional *Condition* element. It describes all requirements that have to be satisfied to trigger an action as a response to an event. The *Condition* element contains one mandatory *Expression* element, one or more *ExpressionArgument* elements and zero or one *Semantics* element.

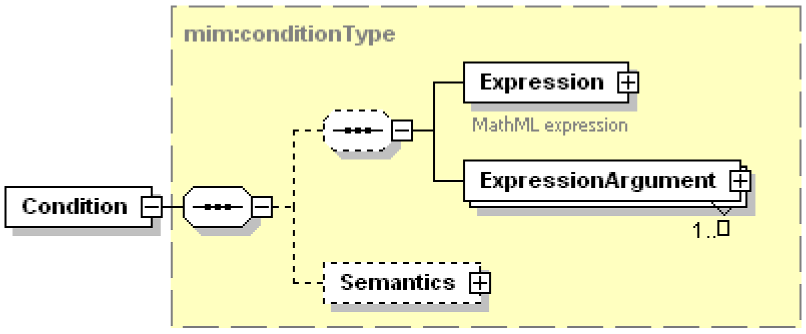

Figure 7-6. Condition schema structure

The *Expression* element contains a MathML expression written according to the MathML content notation and to a mathml:condition.type type. The second Condition child, the *ExpressionArgument* element, contains definition of a variable or a constant used in the MathML expression. The variable or constant definition is written according to *valueReference* type defined in the MIM model schema.

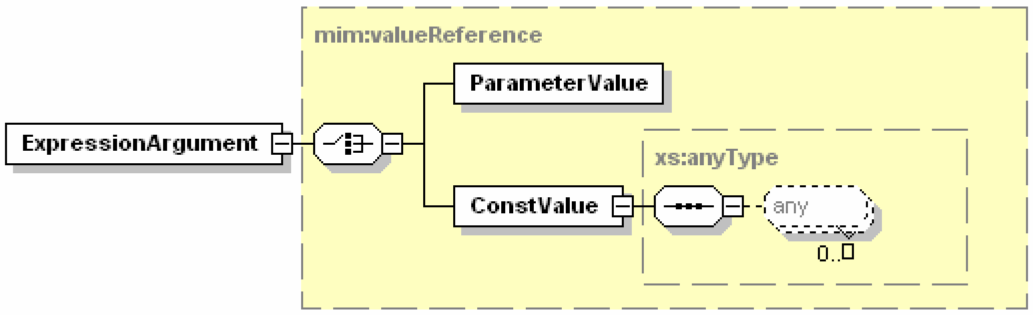

Figure 7-7. ExpressionArgument element schema structure

The *valueReference* type contains *ParameterValue* and *ConstValue* elements. The *ParameterValue* element corresponds to a variable and includes *typeID*, *beforeOrAfter* and *object* attributes, that allow to specify the source of the value. The *typeID* attribute contains a URI to a parameter type. The *beforeOrAfter* attribute allows selecting a value before or after the event. The *object* attribute may contain one of three values: 'trigger', 'interacting' and 'environment'. It indicates if the value is taken from the parameter of the trigger object , interacting object or the environment. By default the last two attributes are optional and have default values of 'after' and 'trigger' correspondingly. The *ConstValue* element also contains the same *typeID* attribute to select a particular parameter type for the value. The constant value is written inside the *ConstValue* element according to the schema of a specified parameter type. Both, *ParameterValue* and *ConstValue* elements contain one more attribute called *variableBinding*. The *variableBinding* attribute contains a symbol used in the MathML expression to reference a variable or a constant value.

The last *Condition* child, the *Semantics* element, provides annotation tools defined by the mpeg7:TextAnnotationType type.

7.2.4. Action element

The last element of the MIM model schema is the *Action* element. It described the action which is a result of the events and satisfied conditions. The action is a change of values of some parameters of the interacting object. The *Action* element contains *SubAction*, *Parameter* and *Semantics* elements. All child elements are optional and apart from the *Semantics* element, they can occur multiple times.

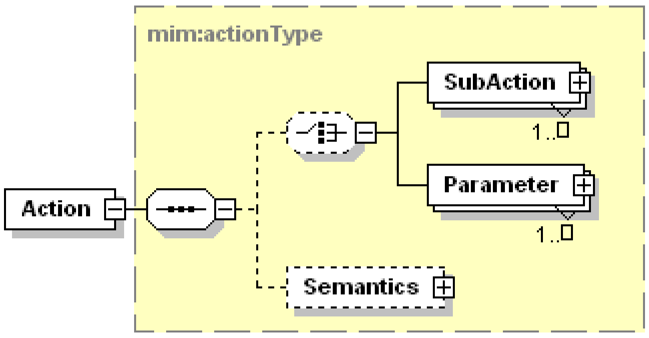

Figure 7-8. Action schema structure

The *SubAction* element allows an action to be decomposed into sub-actions, which allows for more detailed descriptions of action steps. The *Parameter* element contains an expression that describes how to calculate new value of a specified parameter. The *Parameter* element is composed of a *typeID* attribute and two elements: *Result* and *ResultArgument*.

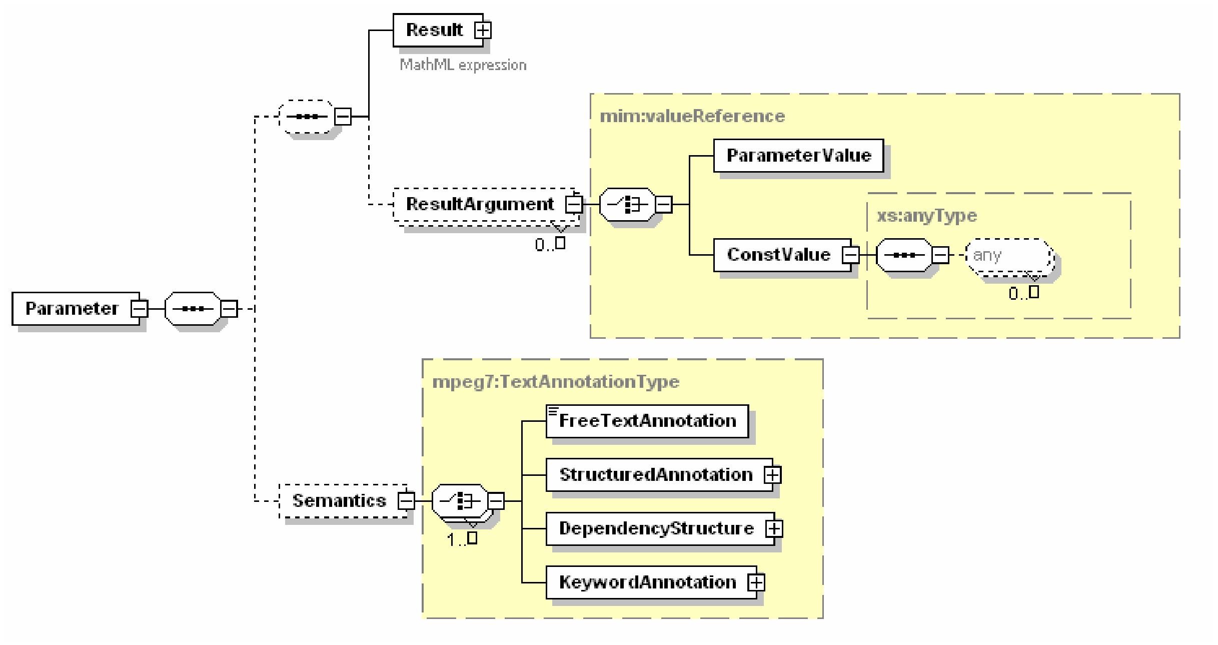

Figure 7-9. Parameter element schema structure

The *typeID* attribute references a parameter type and resolves to a specific parameter of the interacting object. This parameter is a subject of the action and it obtains a new value. The new value for the parameter is calculated according to a MathML expression contained in the *Result* element. The *Result* element contents are written according to the MathML content notation and mathml.apply.type type. The variables and constants used in the expression are defined in *ResultArgument* elements. The *ResultArgument* contents are written according to *valueReference* type defined in the MIM model schema and described in Section 7.2.3.

The *Semantics* element has the mpeg7:TextAnnotationType type and contains semantic description of an action. The *Semantics* element can be used alone or in parallel with new parameter value expression.

7.2.5. Semantics element

The global *Semantics* element corresponds to a semantic description of the whole interaction and can be used along the formal interaction description. The *Semantics* element has the mpeg7:TextAnnotationType type and provides MPEG-7 annotation tools.

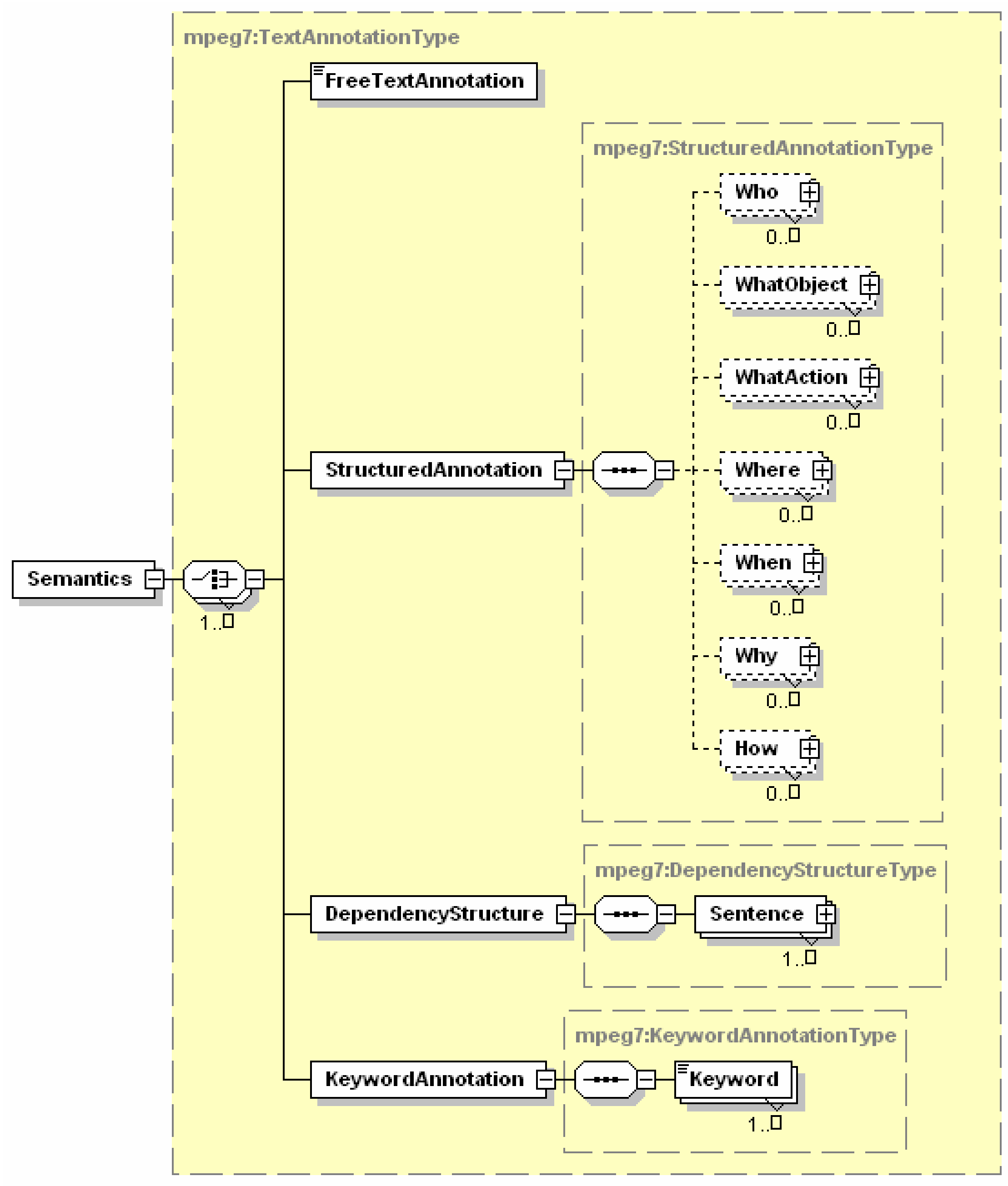

Figure 7-10. Extended schema view of the Semantics element

7.3. Interaction Interface schema

The Interaction Interface schema provides a structure for definition of new interaction interfaces and parameter types. Each interaction interface is composed of a name, ID and a set of object parameter types. The definition of an interaction interface can be also extended with a semantic description. All this information is contained in two attributes: *ID* and *name*, and two elements *ParameterType* and *Semantics*. Each interaction interface definition may contain one or many *ParameterType* elements and zero or one *Semantics* element.

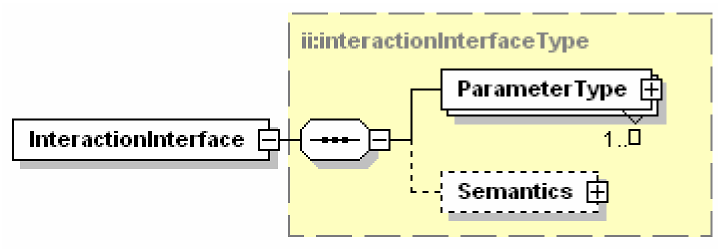

Figure 7-11. Interaction Interface schema structure

The *ID* and *name* attributes contain interaction interface descriptors as defined in Section 6.2.1. The *ParameterType* element contains an *ID* attribute which holds information about parameter type ID and four elements: *DataType*, *Argument*, *Semantics* and *Ontology*.

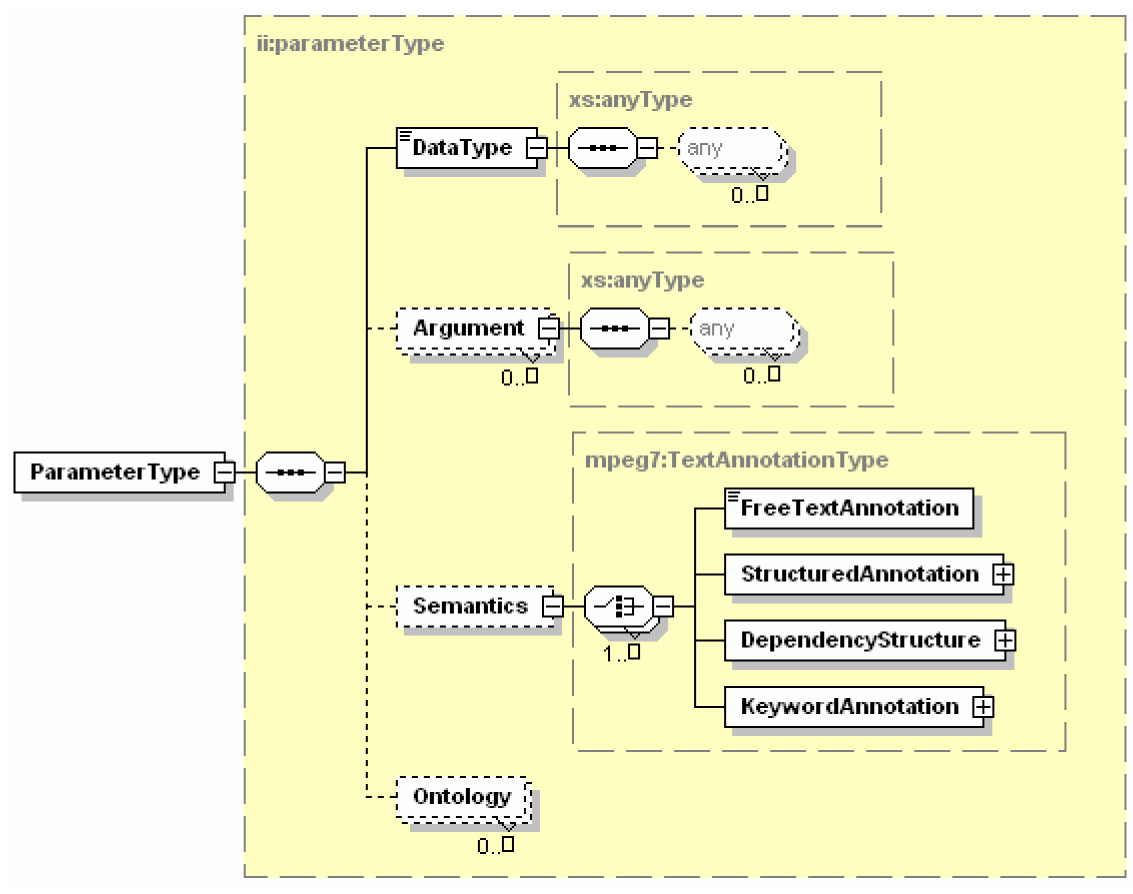

Figure 7-12. ParameterType element schema structure

The *DataType* element contains a definition of a data type of parameter values and it is mandatory. The *Argument* element may occur zero or many times and contain name and definition of a data type of a parameter argument. Both, parameter data type definition and argument data type definition have to be written as simple or complex types of XML Schema. The *Semantics* element is of the mpeg7:TextAnnotationType type and contains semantic description of a parameter type. The last element, *Ontology*, can occur zero or multiple times and it is a relation to a concept in an external ontology system. The *Ontology* element includes two attributes, which are URIs to an ontology and a concept within this ontology.

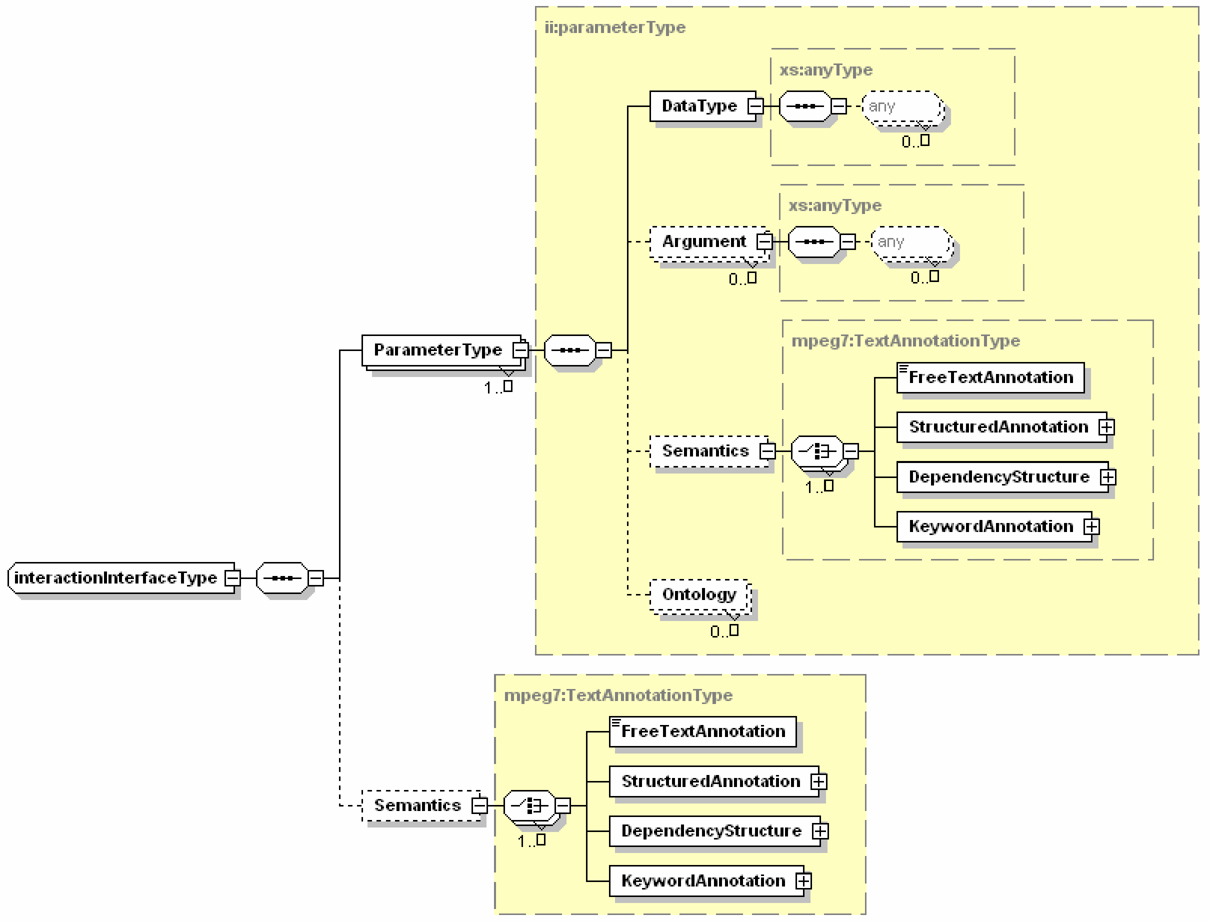

Figure 7-13. Full structure of Interaction Interface schema

The second and last element of the Interaction Interface definition is the *Semantics* element. The *Semantics* element has the mpeg7:TextAnnotationType type and provides MPEG-7 annotation tools for creating semantic description of the whole interaction interface.

8. Interaction Metadata Query sub-Language

8.1. Introduction

To make search engines able to use interaction metadata a query language is required. Since there are no interaction metadata standards, existing metadata query languages does not contain any features for querying objects using interaction criteria. To fully exploit the potential of interaction metadata based on the MIM model such language has to support basic keyword queries, semantic searches and object matching. Also, the query language has to be extensible to be able to employ new Interaction Interfaces, as well as new interaction parameters. Finally, as the interaction metadata will be always only a part of complete object description, the query language has to be modular, to enable its usage along with different search languages and systems.

In this dissertation we propose Interaction Metadata Query sub-Language (IMQL) which fulfils all the above requirements and complements MIM model to create complete solution for interaction metadata. The IMQL provides a set of operators that can be used in simple keyword comparisons and advanced semantic searches. IMQL syntax relates directly to the MIM model structure and provides references to interaction interfaces of compared objects and parameter types of these objects. Object matching queries are be complex and difficult, especially in the field of interactions. The use of interaction interfaces facilitates object matching queries. The IMQL integrates with Interaction Interface definitions permitting the use of new object parameter types and value data included directly in the query. IMQL is composed of two parts: first, defining the source of object metadata; second, specifying filtering criteria. The first part can be replaced by results of another search query, while the output of the filtering criteria can be passed for further processing. Such solution allows embedding an IMQL query within larger query and performing a complex search by a single search system.

The IMQL is based on the following assumptions. It is assumed that interactive objects are grouped in uniquely identified repositories. Each repository contains a set of interactive objects. Each object in the repository is identified by a unique URI and has interaction metadata built according to the MIM model. IMQL defines the syntax of the query language and meaning of different language elements and comparison operators. Any specialized comparison or matching algorithms, especially semantic analysis algorithms, are left to be implemented by search engines.

IMQL query syntax:

```
"FROM" repository_URIs ("WHERE" where_expr)?
```

Listing 8-1. IMQL query syntax

The IMQL is composed of two clauses: the FROM clause and the WHERE clause. The FROM clause defines the source of interactive objects, and the WHERE clause contains filtering criteria. The result of execution of IMQL query is always a set of URIs of objects whose interaction metadata matched given criteria.

8.2. FROM clause

The purpose of the FROM clause is to identify interactive objects whose metadata are evaluated against conditions defined in the WHERE clause. Objects to be evaluated are taken from repositories of interactive objects identified by *repository_URIs*. The FROM clause is the only required part of the query. If the WHERE clause is omitted, the query returns a set of URIs of all objects from all given repositories. Objects from each repository are taken in an arbitrary order. However, the order of groups of objects taken from subsequent repositories is the same as the order of repositories in the query. Object ordering is illustrated by Example 8-1.

Example 8-1

The above ordering rule means that if a query selects all objects from repository A containing objects A1 and A2, and from repository B containing object B1 and B2, the result is one of the following:

```
A1, A2, B1, B2
A2, A1, B1, B2
A1, A2, B2, B1
A2, A1, B2, B1
```

Listing 8-2. Possibile IMQL results

The order of objects from each repository is arbitrary, while the order of groups of objects from different repositories reflects the order of repositories in the query. Any other order of object URIs is invalid.

The ordering of objects resulting from the FROM clause is not final. The results can be reordered after filtering with the WHERE clause. The IMQL, in contrast to SQL or XQuery, does not provide any internal ordering facilities. The fact that it always returns object URIs and it operates on hierarchical metadata means that there is no single property that could be used for ordering. Instead, IMQL allows external ordering mechanisms that can be provided by search systems. Such ordering mechanisms can be based on semantic relevancy factors that depend on particular search engine implementation. The proposed approach enables competition among search systems since better results will be returned by search systems with better relevancy algorithms.

FROM clause syntax:

```
repository_URIs          = single_uri (", "single_uri)*
single_uri               = ""URI""
```

The URI provided in the FROM clause represents a Uniform Resource Identifier as defined by RFC 2396 [69], as amended by RFC 2732 [70]. The URI has to be enclosed in single quotes. Multiple URIs have to be separated by commas.

Example 8-2

Consider Example 8-1. The notation of the query is the following:

```
FROM A, B
```

Listing 8-3. Sample IMSQL query

The above query returns URIs of all objects from repository A and URIs of all objects from repository B. The list of objects is not filtered in any way and the order of objects is as described in Example 8-1.

8.3. WHERE clause

The optional WHERE clause serves as a filter for interactive objects identified by the FROM clause. The expression in the WHERE clause, called the *where_expr*, is evaluated once for each of these objects. The *where_expr* is a logical expression of requirements set on interaction properties of evaluated objects. If the Boolean value of the *where_expr* is *false*, the object is discarded. The order of objects remains the same as defined in the FROM clause.

The IMQL is designed as a modular language and it can be used as an extension of other query languages. The integration is accomplished by embedding the WHERE clause *where_exp* as one of search conditions in the extended query. In such case the FROM clause is omitted and search system has to serve as a proxy between source list of objects and a set of object metadata URIs required for the evaluation of the *where_exp*. The result of *where_exp* evaluation, a filtered set of object URIs, can be passed further for next processing steps.

WHERE clause syntax:

```
where_expr            = condition_expr (logical_operator condition_expr)*
```

The *where_expr* is composed of at least one *condition_expr*. If there is more than one *condition_expr*, each two are separated by a *logical_operator*.

```
condition_expr        = (single_condition (logical_operator single_condition)*) | (“(“where_expr”)”)
```

The *condition_expr* consists of at least one *single_condition* or exactly one *where_expr* enclosed in braces. If there is more than one *single_condition*, each two are separated by a *logical_operator*. The fact that higher-level *where_expr* appears inside *condition_expr* and is enclosed in braces, enables creation of multi-level logical conditions with easily defined operation precedence.

```
logical_operator      = "&" | "|" | "&!" | "|!"
```

The *logical_operator* is either logical AND (&) or logical OR (|). The other two are shortcuts for using these operators in conjunction with logical NOT: logical AND NOT (&!) and logical OR NOT (&!).

```
single_condition      = operand comparison_operator operand
operand               = element_identifier | value
```

The *single_condition* consists of exactly two *operand*s separated by *comparison_operator*. The operand is either an *element_identifier* or *value*.

```
comparison_operator   = "==" | "!=" | "<" | "<=" | ">" | ">=" | “=” | "~=" | ">>"
```

IMQL supports nine comparison operators:

• equal	==
• not equal	!=
• less	<
• less or equal	<=
• grater	>
• grater or equal	>=
• almost equal	=
• like	~=
• in	>>

The first six operators are basic comparison operators. They work according to mathematical rules and their standard use does not require further explanations. Basic comparison operator can be used also to compare 3D features, like 3D points and 3D geometry. For 3D points, two points are equal if all three point coordinates are exactly the same. A 3D point is grater or less than other 3D point if the line from the beginning of local coordinate system to this 3D point is longer or shorter correspondingly. For 3D geometry the comparison is done based on the volume of the geometry. For example, a sphere with diameter equal 2 is less than a box with a length, width and height of 2.

Seventh operator, 'almost equal' (=), is useful for comparing keywords with interaction semantic descriptions. Especially for descriptions linked with an ontology or when they are analyzed with natural language processing (NLP) algorithms. In such case submitted keywords can be matched not only with exact description term, but also with higher-level equivalents from an ontology or terms with similar meaning. In general, the 'almost equal' operator can be used for any values for which it is possible to calculate some similarity measures. For example, for object shape: a 3D shape can be described with an MPEG-7 Shape3D descriptor and some similarity measures can be obtained. The raw result of such comparison is a number: relevancy or similarity factor, not a Boolean value. The final Boolean value is obtained by comparing the resulting number with a given threshold. If the number is above the threshold, the 'almost equal' operator returns true. The algorithm for calculating relevancy or similarity and the comparison threshold are not included in the IMQL definition and are defined by a search system.

The eighth operator, ‘like’ (~=), is a comparison operator that works with regular expressions. It returns *true* if the value of selected object interaction property matches a given pattern. Regular expressions are constructed according the POSIX extended regular expressions syntax. However, depending on the search system, available regular expressions syntax can be extended to support Perl-compatible regular expressions (PCRE).

The last operator, ‘in’ (>>) is not a strict comparison operator. It verifies if a given value is contained in a set provided as the right operand. By default it uses strict comparison to match given value with values taken from the set. However, it is possible to use a set of regular expressions and match the value with regexp patterns. In such case the operator has to be prefixed with tilde ‘~’. This operator can be also used for verifying 3D features, like point in space. In such case, the right operand is not a string but a 3D space, and the left operand has to be a set of 3D points. The ‘in’ operator verifies if 3D points are contained inside the 3D space.

```
element_identifier     = ii | event | condition | action | interaction

interaction            = “interaction.paramtype.name” | interaction.paramtype.id” |
                          “interaction.semantics”

ii                     = “ii” | "ii.id" | "ii.name" | “ii.semantics”

event                  = "event.space" | “event.space.semantics” | "event.paramtype"
                          | event.paramtype.id" | ”event.paramtype.name"
                          | ”event.paramtype.semantics" | “event.semantics”

condition              = "condition.” source ”.paramtype" | “condition.” source ”.paramtype.id"
                          | ”condition.” source ”.paramtype.name"
                          | ”condition.” source ”.paramtype.semantics" | “condition.semantics”
                          | "condition." parameter ".value" ( "." ( "after" | "before" ) )?

action                 = "action.” source ”.paramtype" | “action.” source ”.paramtype.id"
                          | ”action.” source ”.paramtype.name"
                          | ”action.” source ”.paramtype.semantics" | “action.semantics”
                          | "action." parameter ".value" ( "." ( "after" | "before" ) )?

source                 = “trigger” | “interacting” | “environment”

parameter              = source ”.” opt_id

opt_id                 = [Object Parameter Type as defined in Section 6.2.2]
```

The *element_identifier* is a reference to the whole interaction or to specific property of one the MIM model elements. In case of a reference to an interaction (*interaction*) the *element_identifier* can address parameter types used in any element of the MIM model (*intraction.paramtype.name* and *interaction.paramtype.id*) or interaction semantics (*interaction.semantics*). The *element_identifier* can refer also to interactive interface (*ii*), interaction *event*, *condition* or *action*. For interactive interfaces the *ii* may refer to interactive interface ID (*ii.id*), name (*ii.name*), semantic description (*ii.semantics*) or all of them (*ii*). Interactive Interface name and ID are both textual values and can be compared with a string (e.g. keyword), regular expression or a term in an ontology. Interactive Interface semantic description can be either a free text or a set of keywords, or more structural description (structural or dependency annotation). The free text and keyword annotations are handled in the same way as name and ID, while more complex annotations can be analyzed with more advanced text analysis algorithms.

Example 8-3

Consider a search that takes interactive objects from repository ‘LocalRep’ and returns only objects implementing Aural Interaction Interface. The query for such search is the following:

```
FROM ‘LocalRep’ WHERE ii.name == ‘Aural Interaction Interface’
```

Listing 8-4. Sample IMQL query

The same query rewritten to match all interactive interfaces with ‘aural’ in name, not only the Aural Interactive Interface, is the following:

```
FROM ‘LocalRep’ WHERE ii.name ~= ‘.*[aA]ural.*’
```

Listing 8-5. Sample IMQL query

The *event* refers to either an interaction space or types of parameters that trigger the event. The interaction space is represented by a semantic description or a set of 3D geometries and can be compared with other 3D features like 3D points or 3D geometries. Trigger object parameter types are represented by their IDs, names or semantic descriptions. The query may refer to any of these fields separately: ID (*event.paramtype.id*), name (*event.paramtype.name*), semantics (*event.paramtype.semantics*), or to all of them at once (*event.paramtype*). The data types of these fields are identical to those presented for Interaction Interfaces.

The *condition* query element is a reference to types of object parameters used in interaction condition expression or to values of specific object parameters. Object parameter types can be referred in the same way as trigger object parameter types for the *event* query element, with an additional component (*source*), which indicates what interaction party is referenced. References to values of object parameters refer to exact parameters of exact objects and include a flag indicating if the query references a value before or after the event. This flag is optional and by default the reference returns value after the event.

The *action* refers to same fields as the *condition* query element, with the restriction that parameter types and parameter values are taken from the description of the action.

Exact parameter reference that is used in *condition* and *action* query elements, is composed of two parts: object identification and parameter type identification. Each parameter is bound to one of three possible interaction objects: trigger object, interacting object or the environment. The identification of an object whose parameters are referred in the query is accomplished with the use of one of three keywords: “trigger”, “interacting” or “environment”. The identification of a specific parameter for a given object uses the fact that each object can have only one parameter of a specific type. Thus, the parameter identification is done by object parameter type ID. The syntax for object parameter type IDs is defined in Section 6.2.2.

```
value               = literal | xml
literal             = numeric_literal | string_literal
numeric_literal     = integer_literal | decimal_literal | double_literal
integer_literal     = [+-]? digits
decimal_literal     = [+-]? (“.” digits) | (digits (“.” [0-9]*)? )
double_literal      = ((“.” digits) | (digits (“.” [0-9]*)?)) [eE] [+-]? digits
string_literal      = (“’” (char[^’] | escape_apos)* “’”)
digits              = [0-9]+
escape_apos         = “\’”
char                = [http://www.w3.org/TR/REC-xml/#NT-Char]
xml                 = [http://www.w3.org/TR/REC-xml/#NT-element]
```

The *value* query element stands for a real value literal. The *value* can be either a *literal* representing a simple value or an *xml* which represents values with complex data types, like for example object color of x3d:IndexedFaceSet type.

The simple value (*literal*) may be numeric or textual. This is represented by *numeric_literal* and *string_literal* query elements. The *numeric_literal* represents integer (*integer_literal*), decimal (*decimal_literal*) or double (*double_literal*) number. The *string_literal* query element is composed of any Unicode characters (*char*) except apostrophe (‘). The apostrophe character is used to enclose any *string_literal*. Any apostrophe inside the text has to be replaced by *escape_apos* element. The exact syntax for the *char* element is defined in http://www.w3.org/TR/REC-xml/#NT-Char.

The syntax of the *xml* element used for representing complex values is defined in http://www.w3.org/TR/REC-xml/#NT-element and the syntax of a concrete data type depends on a data type XML schema included in a definition of an interactive interface.

8.4. Embedding IMQL

The MIM metadata has the most expression power when combined with more general multimedia metadata standards, like for example MPEG-7. Therefore the IMQL has to be used in conjunction with query languages capable of processing these general metadata standards. As mentioned in Section 8.3, the IMQL can be embedded in queries written in other query languages forming a complex query. Complex queries enable search systems to look in standard metadata descriptions and interaction descriptions in a single process.

To provide universality of IMQL embedding rules, an IMQL query is embedded as a function. The *imql()* function takes object metadata URI (*object_metadata_URI*) and IMQL *WHERE clause* contents (*where_expr*) as arguments. The function returns Boolean value equal *true* if given metadata matches the specified IMQL *where_expr*. IMQL function syntax is the following:

```
imql ( object_metadata_URI, where_expr )
```

Listing 8-6. IMQL function syntax

Embedding IMQL query as a function allows including IMQL in an arbitrary query language: from XML related XQuery, to RDF query language SPARQL, to SQL used for querying relational databases.

Embedding IMQL in XQuery

In XQuery described in Section 3.5.1 the IMQL function is used to construct a comparison expression included in the WHERE clause of an XQuery query. The comparison expression in XQuery has the following structure:

```
ComparisonExpr = RangeExpr ( ( ValueComp | GeneralComp | NodeComp) RangeExpr )?

ValueComp = "eq" | "ne" | "lt" | "le" | "gt" | "ge"
GeneralComp = "=" | "!=" | "<" | "<=" | ">" | ">="
NodeComp = "is" | "<<" | ">>"
```

The *RangeExpr* element has many forms. One of them is *ExtensionExpr*.

```
ExtensionExpr = Pragma+ "{" Expr? "}"

Pragma = "(#" S? QName (S PragmaContents)? "#)"      /* ws: explicit */
PragmaContents = (Char* - (Char* '#)' Char*))
```

The embedded IMQL filtering function is an implementation of the *ExtensionExpr* element.

Example 8-4

Consider a repository of interactive 3D objects – interactive toys. Some of these toys are virtual 'Mega Bloks' and 'LEGO' building blocks that have an ability to change their color to match the color of connected blocks. Objects in the repository are identified by URIs and are described by MPEG-7 metadata with embedded MIM metadata.

Consider an XQuery query that selects URIs of only the LEGO blocks that interact in the way described above. The XQuery query written according to FLWOR notation [106] is the following:

```
LET $doc := doc('http://www.myrepository.com/toys/mpeg7.xml')
FOR
        $m IN $doc/Mpeg7/Description/MultimediaContent/Multimedia
WHERE
        $m//Creation/Creator/Agent[@xsi:type='OrganizationType']/Name = 'LEGO Group'
        AND
        imql(
                $m//Interactions,
                event.paramtype.id  ~= '^V-II\.Position.*'
                AND
                action.interacting.V-II.ApperanceColor.value.after ==
                        action.trigger.V-II.ApperanceColor.value.before
        )
RETURN $m/MediaLocator/MediaURI
```

Listing 8-7. Sample IMQL query embedded in XQuery

The source for the above query is a *toys/mpeg7.xml* document. The document contains MPEG-7 metadata of all toy objects contained in the *MyReporitory*. The query selects all *Multimedia* MPEG-7 metadata and checks if the *Creator* name is equal to 'LEGO Group' and if the interaction fulfills requirements described by the IMQL sub-query. The IMQL sub-query verifies two requirements. First, the IMQL sub-query checks if an object interaction is triggered by parameters related with object *Position*. Second, it checks if such interaction results in change of the color of the interacting object to the color of trigger object. The XQuery query returns URIs of matched objects.

Embedding IMQL in SPARQL

As mentioned in Section 0 metadata represented as RDF documents can be queried with the use of SPARQL. This language designed for querying RDF data sources can be extended with IMQL functionality by using IMQL function as a filter function in the SPARQL WHERE clause. Additionally to triple patterns [80] used to construct SPARQL queries, the SPARQL query language provides also a FILTER pattern [81]. The FILTER pattern can be included in the *WHERE clause* of SPARQL query. The filter defines a restriction on the group of solutions in which the filter appears.

Example 8-5

Consider Example 8-4 with one difference. This time object metadata are written in RDF format and MIM metadata is embedded directly in RDF descriptions. The SPARQL query for the same group of objects as in Example 8-4 is the following:

```
PREFIX toys:            <http://www.myrepository.com/toys/>
SELECT ?objectURI
FROM <http://www.myrepository.com/toys/rdf.owl>
WHERE {
                toys:object toys:creator "LEGO Group" .
                toys:object toys:URI ?objectURI .
                toys:object toys:interactions ?interactions .
                FILTER imql(?interactions,
                        event.paramtype.id  ~= '^V-II\.Position.*'
                        AND
                        action.interacting.V-II.ApperanceColor.value.after ==
                                action.trigger.V-II.ApperanceColor.value.before
                )
        }
```

Listing 8-8. Sample IMQL query embedded in SPARQL

The above SPARQL query selects URIs of all objects for which *creator* field is equal "LEGO Group" and for which one of their *interactions* matches the IMQL sub-query.

Embedding IMQL in SQL

Also the third universal querying language presented in Section **Błąd! Nie można odnaleźć źródła odwołania.** can be extended with IMQL. The IMQL embedding method used for SQL queries follows the solutions presented for XQuery and SPARQL. The IMQL filtering function is used as an additional condition in the *WHERE clause* of the SQL query.

Example 8-6

Consider Example 8-4 with one difference. This time object metadata are stored in a relational database. Object MIM metadata is stored as an XML document in one of database relation attributes. The SQL query for the same group of objects as in Example 8-4 is the following:

```
SELECT t.uri
FROM myrepository.toys t
WHERE
        t.crator = 'LEGO Group'
        AND
        imql(interactions,
                event.paramtype.id  ~= '^V-II\.Position.*'
                AND
                action.interacting.V-II.ApperanceColor.value.after ==
                        action.trigger.V-II.ApperanceColor.value.before
        )
```

Listing 8-9. Sample IMQL query embedded in SQL

The presented SQL query selects values from *uri* attribute of the *toys* relation. The *toys* relation is taken from the *myrepository* database. The *creator* attribute of each selected relation tuple has to be equal to 'LEGO Group" and interactions metadata taken from the *interactions* attribute have to match IMQL sub-query.

By default, it is assumed that the database relation attribute passed as the first argument of IMQL filtering function contains a complete XML document of MIM metadata. However, it is allowed for query engines to act as a proxy and use MIM metadata decomposed into native database attributes and relations.

9. Examples of MIM application and use

9.1. Introduction

To illustrate application the use of Multimedia Interaction Model and Interaction Interfaces introduced in Sections 5 and 6, in this chapter five examples of interaction metadata are presented. All examples include: a general description of an interactive object and its context, detailed description of object interactions and corresponding interaction metadata, illustration of possible metadata usage scenarios, and a short summary. Each example illustrates different aspect of interaction metadata based on MIM.

The first example, called ‘Changing Ball’, presents metadata of a simple interaction. Its goal is to show the structure of an XML metadata document as defined by the Interactions Schema and other sub-schemas introduced in Section 7.

The second example, named ‘Ad Sign’, shows the usage of mathematical expressions in interaction metadata (c.f. Section 3.3 for MathML language description).

The third example, the ‘Treasure Chest’, illustrates a definition of new interaction interface as specified in Section 6.2 and its usage in object interaction metadata.

The fourth example, called ‘Eye’, uses a sample 3D object from the medical domain to present another interaction metadata feature – ability to link interaction interface parameter types and semantic interaction description with domain specific ontology.

The last, but not least, the fifth example, the ‘Sliding Door’, illustrates a construction and usage of complex action description (c.f. Section 5.5).

9.2. ‘Changing Ball’ example

Object and its context

Consider a 3D game where an object called ‘ball’ moves through a 3D space scattered with different objects. All objects besides the ‘ball’ are fixed in the 3D space and have various shapes. The ‘ball’ object is interactive. It changes its shape while passing by other objects. Player has a virtual equivalent of a tennis racket and its goal is to bounce the ‘ball’ from a starting area to an exit area. The ‘ball’ can enter the exit area only if it has appropriate shape. To obtain a shape required by the exit area the player has to maneuver the ‘ball’ to make it pass by an object which will cause the ‘ball’ to properly change its shape. Additional challenge is that the player does not know how the ‘ball’ reacts on objects with different shapes.

The game has three difficulty levels. The ‘easy’ level uses only objects with simple geometry (sphere, box, cone, torus, etc.) and unlimited match time. The ‘medium’ level uses objects with simple geometry, but the match has a time limit. Finally, the ‘hard’ level uses objects of arbitrary shapes and the ‘ball’ has to enter exit area below a specified time limit.

Such game can be implemented for example on Wii gaming console, where the Wii Remote controller could be used to control the racket and navigate through the 3D space.

Interaction and interaction metadata

In this example the *interactive object* (c.f. Section 5.1) is a ‘ball’, while the *trigger object* (c.f. Section 5.1) is one of other objects fixed in the 3D space of the game. The interactive object has only one interaction – ‘shape change’. The ‘Changing Ball’ game allows many different ‘ball’ objects, with various implementations of the ‘shape change’ interaction. In this example one of such interactions is analyzed. Consider a ‘ball’ object with the following interaction. The ‘ball’ has initially a shape of a *sphere*. When the ‘ball’ passes an object with a *cone* shape, it changes into a *box*. The ‘ball’ reacts on *cone* object only if it is closer than 50 virtual meters.

As mentioned in Section 6.1, the ‘ball’ object implements the Visual Interaction Interface (c.f. Section 6.3). This interface specifies types of all parameters required to describe the ‘shape change’ interaction, namely PositionCenter parameter type and ShapePrimitive parameter types used to describe shape of the trigger object and shape of the interacting object. The third component of the interaction, the required distance between the interacting object and the trigger object, is modeled as the 3D interaction space of the Event metadata element (c.f. Section 5.3). Full XML metadata document contains reference to interaction interface and interaction description (c.f Section 7.1), and has the following structure.

```
<Interactions
        xmlns:mim="pl.pue.dit.mim"
        xmlns:v-ii="pl.pue.dit.ii.v-ii"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:math="http://www.w3.org/1998/Math/MathML"
        xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
        xmlns:x3d="x3d"
        xsi:noNamespaceSchemaLocation=" interactions-schema.xsd">
        <InteractionInterface>pl.put.dit.ii.v-ii</InteractionInterface>
        <Interaction>
                <mim:Event>
                        …
                </mim:Event>
                <mim:Condition>
                        …
                </mim:Condition>
                <mim:Action>
                        …
                </mim:Action>
                <mim:Semantics>
                        …
                </mim:Semantics>
        </Interaction>
</Interactions>
```

Listing 9-1. XML document of 'changing ball' interactions metadata

The Event metadata element, apart from the definition of 3D interaction space mentioned above, holds information that the 'shape change' interaction is triggered when a PositionCenter parameter of the 'ball' or other object changes its value. XML description of the Event element is the following.

```
<mim:Event>
        <mim:InteractionSpace>
                <mim:Space>
                        <x3d:Sphere radius="50"/>
                </mim:Space>
                <mim:Semantics>
                        <mpeg7:KeywordAnnotation>
                                <mpeg7:Keyword>sphere</mpeg7:Keyword>
                                <mpeg7:Keyword>surroundings</mpeg7:Keyword>
                        </mpeg7:KeywordAnnotation>
                </mim:Semantics>
        </mim:InteractionSpace>
        <mim:Parameter typeID="V-II.PositionCenter"/>
</mim:Event>
```

Listing 9-2. Excerpt of XML metadata of the 'shape change' interaction: Event element

The metadata description excerpt presented above has the following interpretation. The 3D interaction space is defined as a sphere with radius 50. This means that the 'shape change' interaction may be executed only with objects positioned inside this sphere. As mentioned in Section 5.3, center of the sphere of the 3D interaction space is positioned in the center of the 'ball' object. Thus, to trigger an interaction both interacting objects have to be closer than 50. The 3D interaction space geometry definition is complemented by semantic description which uses MPEG-7 KeywordAnnotation (c.f. Section 3.2) to describe the 3D interaction space as ‘sphere’ and ‘surroundings’. The 3D interaction space definition is followed by specification of an object parameter type that triggers the interaction: the PositionCenter parameter type from Visual Interaction Interface. Such metadata description means that the ‘shape change’ interaction will be executed only in case of a change of PositionCenter parameter value of the interacting object or any other object inside the 3D interaction space.

Second interaction metadata element, the Condition, describes the fact that the ‘shape change’ interaction is executed only with objects that has a *cone* shape. Figure 9-3 contains an XML description of the Condition element.

```
<mim:Condition>
        <mim:Expression>
                <math:apply>
                        <math:eq/>
                        <math:csymbol>S</math:csymbol>
                        <math:cn>cone</math:cn>
                </math:apply>
        </mim:Expression>
        <mim:ExpressionArgument>
                <mim:ParameterValue typeID="V-II.ShapePrimitive" variableBinding="S"/>
        </mim:ExpressionArgument>
</mim:Condition>
```

Listing 9-3. Excerpt of XML metadata of the ‘shape change’ interaction: Condition element

The interaction metadata fragment presented above is interpreted in the following way. The ShapePrimitive parameter of the trigger object is bound to variable *S*. The variable is than used in MathML comparison expression. The expression checks if the *S* variable is equal *cone* value. The described interaction will be executed only if the expression returns *true*.

Last interaction metadata element, the Action, provides information about new geometry of the ‘ball’ object. Figure 9-4 illustrates XML metadata of the Action element.

```
<mim:Action>
        <mim:Parameter typeID="V-II.ShapePrimitive">
                <mim:Result>
                        <math:apply>
                                <math:cn>box</math:cn>
                        </math:apply>
                </mim:Result>
        </mim:Parameter>
        <mim:Semantics>
                <mpeg7:StructuredAnnotation>
                        <mpeg7:WhatObject>
```

```
                    <mpeg7:Name>shape</mpeg7:Name>
                </mpeg7:WhatObject>
                <mpeg7:WhatAction>
                    <mpeg7:Name>change</mpeg7:Name>
                </mpeg7:WhatAction>
                <mpeg7:How>
                    <mpeg7:Name>to box</mpeg7:Name>
                </mpeg7:How>
            </mpeg7:StructuredAnnotation>
        </mim:Semantics>
</mim:Action>
```

Listing 9-4. Excerpt of XML metadata of the 'shape change' interaction: Action element

The listing presented above is interpreted in the following way. The description specifies that the result of the interaction will be a change of ShapePrimitive parameter value of the interacting object. The new value is a result of MathML expression. The expression states that the new value of the mentioned parameter is *box*. The mathematical description of new parameter value is accompanied by a semantic description of the action. The semantic description uses MPEG-7 StructuredAnnotation (c.f. Section 3.2) to indicate that the subject of action is 'shape', the actual action is described as 'change', and the way how the action is performed is described by 'to box' keyword.

Interaction metadata usage

Consider a developer who wants to increase the set of 'ball' objects available in the new edition of the 'Changing Ball' game. Instead of creating new objects, he/she can search for already existing objects that were prepared for other applications but have the required 'shape change' interaction. The IMQL query for such search is the following:

```
FROM
        'http://www.freeObjects.org/'
WHERE
        event.paramtype.id == 'V-II.PositionCenter'
        AND
        condition.trigger.paramtype.id ~= '^V-II\.Shape.*'
        AND
        action.interacting.paramtype.id ~= '^V-II\.Shape.*'
```

Listing 9-5. Sample IMQL query: filter by parameter type

Such query will return URIs of all objects from a 'Free Objects' repository that have interaction, which reacts on object positions, requires a specific shape of the trigger object, and results in modified shape of the interacting object.

Similar approach can be used for building a special edition of the ‘Changing Ball’ game, for example a ‘Changing Aliens’ game. In this case, developer requires objects not only to implement ‘shape change’ interaction, but also to have restrictions concerning the resulting shape. The object should change into something similar to a UFO. Assuming that the ‘XYZ…’ is a value of MPEG-7 3D Shape descriptor representing a UFO, the IMQL query for such search is the following:

```
FROM
        ‘http://www.freeObjects.org/’
WHERE
        event.paramtype.id == ‘V-II.PositionCenter’
        AND
        condition.trigger.paramtype.id ~= ‘^V-II\.Shape.*’
        AND
        action.interacting.V-II.ShapeCustom.value.after = ‘XYZ…’
```

Listing 9-6. Sample IMQL query: filter by parameter value

The ‘Changing Ball’ game concept can act as a framework for rapid development of new variations of the original game idea. It is possible to define new game variants and difficulty levels by introducing ‘ball’ objects that react not only on shape, but also, for example, on color of other objects, or ‘ball’ objects that change shapes in sequences rather than in two way transformation (shape A to shape B and shape B to shape A). Consider a developer searching for new objects to be used as the ‘ball’ in the next edition of the ‘Changing Ball’ game. He/she wants to find all objects that initially are green, react on color of other objects and change their shape and their color. With additional restrictions that the 3D interaction space has to be spherical and the resulting color must match the color of the trigger object. The IMQL query for such search is the following:

```
FROM
        ‘http://www.freeObjects.org/’
WHERE
        event.paramtype.id == ‘V-II.PositionCenter’
        AND
        event.space.semantics = ‘sphere’
        AND
        condition.trigger.paramtype.id ~= ‘^V-II\.ApperanceColor.*’
        AND
        action.interacting.V-II.ApperanceColor.value.before == ‘0 1 0’
        AND
        action.interacting.paramtype.id ~= ‘^V-II\.Shape.*’
```

```
          AND
          action.interacting.V-II.ApperanceColor.value.after ==
                   action.trigger.V-II.ApperanceColor.value.before
```

Listing 9-7. Sample IMQL query: filter by many parameter types and values

According to Section 4.3 interaction metadata can be used together with MPEG-7 descriptions. Moreover, as mentioned in Section 8.4, the IMQL can be embedded in XQuery queries. This allows exploiting MPEG-7 and interaction metadata in a single query. Consider query, presented in Listing 9-7, extended with some restrictions concerning object creator and format. Such extended query formulated in XQuery language is the following.

```
LET $doc := doc('http://www.freeObjects.org/mpeg7.xml')
FOR
          $m IN $doc/Mpeg7/Description/MultimediaContent/Multimedia
WHERE
          $m//Creation/Creator/Agent[@xsi:type='OrganizationType']/Name = 'Boulder Co.'
          AND
          $m//MediaInformation/MediaProfile/MediaFormat/FileFormat/Name = 'mpeg-4'
          AND
          imql(
                   $m//Interactions,
                            event.paramtype.id == 'V-II.PositionCenter'
                            AND
                            event.space.semantics = 'sphere'
                            AND
                            condition.trigger.paramtype.id ~= '^V-II\.ApperanceColor.*'
                            AND
                            action.interacting.V-II.ApperanceColor.value.before == '0 1 0'
                            AND
                            action.interacting.paramtype.id ~= '^V-II\.Shape.*'
                            AND
                            action.interacting.V-II.ApperanceColor.value.after ==
                                     action.trigger.V-II.ApperanceColor.value.before
          )
RETURN $/MediaLocator/MediaURI
```

Listing 9-8. MPEG-7 metadata search: XQuery query with embedded IMQL query

Summary

The example presented above demonstrates that MIM enables search engines to find 3D objects with specific interaction properties. The structure of interaction metadata and employed interaction interfaces provide a base for rich queries and make similar scenarios feasible. Full interaction metadata of an example 'ball' object is presented in Appendix E.

9.3. ‘Ad Sign’ example

Object and its context

Consider 3D Virtual World which mimics the real world. Real people have their avatars, can roam the World freely and meet avatars of other people. It is possible to buy land, build houses and open shops. Shops can sell any possible virtual items, from clothes, to furniture, to cars and other vehicles. Such virtual replicas of the real world already exists, one of the most notable examples is “Second Life”. Consider a virtual clothes shop that wants to open its premises in the 3D Virtual World. To attract customers it will need an advertisement sign.

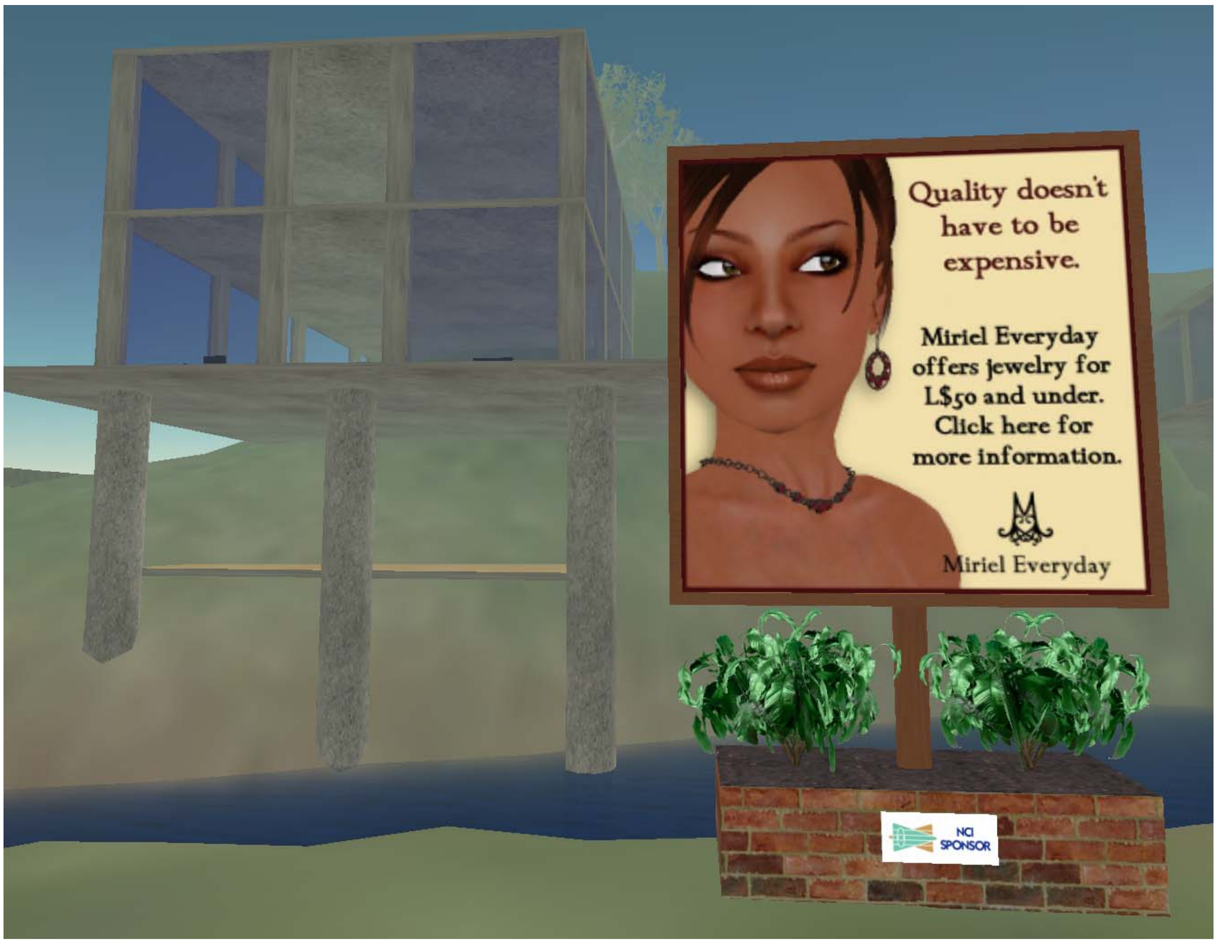
Quality doesn't have to be expensive.
Miriel Everyday offers jewelry for L$50 and under. Click here for more information.
Miriel Everyday
NCI SPONSOR

Figure 9-1. Advertisement sign example (screenshot from Second Life 3D world)

The sign has to be high enough to be seen from a distance. However, it also has to be small enough to be in sight of potential clients near the shop. It could be solved by two separate advertisement signs: one high and one small. However, instead of buying two advertisement billboards, the shop can achieve the same result with one interactive advertisement sign.

The 3D Virtual World has a property which makes such scenario possible. The World does not show the same advertisement sign to all avatars. Each avatar sees its own instance of the advertisement sign. This approach allows the interactive ad sign to adjust its height to a particular avatar.

Interaction and interaction metadata

In this example an *interactive object* (c.f. Section 5.1) is an advertisement sign. It has one interaction – ‘adjust height’. The height of the sign depends on the distance of the *trigger object* (c.f. Section 5.1), which, in this example, is an avatar. To trigger the interaction the avatar has to be in front of the advertisement sign. The ‘ad sign’ is high when the avatar is far away and it decreases its height when the avatar gets closer. On the other side, the ‘ad sign’ increases its height when the avatar moves away. The exact height of the sign depends on the height of the avatar.

As in Example 9.2, the ‘adjust height’ interaction is based on Visual Interaction Interface implemented by all interactive 3D objects (c.f. Sections 6.1 and 6.3). The parameters used in this interaction include: *BoundingBox*, *PositionCenter* and *ShapeCustom*. As defined in Section 6.3, all three parameters are complex. First two are taken from the X3D language (c.f. Section 3.4). They are composed of three numerical values, each describing different dimension of object bounding box or position. The third parameter is taken from the MPEG-7 standard (c.f. Section 2.5.4). It is composed of three complex values which form the MPEG-7 3D Shape descriptor. Full structure of the XML metadata document is the same as presented in Listing 9-1.

In the ‘adjust height’ interaction is executed for avatars that are situated in front of the ‘ad sign’. The interactive object reacts on change of the distance between interacting objects. Therefore, the Event metadata element contains information about the ‘ad sign’ viewpoint and about the parameter representing the distance between objects – *PositionCenter*. XML description of the Event element is the following.

```
<mim:Event>
        <mim:InteractionSpace>
                <mim:Viewpoint orientation="0 1 0 1.57" position="0 0 0"/>
        </mim:InteractionSpace>
        <mim:Parameter typeID="V-II.PositionCenter"/>
</mim:Event>
```

Listing 9-9. Excerpt of XML metadata of the ‘adjust height’ interaction: Event element

The XML presented above specifies the 3D interaction space of the Event as a viewpoint. As defined in Section 7.2.2 and Appendix B the viewpoint is described by a set of attributes taken from the X3D language. In this example, values of two attributes are specified: orientation and position, the remaining attributes use their default values. The value of the orientation attribute means that the viewpoint is directed along the ‘x’ axis. The position attribute value locates the viewpoint position in the center of the ‘ad sign’. The field of view of the viewpoint has a default value, which is equal to 0.785 radians (π/4).

The Condition metadata element contains an expression which verifies if the trigger object has a shape of an avatar. Listing 9-10 presents an XML representation of the Condition metadata element.

```
<mim:Condition>
        <mim:Expression>
                <math:apply>
                        <math:eq/>
                        <math:csymbol>S</math:csymbol>
                        <math:csymbol>avatar</math:csymbol>
                </math:apply>
        </mim:Expression>
        <mim:ExpressionArgument>
                <mim:ParameterValue typeID="V-II.ShapeCustom" variableBinding="S"/>
        </mim:ExpressionArgument>
        <mim:ExpressionArgument>
                <mim:ConstValue typeID="V-II.ShapeCustom" variableBinding="avatar">
                        <v-ii:Shape>
                                <mpeg7:Spectrum>12 130 148 22 301</mpeg7:Spectrum>
                                <mpeg7:PlanarSurfaces>0</mpeg7:PlanarSurfaces>
                                <mpeg7:SingularSurfaces>0</mpeg7:SingularSurfaces>
                        </v-ii:Shape>
                </mim:ConstValue>
        </mim:ExpressionArgument>
</mim:Condition>
```

Listing 9-10. Excerpt of XML metadata of the ‘adjust height’ interaction: Condition element

The metadata description excerpt presented above has the following interpretation. The Condition is composed of a MathML *Expression* element and two *ExpressionArgument* elements. The first *ExpressionArgument* is a value of the ShapeCustom parameter of the trigger object bound to the *S* variable. The second *ExpressionArgument* element defines a constant value called ‘avatar’. The value is a MPEG-7 3D Shape descriptor of an avatar. The Condition expression compares the shape of the trigger object with the predefined constant. The Condition is satisfied if the trigger object has a shape of an avatar.

The result of the ‘adjust height’ interaction, new height of the ad sign, is presented in the Action element. The element contains not only specification of modified parameter, but also mathematical expression which approximates new parameter value. The XML representation of the Action metadata element is the following.

```
<mim:Action>
        <mim:Parameter typeID="V-II.SizeBoundingBox.z">
                <mim:Result>
                        <math:apply>
                                <math:plus/>
                                <math:csymbol>AS.z</math:csymbol>
                                <math:apply>
                                        <math:times/>
                                        <math:csymbol>AS.z</math:csymbol>
                                        <math:apply>
                                                <math:ln/>
                                                <math:apply>
                                                        <math:root/>
                                                        <math:degree>
                                                                <math:cn>2</math:cn>
                                                        </math:degree>
                                                        <math:apply>
                                                                <math:plus/>
                                                                <math:apply>
                                                                        <math:power/>
                                                                        <math:csymbol>
                                                                                AP.x
                                                                        </math:csymbol>
                                                                        <math:cn>2</math:cn>
                                                                </math:apply>
                                                                <math:apply>
                                                                        <math:power/>
                                                                        <math:csymbol>
                                                                                AP.y
                                                                        </math:csymbol>
                                                                        <math:cn>2</math:cn>
                                                                </math:apply>
                                                        </math:apply>
                                                </math:apply>
                                        </math:apply>
                                </math:apply>
                        </math:apply>
                </mim:Result>
                <mim:ResultArgument>
                        <mim:ParameterValue
                                typeID="V-II.PositionCenter"
                                variableBinding="AP"
                                beforeOrAfter="after"/>
                </mim:ResultArgument>
                <mim:ResultArgument>
                        <mim:ParameterValue
                                typeID="V-II.SizeBoundingBox"
                                variableBinding="AS"
                                beforeOrAfter="after"/>
                </mim:ResultArgument>
        </mim:Parameter>
        <mim:Semantics>
                <mpeg7:FreeTextAnnotation>
                        Height modified depending on the avatar height and distance
                </mpeg7:FreeTextAnnotation>
        </mim:Semantics>
</mim:Action>
```

Listing 9-11. Excerpt of XML metadata of the 'adjust height' interaction: Action element

Listing 9-11 presents an XML code of the Action metadata element presents both, mathematical and semantic descriptions of the interaction result. The semantic description uses MPEG-7 *FreeTextAnnotation* (c.f. Section 3.2) to indicate that the interaction result is a modified value of the height of the ad sign. The mathematical description indicates the same fact, but also allows calculating the approximate new value of the modified height. The mathematical description consists of two *ResultArgument* elements and one *Result* element. First *ResultArgument* element binds the value of the *PositionCenter* parameter of the trigger object to the *AP* variable. Second *ResultArgument* binds the value of the *SizeBoundingBox* parameter of the trigger object to the *AS* variable. Both elements have a *beforeOrAfter* attribute set to 'after'. This indicates that the expression uses values after the change event that triggered the interaction. The *Result* element holds information on modified parameter and the actual mathematical expression. The modified parameter is the z component of the *SizeBoundingBox* parameter of the interacting object. The MathML expression used to calculate a new value of this parameter is composed of a natural logarithm of the distance between the ad sign and the avatar. The logarithm value is multiplied by and increased by the avatar height. Full mathematical expression is the following (*SS.z* stands for ad sign height):

$$SS.z = \left(\ln\left(\sqrt{AP.x^2 + AP.y^2}\right) * AS.z\right) + AS.z$$

Interaction metadata usage

Consider a user who wants to place an advertisement sign by his new virtual shop. He/she knows that to attract new customers the sign has to be visible from the main road that is quite far from the shop. To make it visible from the main road for an average avatar the sign should be around 8 meters high. Additionally, for an avatar standing near the shop the advertisement sign should be around 2 meters high. Therefore, he/she needs to find an interactive advertisement sign with appropriate height. Consider interactive 3D objects repository with objects described by MPEG-7 and MIM metadata. The XQuery query for an appropriate ad sign looks like this.

```
LET $doc := doc('http://www.virtual-objects.org/mpeg7.xml')
FOR
        $m IN $doc/Mpeg7/Description/MultimediaContent/Multimedia
WHERE
        (
                $m//KeywordAnnotation/Keyword = 'billboard'
                OR
                $m//KeywordAnnotation/Keyword = 'ad sign'
                OR
                $m//KeywordAnnotation/Keyword = 'banner
        )
        AND
        imql(
                $m//Interactions,
                        event.paramtype.id == 'V-II.PositionCenter'
                        AND
                        condition.trigger.paramtype.id ~= '^V-II\.Shape.*'
                        AND
                        action.trigger.paramtype.id =='V-II.PositionCenter'
                        AND
                        action.interacting.V-II.SizeBoundingBox.z.value.after <= 8
                        AND
                        action.interacting.V-II.SizeBoundingBox.z.value.after >= 2
        )
RETURN $/MediaLocator/MediaURI
```

Listing 9-12. Ad sign search: XQuery query with embedded IMQL query

The query presented above finds all objects described by keywords: 'billboard', or 'ad sign', or 'banner' and with specific interaction properties. The required interaction properties represent the 'adjust height' interaction with additional constraints. These constraints are expressed by two last conditions. Both conditions refer the same parameter and require a specific interaction result. The height of the ad sign has to be less or equal 8 and greater or equal 2.

Summary

The presented 'Ad Sign' example illustrates that MIM metadata enables search engines to perform precise searches that take into account object state after the interaction. This is not possible with existing metadata standards that are able to describe only a single state of an object. Full interaction metadata of the presented 'ad sign' object is available in Appendix E.

9.4. ‘Treasure Chest’ example

Object and its context

Consider a 3D Massively Multiuser Online Role Playing Game (MMORPG), where multiple users from all over the Internet can explore dark dungeons and fight with dragons. The MMORPG allows players to freely roam the virtual world, communicate with other players and exchange virtual items between players. Players are also allowed to create their own virtual items and load them into the game. Items that can be loaded into the game are divided into separate categories, for example: swords, helmets, bags, containers, potions, amulets, etc. Each category has some predefined attributes and usage methods. Execution of a usage method is translated into an interaction between a player and a particular item.

Consider a ‘treasure chest’ item which belongs to containers category. Items from this category have a *contents* attribute and two or four usage methods: *open*, *close* and optional *lock*, *unlock*. If the container item provides *lock*, and *unlock* usage methods, it has to be associated with another item which will act as a key. The key item can be a virtual key or another magic item and only a player with such key item in his/her *inventory* can open the container. In case of the ‘treasure chest’, all usage methods are available and the ‘treasure chest’ is associated with the ‘green key’ item.

Figure 9-2. „Treasure chest” item

Interaction and interaction metadata

The scenario presented above introduces one *interactive object* (c.f. Section 5.1) – the ‘treasure chest’ item. The ‘treasure chest’ has four interactions: ‘open’, ‘close’, ‘lock’, and ‘unlock’. Each of these interactions can be triggered by a *trigger object* (c.f. Section 5.1), which in this case is a player, or more precisely: player avatar. The execution of an interaction is based on a set of parameters. The avatar has to be near the ‘treasure chest’. Not far than 5 meters away. It has to *cast a spell* that identifies a specific interaction. Additionally, each interaction has its own requirements:

- The ‘open’ interaction will be executed only if the ‘treasure chest’ is *closed* and not *locked*.
- The ‘close’ interaction will be executed only it the ‘treasure chest’ is not *closed*.
- The ‘lock’ interaction will be executed only if the ‘treasure chest’ is *closed* and not *locked*, and the player has a ‘green key’ in his/her *inventory*.
- The ‘unlock’ interaction will be executed only if the ‘treasure chest’ is *closed* and *locked*, and the player has a ‘green key’ in his/her *inventory*.

Not all parameters required for ‘treasure chest’ interactions are included in the default Visual Interaction Interface (c.f. Section 6.3). The ‘treasure chest’ implements also a custom interaction interface built according to Interaction Interface Definition Rules specified in Section 6.2. The custom interaction interface is called Container Item Game Interaction Interface (CIG-II). Also, game avatars have to implement a custom interaction interface named Avatar Game Interaction Interface (AG-II). The CIG-II contains definitions of two parameters: *Closed* and *Locked*. Both parameters have a Boolean data type which allows only *true* or *false* value. The definition of CIG-II is presented in Listing 9-13.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<InteractionInterface
                xmlns="pl.pue.dit.ii"
                xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:schemaLocation="pl.pue.dit.ii ii-schema.xsd"
                ID="CIG-II"
                name="Container Item Game Interaction Interface">
        <ParameterType ID="Closed">
                <DataType>
                        <xs:simpleType>
                                <xs:restriction base="xs:boolean"/>
                        </xs:simpleType>
                </DataType>
                <Semantics>
                        <mpeg7:FreeTextAnnotation>
```

```
                                        Container closed state described by a boolean flag
                              </mpeg7:FreeTextAnnotation>
                    </Semantics>
          </ParameterType>
          <ParameterType ID="Locked">
                    <DataType>
                              <xs:simpleType>
                                        <xs:restriction base="xs:boolean"/>
                              </xs:simpleType>
                    </DataType>
                    <Semantics>
                              <mpeg7:FreeTextAnnotation>
                                        Container locked state described by a boolean flag
                              </mpeg7:FreeTextAnnotation>
                    </Semantics>
          </ParameterType>
          <Semantics>
                    <mpeg7:FreeTextAnnotation>Container Item Game Interaction Interface defines
parameters specific to interaction of container items in a hypothetical
MMORPG</mpeg7:FreeTextAnnotation>
          </Semantics>
</InteractionInterface>
```

Listing 9-13. XML definition of the Container Item Game Interaction Interface

The Avatar Game Interaction Interface is more complex. For the sake of this example the AG-II is reduced to parameters required by described interactions. This includes Spell and Inventory parameters. The Spell parameter provides a name of the spell cast by the player. Its data type is string. Second parameter, the Inventory, represents a list of items possessed by a player. The Inventory parameter has a complex data type: a set of strings. Each item in the inventory is represented by its name enclosed in single quotes.

Interaction metadata of all ‘treasure chest’ interactions are similar. The following is a description of the most complex ‘treasure chest’ interaction called ‘unlock’. The Event element of ‘unlock’ interaction metadata provides information that the trigger object has to be located not further than 5 meters of the ‘treasure chest’ and that the actual event is a change of a value of the Spell parameter. XML description of the Event metadata is the following.

```
<mim:Event>
          <mim:InteractionSpace>
                    <mim:Space>
                              <x3d:Sphere radius="5"/>
                    </mim:Space>
                    <mim:Semantics>
                              <mpeg7:FreeTextAnnotation>
                                        A circle of 5 meters around the chest
                              </mpeg7:FreeTextAnnotation>
                    </mim:Semantics>
```

```
            </mim:InteractionSpace>
            <mim:Parameter typeID="AG-II.Spell"/>
</mim:Event>
```

Listing 9-14. Excerpt of XML metadata of the 'open' interaction: Event element

The Condition metadata element contains expressions that verify if interacting object *Closed* and *Locked* parameters have *false* values, if the *Spell* parameter of the trigger object has 'unlock' value and if the avatar *Inventory* parameter contains a name of the required key item: 'green key'. XML representation of the Condition element is presented in Listing 9-15.

```
<mim:Condition>
            <mim:Expression>
                        <math:apply>
                                    <math:and/>
                                    <math:apply>
                                                <math:eq/>
                                                <math:csymbol>C</math:csymbol>
                                                <math:cn>true</math:cn>
                                    </math:apply>
                                    <math:apply>
                                                <math:eq/>
                                                <math:csymbol>L</math:csymbol>
                                                <math:cn>true</math:cn>
                                    </math:apply>
                                    <math:apply>
                                                <math:eq/>
                                                <math:csymbol>S</math:csymbol>
                                                <math:cn>unlock</math:cn>
                                    </math:apply>
                                    <math:apply>
                                                <math:in/>
                                                <math:cn>'green key'</math:cn>
                                                <math:csymbol>I</math:csymbol>
                                    </math:apply>
                        </math:apply>
            </mim:Expression>
            <mim:ExpressionArgument>
                        <mim:ParameterValue typeID="CIG-II.Closed" variableBinding="C"/>
            </mim:ExpressionArgument>
            <mim:ExpressionArgument>
                        <mim:ParameterValue typeID="CIG-II.Locked" variableBinding="L"/>
            </mim:ExpressionArgument>
            <mim:ExpressionArgument>
                        <mim:ParameterValue typeID="AG-II.Spell" variableBinding="S"/>
            </mim:ExpressionArgument>
            <mim:ExpressionArgument>
                        <mim:ParameterValue typeID="AG-II.Inventory" variableBinding="I"/>
            </mim:ExpressionArgument>
</mim:Condition>
```

Listing 9-15 Excerpt of XML metadata of the 'open' interaction: Condition element

The last MIM metadata element – Action – provides information that the result of the ‘unlock’ interaction is unlocked, but still closed ‘treasure chest’. The value of the *Locked* parameter of the ‘treasure chest’ changes from *true* to *false*. Listing 9-16 illustrates XML description of the Action metadata element.

```
<mim:Action>
        <mim:Parameter typeID="CIG-II.Locked">
                <mim:Result>
                        <math:apply>
                                <math:cn>false</math:cn>
                        </math:apply>
                </mim:Result>
        </mim:Parameter>
        <mim:Semantics>
                <mpeg7:FreeTextAnnotation>Unlock the chest</mpeg7:FreeTextAnnotation>
        </mim:Semantics>
</mim:Action>
```

Listing 9-16. Excerpt of XML metadata of the ‘open’ interaction: Action element

Interaction metadata usage

Consider a globally accessible Game Items Repository with objects created by other players. The repository acts as a massive marketplace, where players can sell, buy or exchange their items. Objects in the repository are described by their URI, name, owner, price, and interaction metadata. The repository is implemented with an RDBMS and therefore the search interface is limited to SQL.

Consider a player who wants to load a new chest into the game to hide his/her magic artifacts. Instead of designing and developing such chest from scratch he/she can use the repository to find an appropriate item. Assuming that a player wants to find a chest that costs below $30, the query is the following.

```
SELECT i.uri
FROM repository.game_items i
WHERE
        i.price < 30
        AND
        imql(
                i.interactions,
                ii.id == ‘CIG-II’
        )
```

Listing 9-17. Sample SQL query with embedded IMQL function

If the search results yields too many items, the player can redefine the query. For example, he/she can add a requirement that the chest should be lockable. The SQL query for such search is the following.

```
SELECT i.uri
FROM repository.game_items i
WHERE
        i.price < 30
        AND
        imql(
                i.interactions,
                ii.id == 'CIG-II'
                AND
                action.interaction.CIG-II.Locked.value.after == 'true'
        )
```

Listing 9-18. Sample SQL query with embedded IMQL function (extended)

Summary

The presented example illustrates how technically complex interactions can be easily described and queried with the use of custom interaction interfaces. The simplification of interaction metadata and interaction searches is possible due to additional semantic information introduced by a custom interaction interface. Full interaction metadata of the 'treasure chest' object is presented in Appendix E.

9.5. 'Eye' example

Object and its context

Consider a globally accessible repository of virtual models from the medical domain used for educational purposes. The repository contains interactive 3D models of almost any body part of humans and almost all animals living on Earth. Objects included in the repository come from educational centers located around the globe. When an educational center prepares a new model for use in their courses, the model is described with metadata and uploaded to the repository. Model metadata include mostly semantic descriptions of the model and the real object. Also interaction metadata of such model are based on semantic descriptions rather than mathematical expressions. Due to the fact that models in the repository come from different countries, model metadata are available in many different languages. To avoid confusion in translations of medical terms, metadata are linked with concepts of a medical ontology.

Consider an interactive 3D model of a human eye. The model is stored in the repository and described with metadata. It is used to teach anatomy of an eye and to demonstrate eye behavior in reaction to light.

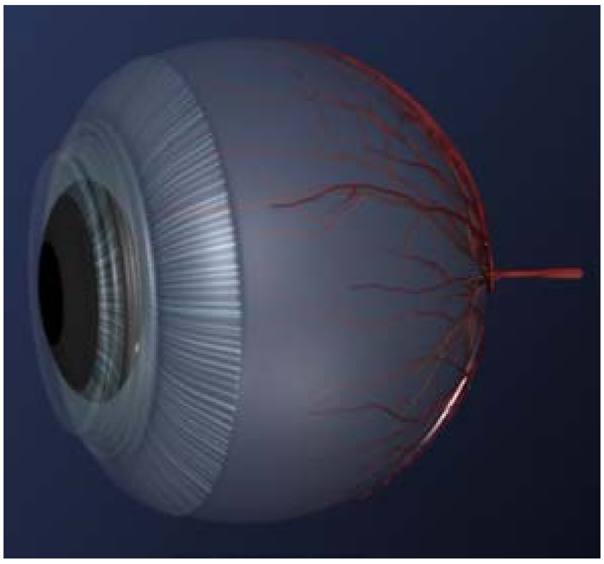

Figure 9-3. 3D model of a human eye

Interaction and interaction metadata

In this example, the *interactive object* (c.f. Section 5.1) is the 'eye model'. It has two interactions: 'dilate pupil' and 'constrict pupil'. Both interactions are triggered by light. The light may come from a specific object or from the environment (ambient light). Therefore, in this example, the *trigger object* (c.f. Section 5.1) is any object, including the *environment* (c.f. Section 5.1). First interaction, 'dilate pupil', is an increase of the size of the eye pupil in reaction to decreased light intensity. Second interaction, 'constrict pupil', is the exact opposite. It is a decrease of the size of the eye pupil in reaction to increased light intensity.

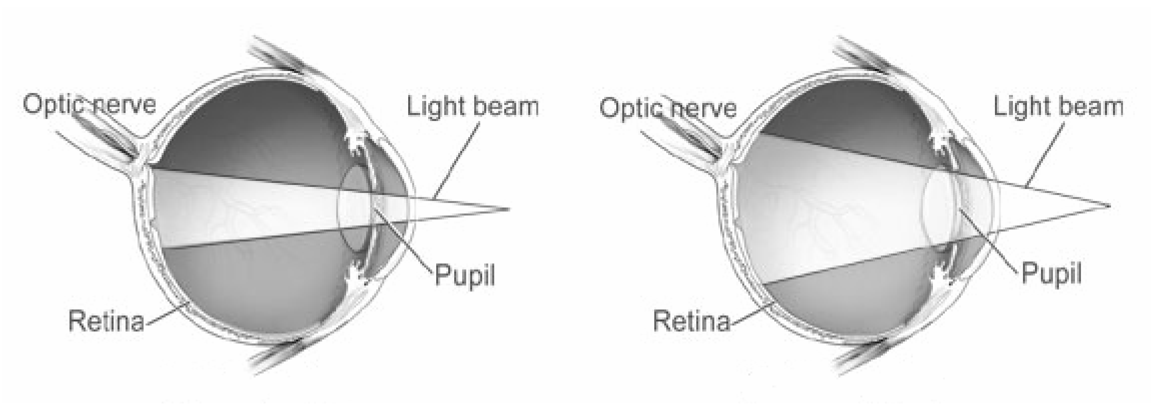

Figure 9-4. Eye diagram – constricted (left) and dilated (right) eye pupil

Both interactions are illustrated in Figure 9-4. The 'dilate pupil' interaction is a change from the left diagram to the right diagram. The 'constrict pupil' interaction is a change in the reverse direction.

Interactions of the 'eye model' cannot be described with parameters from the default Visual Interaction Interface. Parameters related with the 'eye model' are defined in a custom Eye Interaction Interface (E-II). The E-II interaction interface includes a number of parameters related with eye nerve signals, muscles, lens and pupil. Interactions of the 'eye model' described in this example use only one of them – *PupilSize*. The *PupilSize* parameter has a float data type and references the *eye:pupil:size* concept in the anatomy ontology of the Global Medicine Organization (*urn:gmo:anatomy*). Fragment of the Eye Interaction Interface with the *PupilSize* parameter definition is presented in Listing 9-19.

```
<?xml version="1.0" encoding="UTF-8"?>
<InteractionInterface
                xmlns="pl.pue.dit.ii"
                xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:schemaLocation="pl.pue.dit.ii ii-schema.xsd"
                ID="E-II"
                name="Eye Interaction Interface">
        …
        <ParameterType ID="PupilSize">
                <DataType>
                        <xs:simpleType>
                                <xs:restriction base="xs:float"/>
                        </xs:simpleType>
                </DataType>
                <Semantics>
                        <mpeg7:FreeTextAnnotation>Eye pupil size</mpeg7:FreeTextAnnotation>
                </Semantics>
                <Ontology ontology="urn:gmo:anatomy" concept="eye:pupil:size"/>
        </ParameterType>
        …
        <Semantics>
```

```
                    <mpeg7:FreeTextAnnotation>
                              Eye Interaction Interface include interaction parameters associated with eye
interactions.
                    </mpeg7:FreeTextAnnotation>
          </Semantics>
</InteractionInterface>
```

Listing 9-19. Excerpt of the Eye Interaction Interface definition

Parameters related with the light intensity and direction can be also defined by a custom interaction interface. However, this example uses different approach. As mentioned in Section 5.2, mathematical descriptions can be used in parallel to semantic descriptions. In the metadata of the 'eye model' interactions the light related information is expressed by semantic descriptions. Therefore, the custom interaction interface related to light properties is not necessary.

Metadata descriptions of both interactions of the 'eye model' are similar. The following description presents the first interaction called 'dilate pupil'. In this interaction, the Event metadata element has no interaction space definition and no specification of an exact parameter that triggers the interaction. As mentioned in Section 5.3, default interaction space equals the whole 3D Virtual World. The parameter specification is replaced by a semantic description. The XML representation of the Event element is the following.

```
<mim:Event>
          <mim:Semantics>
                    <mpeg7:StructuredAnnotation>
                              <mpeg7:WhatObject>
                                        <mpeg7:Term termID="urn:iso:physics:light:intensity"/>
                              </mpeg7:WhatObject>
                              <mpeg7:WhatAction>
                                        <mpeg7:Term termID="urn:iso:standardverbs:increase"/>
                              </mpeg7:WhatAction>
                    </mpeg7:StructuredAnnotation>
          </mim:Semantics>
</mim:Event>
```

Listing 9-20. Excerpt of XML metadata of the 'dilate pupil' interaction: Event element

The semantic description of the interaction event is composed of an MPEG-7 StructuredAnnotation, with descriptions of two elements: a subject (*WhatObject*) and a change (*WhatAction*). Both elements of StructuredAnnotation do not contain textual values, but URIs to appropriate ontology terms. The whole semantic description of the event states that the event occurs when light intensity increases.

As mentioned in Section 5.4, the Condition element is optional and can be omitted. In this example there is no Condition element. This means that the 'eye model' will react each time the light intensity increases.

The Action metadata element consists of a specification of modified parameter and semantic description of the new value of this parameter. Again, the semantic description is based on a relation to an appropriate ontology concept. In this case the referenced ontology is a dictionary of medical verbs and the concept is a 'constrict' verb. The XML representation of the Action element is the following.

```
<mim:Action>
        <mim:Parameter typeID="E-II.PupilSize"/>
        <mim:Semantics>
                <mpeg7:StructuredAnnotation>
                        <mpeg7:WhatAction>
                                <mpeg7:Term termID="urn:gmo:verbs:constrict"/>
                        </mpeg7:WhatAction>
                </mpeg7:StructuredAnnotation>
        </mim:Semantics>
</mim:Action>
```

Listing 9-21. Excerpt of XML metadata of the 'dilate pupil' interaction: Action element

The 'constrict pupil' interaction metadata contains also semantic description of the whole interaction. This semantic description is based on the MPEG-7 FreeTextAnnotation rather than on ontology relations. This increases information redundancy of interaction metadata and can be used by search engines to optimize queries. Excerpt of interaction metadata XML document showing semantic description of the whole interaction is presented in Listing 9-22.

```
<mim:Semantics xml:lang="en">
        <mpeg7:FreeTextAnnotation>Eye pupil constricts when light intensity
increases</mpeg7:FreeTextAnnotation>
</mim:Semantics>
```

Listing 9-22. Excerpt of XML metadata of the 'dilate pupil' interaction: semantic description

Interaction metadata usage

The repository of medical virtual models is accessible for any teacher around the world. Consider a teacher from Poland who wants to prepare a lecture on the eye anatomy and reactions to light. He/she has to search the repository for appropriate models. Assuming that objects are described with RDF, the query has to be formulated in SPARQL with embedded IMQL function (c.f. Section 8.4). The SPARQL query for an eye model with dilate and constrict interactions is the following.

```
PREFIX models:                <http://repository.gmo.org/ >
SELECT ?modelURI
FROM <http://repository.gmo.org/metadata.owl>
WHERE {
              models:object models:name “eye” .
              models:object models:URI ?modelURI .
              models:object models:interactions ?interactions .
              FILTER imql(?interactions,
                     event.semantics = ‘light intensity change’
                     AND
                     action.interacting.paramtype.id = ‘E-II.PupilSize’
                     AND
                     action.semantics = ‘pupil size change’
              )
       }
```

Listing 9-23. Sample SPARQL query with embedded IMQL

Due to the fact that interaction metadata semantic descriptions are linked with ontology terms, the teacher can achieve the same results without knowing exact interaction parameters. For example, the query presented in Listing 9-23 can be simplified in the following way.

```
PREFIX models:                <http://repository.gmo.org/ >
SELECT ?modelURI
FROM <http://repository.gmo.org/metadata.owl>
WHERE {
              models:object models:name “eye” .
              models:object models:URI ?modelURI .
              models:object models:interactions ?interactions .
              FILTER imql(?interactions,
                     interaction.semantics = ‘pupil size change at changing light intensity’
              )
       }
```

Listing 9-24. Simplified version of the query presented in Listing 9-23

Consider a global medical ontology containing medical concepts available in many languages. In such case the teacher can formulate the query using his/her own language. Listing 9-25 presents the same query written in Polish language.

```
PREFIX models:                    <http://repository.gmo.org/ >
SELECT ?modelURI
FROM <http://repository.gmo.org/metadata.owl>
WHERE {
                models:object models:name “oko” .
                models:object models:URI ?modelURI .
                models:object models:interactions ?interactions .
                FILTER imql(?interactions,
                        interaction.semantics = ‘zmiana rozmiaru źrenicy przy zmianie intensywności
światła’
                )
        }
```

Listing 9-25. Polish version of the query presented in Listing 9-24

Polish terms are compared to available ontology concepts. If there is a match, all translations, generalizations and specializations of a concept will be taken into account by the search engine.

Summary

The ‘Eye’ example demonstrates how interaction interface definitions and MIM semantic descriptions related to appropriate ontology may simplify queries formulation. Queries can be constructed without the knowledge about specific interaction parameters or interaction interfaces. Full interactions metadata for this example object is presented in Appendix E.

9.6. ‘Sliding Door’ example

Object and its context

Consider an architecture studio where engineers work to design new buildings. Each building design is composed of many components, like for example walls, windows, doors, lifts, stairs, etc. All these components are virtual models of real objects that will be used during the development of a new building. To allow designers running some simulations of the building, virtual models are almost exact copies of real objects, including their physical characteristics and their behavior. Such virtual models are usually provided by manufacturers of the real objects. Therefore, architects can search for an object on the web or directly in manufacturer repositories.

Not all architectural objects have a behavior. Example of an interactive architectural object is a model of automatic ‘sliding door’. The ‘sliding door’ *opens* itself when someone approaches the door and *closes* itself automatically. It can be *locked* in open or closed state, which means that the door can be permanently closed or permanently opened. The actions of door opening and closing are visualized as animations and complemented with appropriate sound. The diagram of such object is presented in Figure 9-5.

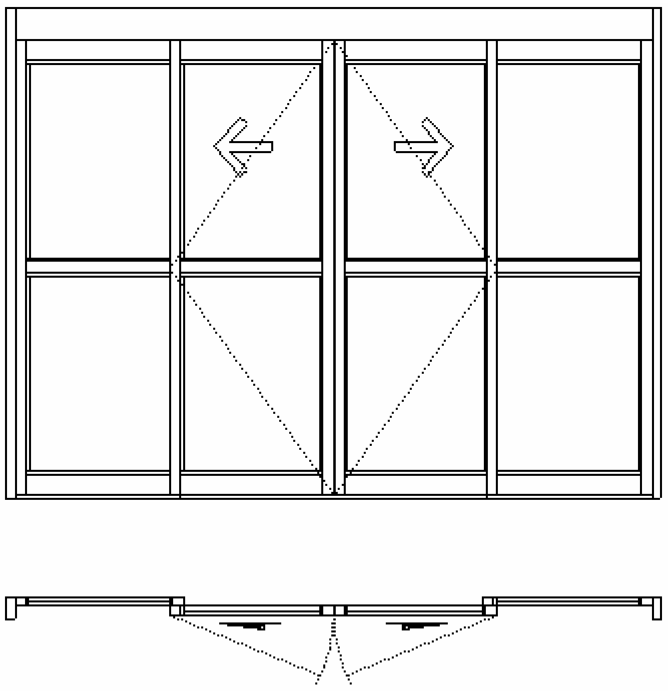

Figure 9-5. 'Sliding door' diagram

Interaction and interaction metadata

In this example, the *interactive object* (c.f. Section 5.1) is a 'sliding door' model. The 'sliding door' has two interactions: 'open' and 'close'. First interaction is triggered by an object moving in the area monitored by door motion sensor. Second interaction, 'close', is triggered by an object leaving the monitored area. The *trigger object* (c.f. Section 5.1) of both interactions can be any object big enough to exceed the sensitivity limit of the door motion sensor. The 'sliding door' can also be *locked* in the closed or opened state. The object does not provide any interaction for changing the *locked* parameter.

Parameters used in the 'open' and 'close' interactions include *PositionCenter* and *SizeBoudingBox* from the Visual Interaction Interface (c.f. Section 6.3) and *Closed* and *Locked* parameters from a custom interaction interface called ArchItems – Door Interaction Interface (AIdoor-II). The AIdoor-II interaction interface defines the *Closed* and *Locked* parameters as binary flags with Boolean data type.

Both interactions of the ‘sliding door’ object are similar. This example describes in detail the first interaction – ‘open’. The Event metadata element of the ‘open’ interaction contains specification of a parameter which value change triggers the event, and definition of the 3D interaction space. The event of the ‘open’ interaction it triggered by a change of the *PositionCenter* parameter of an object located inside the 3D interaction space. The 3D interaction space has a shape of a cone with the apex located above the door and the vertical axis directed down at an angle of 45^o. Excerpt of XML document with Event element description is presented in Listing 9-26.

```
<mim:Event>
        <mim:InteractionSpace>
                <mim:Viewpoint position="0 1.25 0" fieldOfView="0.785398" orientation="1 0 0 -
2.355"/>
                <mim:Semantics>
                        <mpeg7:FreeTextAnnotation>
                                Door motion sensor monitoring range
                        </mpeg7:FreeTextAnnotation>
                </mim:Semantics>
        </mim:InteractionSpace>
        <mim:Parameter typeID="V-II.PositionCenter"/>
</mim:Event>
```

Listing 9-26. Excerpt of XML metadata of the ‘open’ interaction: Event element

The Condition element of ‘open’ interaction metadata contains a mathematical expression which checks if the size of the trigger object is above the threshold required to execute the interaction. Other conditions that have to be met to execute the ‘open’ interaction refer to the state of the ‘sliding door’. The door has to be *closed* and *not locked*. XML representation of the Condition metadata element is the following.

```
<mim:Condition>
        <mim:Expression>
                <math:apply>
                        <math:and/>
                        <math:apply>
                                <math:eq/>
                                <math:csymbol>C</math:csymbol>
                                <math:cn>true</math:cn>
                        </math:apply>
                        <math:apply>
                                <math:eq/>
                                <math:csymbol>L</math:csymbol>
                                <math:cn>false</math:cn>
                        </math:apply>
                        <math:apply>
                                <math:geq/>
```

```
                                        <math:apply>
                                                <math:times/>
                                                <math:csymbol>S.x</math:csymbol>
                                                <math:csymbol>S.y</math:csymbol>
                                                <math:csymbol>S.z</math:csymbol>
                                        </math:apply>
                                        <math:cn>0.82</math:cn>
                                </math:apply>
                        </math:apply>
                </mim:Expression>
                <mim:ExpressionArgument>
                        <mim:ParameterValue typeID="AIdoor-II.Closed" variableBinding="C"/>
                </mim:ExpressionArgument>
                <mim:ExpressionArgument>
                        <mim:ParameterValue typeID="AIdoor-II.Closed" variableBinding="L"/>
                </mim:ExpressionArgument>
                <mim:ExpressionArgument>
                        <mim:ParameterValue typeID="V-II.SizeBoundingBox" variableBinding="S"/>
                </mim:ExpressionArgument>
        </mim:Condition>
```

Listing 9-27. Excerpt of XML metadata of the 'open' interaction: Condition element

As mentioned in Section 5.5, the description of the MIM Action element can be composed of an arbitrary number of sub-actions. The interaction metadata of 'open' interaction uses such complex structure. It consists of descriptions of two parallel actions. First action is an animation of door opening. Second action is a sound that occurs when the door is opening. The animation action is described by a semantic description. The action related with sound includes sound characteristics described by value of parameter taken from a custom Aural Interaction Interface. Listing 9-28 illustrates the XML representation of the Action element.

```
<mim:Action>
        <mim:SubAction>
                <mim:Parameter typeID="A-II.PlayedSound">
                        <mim:Result
                                        xsi:type="mpeg7:AudioSpectrumBasisType"
                                        mpeg7:loEdge="62.5"
                                        mpeg7:hiEdge="8000"
                                        mpeg7:resolution="1/4 octave">
                                <mpeg7:BasisFunctions>
                                        <mpeg7:Matrix dim="10 5">
                                                0.26 -0.05 0.01 -0.70 0.44
                                                0.34 0.09 0.21 -0.42 -0.05
                                                0.33 0.85 0.24 -0.05 -0.39
                                                0.33 0.85 0.24 -0.05 -0.39
                                                0.27 0.83 0.16 0.24 -0.04
                                                0.27 0.83 0.16 0.24 -0.04
                                                0.23 0.83 0.09 0.27 0.24
                                                0.20 0.83 0.04 0.22 0.40
                                                0.17 0.81 0.01 0.14 0.37
                                                0.33 -0.15 0.24 0.05 0.39
                                        </mpeg7:Matrix>
                                </mpeg7:BasisFunctions>
                        </mim:Result>
                </mim:Parameter>
        </mim:SubAction>
        <mim:SubAction>
                <mim:Semantics>
                        <mpeg7:FreeTextAnnotation>
                                Door sides slide slowly back to the center.
                        </mpeg7:FreeTextAnnotation>
                </mim:Semantics>
        </mim:SubAction>
</mim:Action>
```

Listing 9-28. Excerpt of XML metadata of the 'open' interaction: Action element

Interaction metadata usage

Consider a repository of interactive 3D models of door manufactured by a certain company. Each object consists of a 3D geometry, embedded behavior, and MPEG-7 metadata with embedded MIM metadata. Such repository is accessible to architects designing new buildings. Architect specifies door properties that are required by his/her design and looks for doors that match these properties. The query for any automatic door is the following.

```
LET $doc := doc('http://www.kampin.com/doors.xml')
FOR
        $m IN $doc/Mpeg7/Description/MultimediaContent/Multimedia
WHERE
        (
                $m//KeywordAnnotation/Keyword = 'door'
                OR
                $m//KeywordAnnotation/Keyword = 'automatic'
        )
        AND
        imql(
                $m//Interactions,
                        event.paramtype.id == 'V-II.PositionCenter'
        )
RETURN $/MediaLocator/MediaURI
```

Listing 9-29. Sample XQuery query with embedded IMQL query

The architect can also specify requirements on the area of the motion sensor. For example a set of points may be specified that have to be included in the sensor area. As mentioned in Section 8.3, points have to be specified in the x3d SFVector3f format. For example: furthest point on the ground (0 0 -5) and at half of the door height (0 1 -3), and furthest points located to the sides of the door (-1 1 -1) and (1 1 -1). The query with such requirement is the following.

```
LET $doc := doc('http://www.kampin.com/doors.xml')
FOR
        $m IN $doc/Mpeg7/Description/MultimediaContent/Multimedia
WHERE
        (
                $m//KeywordAnnotation/Keyword = 'door'
                OR
                $m//KeywordAnnotation/Keyword = 'automatic'
        )
        AND
        imql(
                $m//Interactions,
                        event.paramtype.id == 'V-II.PositionCenter'
                        AND
                        '0 0 -5' >> event.space
                        AND
                        '0 1 -3' >> event.space
                        AND
                        '-1 1 -1' >> event.space
                        AND
                        '1 1 -1' >> event.space
        )
RETURN $/MediaLocator/MediaURI
```

Listing 9-30. Query presented in Listing 9-29 extended with 3D interaction space requirement

Moreover, also detailed action properties can be included in the query. Assuming that an architect wants to find all sliding doors with a specific opening sound, the query is the following.

```
LET $doc := doc('http://www.kampin.com/doors.xml')
FOR
        $m IN $doc/Mpeg7/Description/MultimediaContent/Multimedia
WHERE
        (
                $m//KeywordAnnotation/Keyword = 'door'
                OR
                $m//KeywordAnnotation/Keyword = 'sliding'
                OR
                $m//KeywordAnnotation/Keyword = 'automatic'

        )
        AND
        imql(
                $m//Interactions,
                        event.paramtype.id == 'V-II.PositionCenter'
                        AND
                        action.semantics = 'slide'
                        AND
                        action.interactive.paramtype.A-II.PlayedSound..after.value =
                                '<value
```

```
                                        xsi:type="mpeg7:AudioSpectrumBasisType"
                                        mpeg7:loEdge="62.5"
                                        mpeg7:hiEdge="8000"
                                        mpeg7:resolution="1/4 octave">
                                    <mpeg7:BasisFunctions>
                                        <mpeg7:Matrix dim="5 5">
                                            0.26 -0.05 0.01 -0.70 0.44
                                            0.33 0.85 0.24 -0.05 -0.39
                                            0.27 0.83 0.16 0.24 -0.04
                                            0.23 0.83 0.09 0.27 0.24
                                            0.17 0.81 0.01 0.14 0.37
                                        </mpeg7:Matrix>
                                    </mpeg7:BasisFunctions>
                                </value>'
    )
RETURN $/MediaLocator/MediaURI
```

Listing 9-31. Query presented in Listing 9-29 extended with sub-action requirements

Summary

The example presented above illustrates that metadata with complex descriptions of MIM Action element enable users to look for interactive objects based on every detail of an interaction. Full interactions metadata for the 'Sliding Door' example is presented in Appendix E.

10. Conclusions

The MIM concept presented in this dissertation enables search engines to find interactive 3D objects with specific interaction properties. The MIM concept provides a solution for creating and using metadata of interactions of 3D objects. With tools offered by the MIM existing metadata standards can be extended and used not only to describe general semantics, production, technical, and management information, or object geometry, but also to describe semantics of interactions and to specify interaction parameters.

Three new tools are employed to achieve this goal: definition rules for interaction interfaces, interaction model and its metadata description schema, and a query sub-language tailored to exploit information embedded in interaction interfaces and in interaction metadata. All these tools tied together form a solution that covers all aspects of composition and exploitation of interaction metadata.

Due to the approach proposed in the MIM concept, interaction metadata not only describe a new set of object properties, but also enable analyzing all possible states of an object. Existing metadata standards focused mostly on general semantics and classification, allow creating descriptions of a single state of an object. Full description of an interactive object would require descriptions of all possible states of this object. With the MIM concept it is not necessary. MIM based metadata describe how the object changes from one state to another. This allows creating interaction metadata that complement general descriptions provided by existing metadata standards. The MIM solution allows for detailed semantic descriptions of object parameters involved in the interaction. It also provides metadata tools for describing what can trigger an interaction, what context is required and what is the result of the interaction.

The main achievements of this dissertation are the following:

1. Evaluation of existing metadata standards for multimedia objects and analysis of their potential to describe interactions (c.f. Sections 2 and 3).
2. Definition of the Multimedia Interaction Model (MIM) based on the *event – condition – action* concept, which allows decomposition of interaction into smaller components. Such decomposition facilitates creation of interaction descriptions and provides interaction components structure that can be used for metadata analysis (c.f. Sections 4 and 5).

3. Formulation of the Interaction Interface (II) concept and development of Interaction Interface Definition Rules. The II concept allows treating interactive objects in a way similar to programming objects, with a clearly defined set of methods and attributes. Due to such approach, interaction descriptions are related to parameters with known semantics which improves effectiveness of interactive objects searching (c.f. Section 6).
4. Identification of common 3D visual interaction parameters and definition of Visual Interaction Interface, which acts a base for any 3D interaction (c.f. Section 6.3).
5. Development of XML Schemas for interaction metadata. The set of XML Schemas include Interaction Interface schema, Multimedia Interaction Model schema and overall Interactions schema which ties the previous two together (c.f. Section 7).
6. Development of a new query sub-language – called Interaction Metadata Query sub-Language (IMQL) – that was built to fully exploit the interaction metadata (c.f. Section 8).
7. Definition of rules for embedding the IMQL in other query languages like XQuery, SPARQL or SQL (c.f. Section 8.4).
8. Elaboration of possible usage scenarios of interaction metadata demonstrating features of the MIM concept and proving usefulness of the proposed approach (c.f. Section 9).

The MIM solution is characterized by two important features: extensibility and independence.

Due to inclusion of Interaction Interface Definition Rules and associated XML Schemas, new interaction interfaces – specialized for particular domains or applications – can be easily created and used in interaction metadata. Such extensibility was one of the major requirements for the Interaction Interface concept and makes the MIM solution adaptable to different application requirements.

Independence of the MIM solution was achieved due to appropriate definitions of the MIM model and the IMQL language. Both elements are self-contained and provide general embedding rules. Therefore, interaction metadata can be embedded and used with any metadata standard, which allows easy adaptation to different applications or new metadata standards used for general object description.

The domain of interaction metadata is an emerging research area. The MIM model may be a starting point for several research initiatives. The most natural area of research is development of new search engines focused on object interaction properties. Another research area is automatic extraction of interaction properties from the object, which enables automatic metadata creation. Research on automatic metadata extraction is flourishing in areas of image, audio and video analysis. Therefore, it is likely that such works will cover also interaction properties. Apart from both mentioned research activities, the development of new interactive 3D applications will drive the increase of a number of repositories of interactive 3D objects from different domains. This can trigger additional research initiative focused on building globally accessible repositories and discovery services for interaction interfaces for different domains.

Bibliography

[1] 3D Model Search Engine, Princeton Shape Retrieval and Analysis Group
`http://shape.cs.princeton.edu/search.html`

[2] AAF Specifications
`http://www.aafassociation.org/html/techinfo/index.html#aaf_specifications`

[3] Adobe Systems Incorporated
`http://www.adobe.com/`

[4] Advanced Media Workflow Association (AMWA), formerly AAF Association
`http://www.aafassociation.org/`

[5] Alashqur A. M., Su S., Lam H., OQL: A Query Language for Manipulating Object-oriented Databases, Proceedings of the Fifteenth International Conference on Very Large Data Bases, August 22-25, 1989, Amsterdam, The Netherlands, pp: 433-442

[6] ARCO Consortium, *Report on the XML descriptions of the database of cultural objects*, ARCO Project public report, 2002
`http://www.arco-web.org/publicDocuments/D8/ARCO-D8-1.0-R-251002.pdf`

[7] Asperti A., Padovani L., Sacerdoti Coen C., Schena I., *Formal Mathematics in MathML*, Proceedings of the first International Conference on MathML and Math on the Web (MathML 2000), October 20–21, 2000, Urbana-Champaign, IL USA

[8] Beckett D., Turtle - Terse RDF Triple Language
`http://www.dajobe.org/2004/01/turtle/`, 2006

[9] Bilasco I.M., Gensel J., Villanova-Oliver M., Martin H., *On indexing of 3D scenes using MPEG-7*, Proceedings of ACM Multimedia 2005, Singapore, 2005, pp: 471-474

[10] Bilasco I.M., Gensel J., Villanova-Oliver M., Martin H., *An MPEG-7 framework enhancing the reuse of 3D models*, Proceedings of Web3D Symposium 2006, Columbia, Maryland USA, 2006, pp: 65-74

[11] Broekstra J., Kampman A., SeRQL: A Second Generation RDF Query Language, SWAD-Europe Workshop on Semantic Web Storage and Retrieval, 13-14 November 2003, Vrije Universiteit, Amsterdam, Netherlands

[12] Bry F. Patrânjan P., *Reactivity on the web: paradigms and applications of the language XChange*, Proceedings of the 2005 ACM Symposium on Applied Computing, Santa Fe, New Mexico USA, 2005, pp: 1645-1649

[13] Chamberlin D., Robie J., Florescu D., *Quilt: An XML Query Language for Heterogeneous Data Sources*, Proceedings of WebDB 2000 Conference, Lecture Notes in Computer Science, Springer-Verlag, 2000

[14] CIMI XML Schema for SPECTRUM, 2002
`http://xml.coverpages.org/CIMIv15-Schema.html`

[15] Croquet, The Croquet Consortium
`http://www.opencroquet.org/`

[16] Dayal U. et al., *HiPAC: a Research Project in Active, Time-Constrained Database Management, Interim Report*, Technical Report XAIT-88-02, Xerox Advanced Information Technology, June 1988

[17] Declerck T., Busemann S., Rehatschek H., Kienast G., *Annotating text using the Linguistic Description Scheme of MPEG-7: The DIRECT-INFO Scenario*, Proceedings of NLPXML-2006: Multi-dimensional Markup in Natural Language Processing, 2006

[18] Declerck T., Vela M., *Linguistic Dependencies as a Basis for the Extraction of Semantic Relations*, Proceedings of the ECCB'05 Workshop on Bio-medical Ontologies and Text Processing, Madrid 2005

[19] Digital Imaging Group, DIG35 Specification, Metadata for Digital Images 1.0, 2000
`http://xml.coverpages.org/FU-Berlin-DIG35-v10-Sept00.pdf`

[20] Dublin Core Metadata Element Set, Version 1.1
`http://dublincore.org/documents/dces/`

[21] Dublin Core Metadata Initiative
`http://dublincore.org/`

[22] Dublin Core Workshop DC-1: OCLC/NCSA Metadata Workshop: The Essential Elements of Network Object Description
`http://dublincore.org/workshops/dc1/`

[23] European Broadcasting Union (EBU)
`http://www.ebu.ch/`

[24] EBU Metadata Exchange Scheme (EBU P/Meta) 1.2, EBU Tech 3295
`http://www.ebu.ch/CMSimages/en/`
`tec_doc_t3295_v0102_tcm6-40957.pdf`

[25] ETSI TS 102 822-1 *Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime Phase 1")*, 2004

[26] Extensible Markup Language (XML) 1.0, W3C Recommendation 16 August 2006
`http://www.w3.org/TR/xml/`

[27] Funkhouser, T., Min, P., Kazhdan, M., Chen, J., Halderman, A., Dobkin, D., Jacobs, D., *A search engine for 3D models*, ACM Trans. Graph., Volume 22, Issue 1, January 2003, pp: 83-105

[28] Goczyła K., Grabowska T., Waloszek W., Zawadzki M., *The cartographer algorithm for processing and querying description logics ontologies*, Proceedings of the Third International Atlantic Web Intelligence Conference, Advances in Web Intelligence AWIC 2005, Lodz, Poland, 6-9 June 2005, pp. 163-169

[29] Google Earth, Google
`http://earth.google.com/`

[30] Hausenblas M. et al, *Multimedia Vocabularies on the Semantic Web*, W3C Incubator Group Report 19 April 2007
`http://www.w3.org/2005/Incubator/mmsem/XGR-vocabularies/`

[31] Herrera P., Serra X., Peeters G., Audio Descriptors and Descriptors Schemes in the Context of MPEG-7, Proceedings of ICMC, 1999

[32] ID3 Informal standard, ID3 tag version 2.4.0
`http://www.id3.org/id3v2.4.0-frames`

[33] IPTC, The International Press and Telecommunications Council
`http://www.iptc.org/`

[34] IPTC Metadata for XMP
`http://www.iptc.org/IPTC4XMP/`

[35] ISO International Standard, ISO 9075:2003, Information technology – Database languages – SQL

[36] ISO International Standards, ISO/IEC 11179 , Information Technology – Metadata Registries (MDR)
`http://www.metadata-stds.org/`

[37] ISO International Standards, ISO/IEC 14772-1:1997 and ISO/IEC 14772-2:2004 – Virtual Reality Modeling Language (VRML)
`http://www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-VRML97/`

[38] ISO International Standard, ISO 15836:2003, Information and documentation – The Dublin Core metadata element set
`http://www.niso.org/international/SC4/n515.pdf`

[39] ISO International Standard, ISO/IEC 15938, Information technology – Multimedia content description interface
`http://www.iso.org/iso/en/prods-services/popstds/mpeg.html`

[40] ISO International Standard, ISO/IEC 19775:2004 – Extensible 3D (X3D)
`http://www.web3d.org/x3d/specifications/ISO-IEC-19775-X3DAbstractSpecification/`

[41] Japan Electronics and Information Technology Industries Association (JEITA) Standard, JEITA CP-3451, *Exchangeable image file format for digital still cameras: Exif Version 2.2*
`http://www.exif.org/Exif2-2.PDF`

[42] Knuth D.E., *Computers & Typesetting, Volume A: The TeXbook*, Addison Wesley Professional, 1986

[43] Kruijff GJM., Duchier D., *Information Structure in Topological Dependency Grammar*, Proceedings EACL, 2003

[44] Kruijff GJM., *Formal & Computational Aspects of Dependency Grammar*, ESSLLI-2002

[45] Lamport L., *LaTeX: A Document Preparation System*, Addison Wesley Professional, 1994

[46] LaTeX project website
`http://www.latex-project.org/`

[47] LaTeX Wikibook
`http://en.wikibooks.org/wiki/LaTeX`

[48] Lorenz B., Ohlbach H.J., Stoffel E.P., *A Hybrid Spatial Model for Representing Indoor Environments*, Proceedings of W2GIS 2006, (LNCS 4295), Hong Kong, China, 2006, pp: 102-112

[49] Martinez J.M., *MPEG-7: Overview of MPEG-7 Description Tools, part 2*, IEEE Multimedia, Volume 9, Issue 3, Jul-Sep 2002, pp: 83- 93

[50] MetaBrainz Foundation
`http://metabrainz.org/`

[51] Mathematical Markup Language (MathML) Version 2.0, World Wide Web Consortium Recommendation, 2001
`http://www.w3.org/TR/MathML2/`

[52] MODS, Metadata Object Description Schema, Library of Congress' Network Development and MARC Standards Office
`http://www.loc.gov/standards/mods/`

[53] Mourkoussis N., White M., Patel M., Chmielewski J., Walczak K., *AMS - Metadata for Cultural Exhibitions using Virtual Reality*, Proceedings of Dublin Core International Conference DC 2003, Seattle, Washington (USA) ; September 28 - October 2, 2003; pp. 193-20

[54] Moving Picture Experts Group (MPEG)
`http://www.chiariglione.org/mpeg/`

[55] MPEG-4 Overview
`http://www.chiariglione.org/mpeg/standards/mpeg-4/`
`mpeg-4.htm`

[56] MPEG-7 Overview
`http://www.chiariglione.org/MPEG/standards/mpeg-7/`
`mpeg-7.htm`

[57] MusicBrainz project
`http://musicbrainz.org/doc/MusicBrainz`

[58] NISO American National Standard, ANSI/NISO Z39.85-2001, *The Dublin Core Metadata Element Set*
`http://www.niso.org/standards/resources/Z39-85.pdf`

[59] NISO American National Standard, ANSI/NISO Z39.87-2006, Data Dictionary – Technical Metadata for Digital Still Images
`http://www.niso.org/standards/resources/Z39-87-2006.pdf`

[60] Papamarkos G., Poulovassilis A., Wood PT., *Event-Condition-Action Rule Languages for the Semantic Web*, Workshop on Semantic Web and Databases, 2003

[61] Patel, M., White M., Mourkoussis N., Walczak K., Wojciechowski R., Chmielewski J., *Metadata Requirements for Digital Museum Environments*, International Journal on Digital Libraries, Special Issue on Digital Museum, Volume 5, Number 3, May/2005; Springer-Verlag; pp. 179-192

[62] Pitarello F., de Faveri A., *Semantic Description of 3D Environments: a Proposal Based on Web Standards*, Proceedings of Web3D Symposium 2006, Columbia, Maryland USA, 2006, pp: 85-95

[63] Project Looking Glass, Sun Microsystems
`http://www.sun.com/software/looking_glass/`

[64] Query Languages Workshop, Boston, USA, 1998
`http://www.w3.org/TandS/QL/QL98/`

[65] rdfDB : An RDF Database
`http://www.guha.com/rdfdb/`

[66] RDF Data Access Working Group, W3C Working Group
`http://www.w3.org/2003/12/swa/dawg-charter`

[67] RDQL - A Query Language for RDF, W3C Member Submission 9 January 2004
`http://www.w3.org/Submission/RDQL/`

[68] Resource Description Framework (RDF):Concepts and Abstract Syntax, W3C Recommendation 10 February 2004
`http://www.w3.org/TR/rdf-concepts/`

[69] RFC 2396, *Uniform Resource Identifiers (URI): Generic Syntax*
`http://www.ietf.org/rfc/rfc2396.txt`

[70] RFC 2732, *Format for Literal IPv6 Addresses in URL's*
`http://www.ietf.org/rfc/rfc2732.txt`

[71] Salembier P., Smith J.R., *MPEG-7 Multimedia Description Schemes*, IEEE Transactions on Circuits and Systems for Video Technology, Volume 11, Issue 6, June 2001, pp: 748-759

[72] Sandhu P., *The Mathml Handbook*, Charles River Media, 2002

[73] Skarbek W., Kukielka G, *Optimal intervals for fuzzy categories of color temperature with application to image browsing*, Machine Graphics & Vision International Journal, Volume 11, Issue 2/3, 2002, pp: 297-310

[74] Smith J.R., Schirling P., *Metadata standards roundup***,** IEEE Multimedia, Volume 13, Issue 2, April-June 2006, pp: 84-88

[75] SMPTE 377M-2004 Television Material Exchange Format (MXF) Standard – File Format Specification
`http://store.smpte.org/product-p/smpte%20377m-2004.htm`

[76] SMPTE 380M-2004 Television Material Exchange Format (MXF) Standard – Descriptive Metadata Scheme-1
`http://store.smpte.org/product-p/smpte%20380m-2004.htm`

[77] Society of Motion Picture and Television Engineers (SMPTE)
`http://www.smpte.org/`

[78] SPARQL Protocol for RDF, W3C Candidate Recommendation 6 April 2006
`http://www.w3.org/TR/rdf-sparql-protocol/`

[79] SPARQL Query Language for RDF, W3C Candidate Recommendation 14 June 2007
`http://www.w3.org/TR/rdf-sparql-query/`

[80] SPARQL Query Language for RDF, W3C Candidate Recommendation 14 June 2007, *Triple Patterns*
`http://www.w3.org/TR/rdf-sparql-query/#QSynTriples`

[81] SPARQL Query Language for RDF, W3C Candidate Recommendation 14 June 2007, *Filters*
`http://www.w3.org/TR/rdf-sparql-query/#scopeFilters`

[82] SPARQL Query Results XML Format, W3C Working Draft 14 June 2007 `http://www.w3.org/TR/rdf-sparql-XMLres/`

[83] SPECTRUM, Museum Documentation Association `http://www.mda.org.uk/stand.htm`

[84] Swartz A., *MusicBrainz: A Semantic Web Service*, 2001 `http://logicerror.com/musicbrainzArticle`

[85] Thome B., Gawlick D., Pratt M. *Event processing with an oracle database*, Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland USA, 2005, pp: 863-867

[86] Virtual Reality Modeling Language, VRML97 Specification `http://www.web3d.org/x3d/specifications/vrml/`

[87] Walczak, K., Cellary W., *Architectures for Interactive Television Services*, e-Minds: International Journal on Human-Computer Interaction, Volume 1, No 1, 2005, pp. 23-40

[88] Walczak, K., Cellary W., *X-VRML for Advanced Virtual Reality Applications*, IEEE Computer, Volume 36, Nr 3, pp. 89-92

[89] Walczak, K., Cellary W., Chmielewski J., Staniak M., Strykowski S., Wiza W., Wojciechowski R., Wojtowicz A., *An Architecture for Parameterised Production of Interactive TV Contents*, Proceedings of the 11th International Workshop on Systems, Signals And Image Processing; International Conference on Signals And Electronic Systems IWSSIP 2004, 13-15 September 2004, Poznań, Poland, pp. 465-468

[90] Walczak K., Cellary W., White M., `Virtual Museum Exhibitions`, IEEE Computer, Volume 39, Issue 3; March 2006, pp. 93-95

[91] Walczak K., Chmielewski J., Staniak M., Strykowski S., *Extensible Metadata Framework for Describing Virtual Reality and Multimedia Contents*, Proceedings of the 7th IASTED International Conference on Databases and Applications DBA 2006, February 14 - 16, 2006, Innsbruck, Austria, pp. 168-175

[92] Walczak K., Wojciechowski R., Cellary W., *Dynamic Interactive VR Network Services for Education*, Proceedings of the 13th ACM Symposium on Virtual Reality Software and Technology - VRST 2006, November 1-3, 2006, Limassol, Cyprus, pp. 277-286

[93] Web3D Consortium
`http://www.web3d.org/`

[94] Weibel S., Godby J., Miller E., Daniel R., *OCLC/NCSA Metadata Workshop Report*, DC-1: OCLC/NCSA Metadata Workshop, March 1-3, 1995, Dublin, Ohio USA, 1995
`http://dublincore.org/workshops/dc1/report.shtml`

[95] White, M., Mourkoussis N., Darcy J., Petridis P., Liarokapis F., Lister P., Walczak K., Wojciechowski R., Cellary W., Chmielewski J., Stawniak M., Wiza W., Patel M., Stevenson J., Manley J., Giorgini F., Sayd P., Gaspard F., *ARCO - An Architecture for Digitization, Management and Presentation of Virtual Exhibitions*, Proceedings of the Computer Graphics International CGI 2004, Hersonissos, Crete, Greece, pp. 622-625

[96] X3D Specification
`http://www.web3d.org/x3d/specifications/`

[97] XML Path Language (XPath) 2.0, W3C Recommendation 23 January 2007
`http://www.w3.org/TR/xpath20/`

[98] XML-QL: A Query Language for XML, W3C Proposal August 1998
`http://www.w3.org/TR/NOTE-xml-ql/`

[99] XML Query (XQuery) Requirements, W3C Working Group Note 23 March 2007
`http://www.w3.org/TR/xquery-requirements/`

[100] XML Query Working Group, W3C Working Group
`http://www.w3.org/XML/Query/`

[101] XML Schema Part 0: Primer Second Edition, W3C Recommendation
`http://www.w3.org/TR/xmlschema-0/`

[102] XML Syntax for XQuery 1.0 (XQueryX), W3C Recommendation 23 January 2007
`http://www.w3.org/TR/xqueryx/`

[103] XMP Specification, Adobe Systems Incorporated
`http://partners.adobe.com/public/developer/en/xmp/sdk/ XMPspecification.pdf`

[104] XQL (XML Query Language), W3C Proposal August 1999
`http://www.ibiblio.org/xql/xql-proposal.html`

[105] XQuery 1.0, W3C Recommendation 23 January 2007
`http://www.w3.org/TR/xquery/`

[106] XQuery 1.0, W3C Recommendation 23 January 2007, *FLWOR Expressions*
`http://www.w3.org/TR/xquery/#id-flwor-expressions`

[107] XQuery 1.0 and XPath 2.0 Data Model (XDM), W3C Recommendation 23 January 2007
`http://www.w3.org/TR/xpath-datamodel/`

[108] XQuery 1.0 and XPath 2.0 Formal Semantics, W3C Recommendation 23 January 2007
`http://www.w3.org/TR/xquery-semantics/`

[109] Zhou X., Zhang S.,Cao J., Dai K., *Event Condition Action Rule Based Intelligent Agent Architecture*, Journal of Shanghai Jiaotong University, Volume 38, Issue 1, January 2004, pp: 14-17

Appendix A. Interactions XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:v-ii="pl.pue.dit.ii.v-ii"
xmlns:mim="pl.pue.dit.mim" xmlns:ii="pl.pue.dit.ii" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:import namespace="pl.pue.dit.mim" schemaLocation="mim-schema.xsd"/>
    <xs:import namespace="pl.pue.dit.ii.v-ii" schemaLocation="vii-types.xsd"/>
    <xs:element name="Interactions">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="InteractionInterface" type="xs:anyURI" maxOccurs="unbounded"/>
                <xs:element name="Interaction" type="mim:interactionType" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="objectURI" type="xs:anyURI"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Appendix B. Multimedia Interaction Model XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="pl.pue.dit.mim" xmlns:ii="pl.pue.dit.ii" xmlns:mim="pl.pue.dit.mim"
xmlns:mathml="http://www.w3.org/1998/Math/MathML" xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:x3d="x3d" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:import namespace="http://www.w3.org/1998/Math/MathML" schemaLocation="mathml\mathml2.xsd"/>
    <xs:import namespace="urn:mpeg:mpeg7:schema:2001" schemaLocation="mpeg-7\Mpeg7-2001.xsd"/>
    <xs:import namespace="pl.pue.dit.ii" schemaLocation="ii-schema.xsd"/>
    <xs:import namespace="x3d" schemaLocation="x3d\x3d-3.0.xsd"/>
    <xs:simpleType name="valueBeforeOrAfterType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="before"/>
            <xs:enumeration value="after"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="conditionType">
        <xs:sequence>
            <xs:sequence minOccurs="0">
                <xs:element name="Expression" type="mathml:condition.type">
                    <xs:annotation>
                        <xs:documentation>MathML expression</xs:documentation>
                    </xs:annotation>
                </xs:element>
                <xs:element name="ExpressionArgument" type="mim:valueReference" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:element name="Semantics" type="mpeg7:TextAnnotationType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="eventType">
        <xs:sequence>
            <xs:element name="InteractionSpace" minOccurs="0">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="Space" minOccurs="0" maxOccurs="unbounded">
                            <xs:complexType>
                                <xs:group ref="x3d:GeometryContentModelInterchange"/>
                                <xs:attribute name="center" type="x3d:SFVec3f" default="0 0 0"/>
                                <xs:attribute name="rotation" type="x3d:SFRotation" default="0 0 1 0"/>
                                <xs:attribute name="scale" type="x3d:SFVec3f" default="1 1 1"/>
                                <xs:attribute name="scaleOrientation" type="x3d:SFRotation" default="0 0 1 0"/>
                                <xs:attribute name="translation" type="x3d:SFVec3f" default="0 0 0"/>
                            </xs:complexType>
                        </xs:element>
                        <xs:element name="Viewpoint" minOccurs="0" maxOccurs="unbounded">
                            <xs:complexType>
                                <xs:attribute name="fieldOfView" type="x3d:SFFloat" default="0.785398"/>
                                <xs:attribute name="jump" type="x3d:SFBool" default="true"/>
                                <xs:attribute name="orientation" type="x3d:SFRotation" default="0 0 1 0"/>
                                <xs:attribute name="position" type="x3d:SFVec3f" default="0 0 10"/>
                                <xs:attribute name="description" type="x3d:SFString"/>
                                <xs:attribute name="centerOfRotation" type="x3d:SFVec3f" default="0 0 0"/>
                            </xs:complexType>
                        </xs:element>
                        <xs:element name="Semantics" type="mpeg7:TextAnnotationType" minOccurs="0"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:sequence minOccurs="0">
                <xs:element name="Parameter" type="mim:parameterReference"/>
                <xs:element name="Argument" type="xs:anyType" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:element name="Semantics" type="mpeg7:TextAnnotationType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="actionType">
        <xs:sequence>
```

```
            <xs:choice minOccurs="0">
                <xs:element name="SubAction" type="mim:actionType" maxOccurs="unbounded"/>
                <xs:element name="Parameter" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:complexContent>
                            <xs:extension base="mim:parameterReference">
                                <xs:sequence>
                                    <xs:sequence minOccurs="0">
                                        <xs:element name="Result" type="mathml:apply.type">
                                            <xs:annotation>
                                                <xs:documentation>MathML expression</xs:documentation>
                                            </xs:annotation>
                                        </xs:element>
                                        <xs:element name="ResultArgument" type="mim:valueReference" minOccurs="0"
maxOccurs="unbounded"/>
                                    </xs:sequence>
                                    <xs:element name="Semantics" type="mpeg7:TextAnnotationType" minOccurs="0"/>
                                </xs:sequence>
                            </xs:extension>
                        </xs:complexContent>
                    </xs:complexType>
                </xs:element>
            </xs:choice>
            <xs:element name="Semantics" type="mpeg7:TextAnnotationType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="interactionType">
        <xs:sequence>
            <xs:element name="Event" type="mim:eventType" maxOccurs="unbounded"/>
            <xs:element name="Condition" type="mim:conditionType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="Action">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:extension base="mim:actionType"/>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="Semantics" type="mpeg7:TextAnnotationType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="parameterReference">
        <xs:attribute name="typeID" type="xs:anyURI" use="required"/>
    </xs:complexType>
    <xs:complexType name="valueReference">
        <xs:choice>
            <xs:element name="ParameterValue">
                <xs:complexType>
                    <xs:attribute name="typeID" type="xs:anyURI" use="required"/>
                    <xs:attribute name="variableBinding" type="xs:NCName" use="required"/>
                    <xs:attribute name="beforeOrAfter" type="mim:valueBeforeOrAfterType" use="optional"
default="after"/>
                    <xs:attribute name="object" use="optional" default="trigger">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:enumeration value="trigger"/>
                                <xs:enumeration value="interacting"/>
                                <xs:enumeration value="environment"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:attribute>
                </xs:complexType>
            </xs:element>
            <xs:element name="ConstValue">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:extension base="xs:anyType">
                            <xs:attribute name="typeID" type="xs:anyURI" use="required"/>
                            <xs:attribute name="variableBinding" type="xs:NCName" use="required"/>
                        </xs:extension>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:choice>
    </xs:complexType>
</xs:schema>
```

Appendix C. Interaction Interface XML Schema

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="pl.pue.dit.ii" xmlns:ii="pl.pue.dit.ii"
xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001" xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:import namespace="urn:mpeg:mpeg7:schema:2001" schemaLocation="mpeg-7\Mpeg7-2001.xsd"/>
    <xs:complexType name="parameterType">
        <xs:sequence>
            <xs:element name="DataType">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:extension base="xs:anyType"/>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="Argument" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:extension base="xs:anyType">
                            <xs:attribute name="name" type="xs:NCName" use="required"/>
                        </xs:extension>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="Semantics" type="mpeg7:TextAnnotationType" minOccurs="0"/>
            <xs:element name="Ontology" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:attribute name="ontology" type="xs:anyURI" use="required"/>
                    <xs:attribute name="concept" type="xs:anyURI" use="required"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="ID" type="xs:anyURI" use="required"/>
    </xs:complexType>
    <xs:complexType name="interactionInterfaceType">
        <xs:sequence>
            <xs:element name="ParameterType" type="ii:parameterType" maxOccurs="unbounded"/>
            <xs:element name="Semantics" type="mpeg7:TextAnnotationType" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="ID" type="xs:NMTOKEN" use="required"/>
        <xs:attribute name="name" type="xs:string"/>
    </xs:complexType>
    <xs:element name="InteractionInterface" type="ii:interactionInterfaceType"/>
</xs:schema>
```

Appendix D. Visual Interaction Interface definition

```
<?xml version="1.0" encoding="UTF-8"?>
<InteractionInterface xmlns="pl.pue.dit.ii" xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="pl.pue.dit.ii
ii-schema.xsd" ID="V-II" name="Visual Interaction Interface">
    <ParameterType ID="SizeBoundingBox">
        <DataType>
            <xs:simpleType>
                <xs:restriction base="x3d:BoundingBoxSize"/>
            </xs:simpleType>
        </DataType>
        <Semantics>
            <mpeg7:FreeTextAnnotation>Object size represented by a bounding box</mpeg7:FreeTextAnnotation>
        </Semantics>
    </ParameterType>
    <ParameterType ID="SizeBoundingSphere">
        <DataType>
            <xs:simpleType>
                <xs:restriction base="x3d:SFFloat"/>
            </xs:simpleType>
        </DataType>
        <Semantics>
            <mpeg7:FreeTextAnnotation>Object size represented by a bounding sphere</mpeg7:FreeTextAnnotation>
        </Semantics>
    </ParameterType>
    <ParameterType ID="SizeDimension">
        <DataType>
            <xs:simpleType>
                <xs:restriction base="x3d:SFFloat"/>
            </xs:simpleType>
        </DataType>
        <Argument name="vector">
            <xs:complexType>
                <xs:attribute name="start" type="x3d:SFVec3f" use="optional" default="0 0 0"/>
                <xs:attribute name="end" type="x3d:SFVec3f" use="required"/>
            </xs:complexType>
        </Argument>
        <Semantics>
            <mpeg7:FreeTextAnnotation>Distance between two furthest points of an object measured along a given
vector</mpeg7:FreeTextAnnotation>
        </Semantics>
    </ParameterType>
    <ParameterType ID="ShapePrimitive">
        <DataType>
            <xs:complexType>
                <xs:attribute name="name" use="required">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="box"/>
                            <xs:enumeration value="cone"/>
                            <xs:enumeration value="cylinder"/>
                            <xs:enumeration value="sphere"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
            </xs:complexType>
        </DataType>
        <Semantics>
            <mpeg7:FreeTextAnnotation>Name of the shape of an object taken from a predefined dictionary, e.g. box,
cone, sphere, etc.</mpeg7:FreeTextAnnotation>
        </Semantics>
    </ParameterType>
    <ParameterType ID="ShapeFaceSet">
        <DataType>
            <xs:complexType>
                <xs:all>
                    <xs:element ref="x3d:IndexedFaceSet"/>
                </xs:all>
            </xs:complexType>
        </DataType>
        <Semantics>
            <mpeg7:FreeTextAnnotation>Shape of an object described by a set of points and faces connecting these
points</mpeg7:FreeTextAnnotation>
        </Semantics>
    </ParameterType>
    <ParameterType ID="ShapeCustom">
        <DataType>
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="Shape" type="mpeg7:Shape3DType"/>
                </xs:sequence>
            </xs:complexType>
        </DataType>
        <Semantics>
            <mpeg7:FreeTextAnnotation>Shape of an object described by an MPEG-7 3D Shape
    descriptor</mpeg7:FreeTextAnnotation>
        </Semantics>
    </ParameterType>
    <ParameterType ID="ShapeContour">
        <DataType>
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="Contour" type="mpeg7:ContourShapeType"/>
                </xs:sequence>
            </xs:complexType>
        </DataType>
        <Argument name="plane">
            <xs:complexType>
                <xs:attribute name="p1" type="x3d:SFVec3f" use="required"/>
                <xs:attribute name="p2" type="x3d:SFVec3f" use="required"/>
                <xs:attribute name="p3" type="x3d:SFVec3f" use="required"/>
            </xs:complexType>
        </Argument>
        <Semantics>
            <mpeg7:FreeTextAnnotation>Shape of a contour of a 3D object projection onto a 3D plane described by
MPEG-7 Contour-Based Shape descriptor</mpeg7:FreeTextAnnotation>
        </Semantics>
    </ParameterType>
    <ParameterType ID="ShapeRegion">
        <DataType>
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="Region" type="mpeg7:RegionShapeType"/>
                </xs:sequence>
            </xs:complexType>
        </DataType>
        <Argument name="plane">
            <xs:complexType>
                <xs:attribute name="p1" type="x3d:SFVec3f" use="required"/>
                <xs:attribute name="p2" type="x3d:SFVec3f" use="required"/>
                <xs:attribute name="p3" type="x3d:SFVec3f" use="required"/>
            </xs:complexType>
        </Argument>
        <Semantics>
            <mpeg7:FreeTextAnnotation/>
        </Semantics>
    </ParameterType>
    <ParameterType ID="PositionCenter">
        <DataType>
            <xs:simpleType>
                <xs:restriction base="x3d:SFVec3f"/>
            </xs:simpleType>
        </DataType>
        <Argument name="bbox">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="box"/>
                    <xs:enumeration value="sphere"/>
                </xs:restriction>
            </xs:simpleType>
        </Argument>
        <Semantics>
            <mpeg7:FreeTextAnnotation>Coordinates of the center of object bounding sphere or object bounding box.
Relatively to the described object</mpeg7:FreeTextAnnotation>
        </Semantics>
    </ParameterType>
    <ParameterType ID="PositionCollision">
        <DataType>
            <xs:simpleType>
                <xs:restriction base="zs:boolean"/>
            </xs:simpleType>
        </DataType>
        <Semantics>
            <mpeg7:FreeTextAnnotation>Collision flag indicating collision of interacting object and trigger
object</mpeg7:FreeTextAnnotation>
        </Semantics>
    </ParameterType>
    <ParameterType ID="Orientation">
        <DataType>
            <xs:simpleType>
                <xs:restriction base="x3d:MFRotation"/>
            </xs:simpleType>
        </DataType>
        <Semantics>
            <mpeg7:FreeTextAnnotation>Object orientation relative to 3D Virtual World
axes</mpeg7:FreeTextAnnotation>
        </Semantics>
    </ParameterType>
    <ParameterType ID="AppearanceTexture">
        <DataType>
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="TextureBrowsing" type="mpeg7:TextureBrowsingType"/>
                </xs:sequence>
            </xs:complexType>
        </DataType>
        <Semantics>
            <mpeg7:FreeTextAnnotation>Object texture described by MPEG-7 Texture Browsing
descriptor</mpeg7:FreeTextAnnotation>
        </Semantics>
    </ParameterType>
    <ParameterType ID="AppearanceTextureHomogeneous">
        <DataType>
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="HomogeneousTexture" type="mpeg7:HomogeneousTextureType"/>
                </xs:sequence>
            </xs:complexType>
        </DataType>
        <Semantics>
            <mpeg7:FreeTextAnnotation>Object texture described by MPEG-7 Homogeneous  Texture
descriptor</mpeg7:FreeTextAnnotation>
        </Semantics>
    </ParameterType>
    <ParameterType ID="AppearanceColor">
        <DataType>
            <xs:simpleType>
                <xs:restriction base="SFColor"/>
            </xs:simpleType>
        </DataType>
        <Semantics>
            <mpeg7:FreeTextAnnotation>Object uniform color</mpeg7:FreeTextAnnotation>
        </Semantics>
    </ParameterType>
    <ParameterType ID="AppearanceColorDominant">
        <DataType>
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="DominantColor" type="mpeg7:DominantColorType"/>
                </xs:sequence>
            </xs:complexType>
        </DataType>
        <Semantics>
            <mpeg7:FreeTextAnnotation>Object color described by MPEG-7 Dominant Color
descriptor</mpeg7:FreeTextAnnotation>
        </Semantics>
    </ParameterType>
    <ParameterType ID="AppearanceColorLayout">
        <DataType>
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="ColorLayout" type="mpeg7:ColorLayoutType"/>
                </xs:sequence>
            </xs:complexType>
        </DataType>
        <Semantics>
            <mpeg7:FreeTextAnnotation>Object color described by MPEG-7 Color Layout
descriptor</mpeg7:FreeTextAnnotation>
        </Semantics>
    </ParameterType>
    <ParameterType ID="AppearanceColorStructure">
        <DataType>
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="ColorStructure" type="mpeg7:ColorStructureType"/>
                </xs:sequence>
            </xs:complexType>
        </DataType>
        <Semantics>
            <mpeg7:FreeTextAnnotation>Object color described by MPEG-7 Color Structure
descriptor</mpeg7:FreeTextAnnotation>
        </Semantics>
    </ParameterType>
    <ParameterType ID="AppearanceTransparency">
        <DataType>
            <xs:simpleType>
                <xs:restriction base="SFFloat">
                    <xs:minInclusive value="0"/>
                    <xs:maxInclusive value="1"/>
                </xs:restriction>
            </xs:simpleType>
        </DataType>
        <Semantics>
            <mpeg7:FreeTextAnnotation>Object transparency</mpeg7:FreeTextAnnotation>
        </Semantics>
    </ParameterType>
    <Semantics>
        <mpeg7:FreeTextAnnotation>Visual Interaction Interface defines types of parameters required to desctibe basic
interactions based on object visual characteristics.</mpeg7:FreeTextAnnotation>
    </Semantics>
</InteractionInterface>
```

Appendix E. Examples of interaction metadata

Changing Ball

```
<?xml version="1.0" encoding="UTF-8"?>
<Interactions xmlns:ii="pl.pue.dit.ii" xmlns:mim="pl.pue.dit.mim" xmlns:v-ii="pl.pue.dit.ii.v-ii"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:math="http://www.w3.org/1998/Math/MathML"
xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001" xmlns:x3d="x3d" xsi:noNamespaceSchemaLocation="interactions-
schema.xsd">
   <InteractionInterface>pl.put.dit.ii.v-ii</InteractionInterface>
   <Interaction>
      <mim:Event>
         <mim:InteractionSpace>
            <mim:Space>
               <x3d:Sphere radius="50"/>
            </mim:Space>
            <mim:Semantics>
               <mpeg7:KeywordAnnotation>
                  <mpeg7:Keyword>sphere</mpeg7:Keyword>
                  <mpeg7:Keyword>surroundings</mpeg7:Keyword>
               </mpeg7:KeywordAnnotation>
            </mim:Semantics>
         </mim:InteractionSpace>
         <mim:Parameter typeID="V-II.PositionCenter"/>
      </mim:Event>
      <mim:Condition>
         <mim:Expression>
            <math:apply>
               <math:eq/>
               <math:csymbol>S</math:csymbol>
               <math:cn>cone</math:cn>
            </math:apply>
         </mim:Expression>
         <mim:ExpressionArgument>
            <mim:ParameterValue typeID="V-II.ShapePrimitive" variableBinding="S"/>
         </mim:ExpressionArgument>
      </mim:Condition>
      <mim:Action>
         <mim:Parameter typeID="V-II.ShapePrimitive">
            <mim:Result>
               <math:apply>
                  <math:cn>box</math:cn>
               </math:apply>
            </mim:Result>
         </mim:Parameter>
         <mim:Semantics>
            <mpeg7:StructuredAnnotation>
               <mpeg7:WhatObject>
                  <mpeg7:Name>shape</mpeg7:Name>
               </mpeg7:WhatObject>
               <mpeg7:WhatAction>
                  <mpeg7:Name>change</mpeg7:Name>
               </mpeg7:WhatAction>
               <mpeg7:How>
                  <mpeg7:Name>to box</mpeg7:Name>
               </mpeg7:How>
            </mpeg7:StructuredAnnotation>
         </mim:Semantics>
      </mim:Action>
      <mim:Semantics>
         <mpeg7:FreeTextAnnotation>Object changes its shape to box when approached by an object with cone
shape</mpeg7:FreeTextAnnotation>
      </mim:Semantics>
   </Interaction>
</Interactions>
```

Ad sign

```
<?xml version="1.0" encoding="UTF-8"?>
<Interactions xmlns:ii="pl.pue.dit.ii" xmlns:mim="pl.pue.dit.mim" xmlns:v-ii="pl.pue.dit.ii.v-ii"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:math="http://www.w3.org/1998/Math/MathML"
xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001" xmlns:x3d="x3d" xsi:noNamespaceSchemaLocation="interactions-
schema.xsd">
    <InteractionInterface>pl.put.dit.ii.v-ii</InteractionInterface>
    <Interaction>
        <mim:Event>
            <mim:InteractionSpace>
                <mim:Viewpoint orientation="0 1 0 1.57" position="0 0 0"/>
            </mim:InteractionSpace>
            <mim:Parameter typeID="V-II.PositionCenter"/>
        </mim:Event>
        <mim:Condition>
            <mim:Expression>
                <math:apply>
                    <math:eq/>
                    <math:csymbol>S</math:csymbol>
                    <math:csymbol>avatar</math:csymbol>
                </math:apply>
            </mim:Expression>
            <mim:ExpressionArgument>
                <mim:ParameterValue typeID="V-II.ShapeCustom" variableBinding="S"/>
            </mim:ExpressionArgument>
            <mim:ExpressionArgument>
                <mim:ConstValue typeID="V-II.ShapeCustom" variableBinding="avatar">
                    <v-ii:Shape>
                        <mpeg7:Spectrum>12 130 148 22 301</mpeg7:Spectrum>
                        <mpeg7:PlanarSurfaces>0</mpeg7:PlanarSurfaces>
                        <mpeg7:SingularSurfaces>0</mpeg7:SingularSurfaces>
                    </v-ii:Shape>
                </mim:ConstValue>
            </mim:ExpressionArgument>
        </mim:Condition>
        <mim:Action>
            <mim:Parameter typeID="V-II.SizeBoundingBox.z">
                <mim:Result>
                    <math:apply>
                        <math:plus/>
                        <math:csymbol>AS.z</math:csymbol>
                        <math:apply>
                            <math:times/>
                            <math:csymbol>AS.z</math:csymbol>
                            <math:apply>
                                <math:ln/>
                                <math:apply>
                                    <math:root/>
                                    <math:degree>
                                        <math:cn>2</math:cn>
                                    </math:degree>
                                    <math:apply>
                                        <math:plus/>
                                        <math:apply>
                                            <math:power/>
                                            <math:csymbol>AP.x</math:csymbol>
                                            <math:cn>2</math:cn>
                                        </math:apply>
                                        <math:apply>
                                            <math:power/>
                                            <math:csymbol>AP.y</math:csymbol>
                                            <math:cn>2</math:cn>
                                        </math:apply>
                                    </math:apply>
                                </math:apply>
```

```
                              </math:apply>
                          </math:apply>
                      </math:apply>
                  </mim:Result>
                  <mim:ResultArgument>
                      <mim:ParameterValue typeID="V-II.PositionCenter" variableBinding="AP" beforeOrAfter="after"/>
                  </mim:ResultArgument>
                  <mim:ResultArgument>
                      <mim:ParameterValue typeID="V-II.SizeBoundingBox" variableBinding="AS" beforeOrAfter="after"/>
                  </mim:ResultArgument>
              </mim:Parameter>
              <mim:Semantics>
                  <mpeg7:FreeTextAnnotation>Height modified depending on the avatar height and
distance</mpeg7:FreeTextAnnotation>
              </mim:Semantics>
          </mim:Action>
          <mim:Semantics>
              <mpeg7:FreeTextAnnotation>An ad sign that modifies its height depending on the distance and hight of the
avatar approaching the sign.</mpeg7:FreeTextAnnotation>
          </mim:Semantics>
      </Interaction>
</Interactions>
```

Treasure chest

```
<?xml version="1.0" encoding="UTF-8"?>
<Interactions xmlns:ii="pl.pue.dit.ii" xmlns:mim="pl.pue.dit.mim" xmlns:v-ii="pl.pue.dit.ii.v-ii" xmlns:sg-ii="pl.pue.dit.ii.sg-ii"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:math="http://www.w3.org/1998/Math/MathML"
xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001" xmlns:x3d="x3d" xsi:noNamespaceSchemaLocation="interactions-
schema.xsd">
      <InteractionInterface>pl.put.dit.ii.sg-ii</InteractionInterface>
      <Interaction>
          <mim:Event>
              <mim:InteractionSpace>
                  <mim:Space>
                      <x3d:Sphere radius="5"/>
                  </mim:Space>
                  <mim:Semantics>
                      <mpeg7:FreeTextAnnotation>A circle of 5 meters around the chest</mpeg7:FreeTextAnnotation>
                  </mim:Semantics>
              </mim:InteractionSpace>
              <mim:Parameter typeID="AG-II.Spell"/>
          </mim:Event>
          <mim:Condition>
              <mim:Expression>
                  <math:apply>
                      <math:and/>
                      <math:apply>
                          <math:eq/>
                          <math:csymbol>C</math:csymbol>
                          <math:cn>true</math:cn>
                      </math:apply>
                      <math:apply>
                          <math:eq/>
                          <math:csymbol>L</math:csymbol>
                          <math:cn>true</math:cn>
                      </math:apply>
                      <math:apply>
                          <math:eq/>
                          <math:csymbol>S</math:csymbol>
                          <math:cn>unlock</math:cn>
                      </math:apply>
                      <math:apply>
                          <math:in/>
                          <math:cn>'green key'</math:cn>
```

```
                    <math:csymbol>I</math:csymbol>
                </math:apply>
            </math:apply>
        </mim:Expression>
        <mim:ExpressionArgument>
            <mim:ParameterValue typeID="CIG-II.Closed" variableBinding="C"/>
        </mim:ExpressionArgument>
        <mim:ExpressionArgument>
            <mim:ParameterValue typeID="CIG-II.Locked" variableBinding="L"/>
        </mim:ExpressionArgument>
        <mim:ExpressionArgument>
            <mim:ParameterValue typeID="AG-II.Spell" variableBinding="S"/>
        </mim:ExpressionArgument>
        <mim:ExpressionArgument>
            <mim:ParameterValue typeID="AG-II.Inventory" variableBinding="I"/>
        </mim:ExpressionArgument>
    </mim:Condition>
    <mim:Action>
        <mim:Parameter typeID="CIG-II.Locked">
            <mim:Result>
                <math:apply>
                    <math:cn>false</math:cn>
                </math:apply>
            </mim:Result>
        </mim:Parameter>
        <mim:Semantics>
            <mpeg7:FreeTextAnnotation>Unlock the chest</mpeg7:FreeTextAnnotation>
        </mim:Semantics>
    </mim:Action>
    <mim:Semantics>
        <mpeg7:FreeTextAnnotation>Chest full of treasures opens when approached by avatar with key in the
inventory</mpeg7:FreeTextAnnotation>
    </mim:Semantics>
  </Interaction>
</Interactions>
```

Eye

```
<?xml version="1.0" encoding="UTF-8"?>
<Interactions xmlns:ii="pl.pue.dit.ii" xmlns:mim="pl.pue.dit.mim" xmlns:e-ii="pl.pue.dit.ii.e-ii"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:math="http://www.w3.org/1998/Math/MathML"
xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001" xsi:noNamespaceSchemaLocation="interactions-schema.xsd">
  <InteractionInterface>pl.put.dit.ii.e-ii</InteractionInterface>
  <Interaction>
    <mim:Event>
        <mim:Semantics>
            <mpeg7:StructuredAnnotation>
                <mpeg7:WhatObject>
                    <mpeg7:Term termID="urn:iso:physics:light:intensity"/>
                </mpeg7:WhatObject>
                <mpeg7:WhatAction>
                    <mpeg7:Term termID="urn:iso:standardverbs:increase"/>
                </mpeg7:WhatAction>
            </mpeg7:StructuredAnnotation>
        </mim:Semantics>
    </mim:Event>
    <mim:Action>
        <mim:Parameter typeID="E-II.PupilSize"/>
        <mim:Semantics>
            <mpeg7:StructuredAnnotation>
                <mpeg7:WhatAction>
                    <mpeg7:Term termID="urn:gmo:verbs:constrict"/>
                </mpeg7:WhatAction>
            </mpeg7:StructuredAnnotation>
        </mim:Semantics>
    </mim:Action>
```

```
            <mim:Semantics xml:lang="en">
                <mpeg7:FreeTextAnnotation>Eye pupil constricts when light intensity
increases</mpeg7:FreeTextAnnotation>
            </mim:Semantics>
        </Interaction>
        <Interaction>
            <mim:Event>
                <mim:Semantics>
                    <mpeg7:StructuredAnnotation>
                        <mpeg7:WhatObject>
                            <mpeg7:Term termID="urn:iso:physics:light:intensity"/>
                        </mpeg7:WhatObject>
                        <mpeg7:WhatAction>
                            <mpeg7:Term termID="urn:iso:standardverbs:decrease"/>
                        </mpeg7:WhatAction>
                    </mpeg7:StructuredAnnotation>
                </mim:Semantics>
            </mim:Event>
            <mim:Action>
                <mim:Parameter typeID="E-II.PupilSize"/>
                <mim:Semantics>
                    <mpeg7:StructuredAnnotation>
                        <mpeg7:WhatAction>
                            <mpeg7:Term termID="urn:gmo:verbs:dilate"/>
                        </mpeg7:WhatAction>
                    </mpeg7:StructuredAnnotation>
                </mim:Semantics>
            </mim:Action>
            <mim:Semantics xml:lang="en">
                <mpeg7:FreeTextAnnotation>Eye pupil dilates when light intensity decreases</mpeg7:FreeTextAnnotation>
            </mim:Semantics>
        </Interaction>
</Interactions>
```

Sliding door

```
<?xml version="1.0" encoding="UTF-8"?>
<Interactions xmlns:ii="pl.pue.dit.ii" xmlns:mim="pl.pue.dit.mim" xmlns:v-ii="pl.pue.dit.ii.v-ii" xmlns:a-ii="pl.pue.dit.ii.a-ii"
xmlns:door-ii="pl.pue.dit.ii.door-ii" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:math="http://www.w3.org/1998/Math/MathML" xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
xsi:noNamespaceSchemaLocation=" interactions-schema.xsd">
        <InteractionInterface>pl.put.dit.ii.v-ii</InteractionInterface>
        <InteractionInterface>pl.put.dit.ii.a-ii</InteractionInterface>
        <InteractionInterface>pl.put.dit.ii.AIdoor-ii</InteractionInterface>
        <Interaction>
            <mim:Event>
                <mim:InteractionSpace>
                    <mim:Viewpoint position="0 1.25 0" fieldOfView="0.785398" orientation="1 0 0 -2.355"/>
                    <mim:Semantics>
                        <mpeg7:FreeTextAnnotation>Door motion sensor monitoring range</mpeg7:FreeTextAnnotation>
                    </mim:Semantics>
                </mim:InteractionSpace>
                <mim:Parameter typeID="V-II.PositionCenter"/>
            </mim:Event>
            <mim:Condition>
                <mim:Expression>
                    <math:apply>
                        <math:and/>
                        <math:apply>
                            <math:eq/>
                            <math:csymbol>C</math:csymbol>
                            <math:cn>true</math:cn>
                        </math:apply>
                        <math:apply>
                            <math:eq/>
                            <math:csymbol>L</math:csymbol>
                            <math:cn>false</math:cn>
                        </math:apply>
                        <math:apply>
                            <math:geq/>
                            <math:apply>
                                <math:times/>
                                <math:csymbol>S.x</math:csymbol>
                                <math:csymbol>S.y</math:csymbol>
                                <math:csymbol>S.z</math:csymbol>
                            </math:apply>
                            <math:cn>0.82</math:cn>
                        </math:apply>
                    </math:apply>
                </mim:Expression>
                <mim:ExpressionArgument>
                    <mim:ParameterValue typeID="AIdoor-II.Closed" variableBinding="C"/>
                </mim:ExpressionArgument>
                <mim:ExpressionArgument>
                    <mim:ParameterValue typeID="AIdoor-II.Closed" variableBinding="L"/>
                </mim:ExpressionArgument>
                <mim:ExpressionArgument>
                    <mim:ParameterValue typeID="V-II.SizeBoundingBox" variableBinding="S"/>
                </mim:ExpressionArgument>
            </mim:Condition>
            <mim:Action>
                <mim:SubAction>
                    <mim:Parameter typeID="A-II.PlayedSound">
                        <mim:Result>
                            <math:apply>
                                <math:csymbol>sound</math:csymbol>
                            </math:apply>
                        </mim:Result>
                        <mim:ResultArgument>
                            <mim:ConstValue typeID="A-II.PlayedSound" variableBinding="sound">
                                <mpeg7:AudioD xsi:type="mpeg7:AudioSpectrumBasisType" mpeg7:loEdge="62.5"
mpeg7:hiEdge="8000" mpeg7:resolution="1/4 octave">
                            <mpeg7:BasisFunctions>
                                <mpeg7:Matrix dim="10 5">0.26 -0.05 0.01 -0.70 0.44
0.34 0.09 0.21 -0.42 -0.05
0.33 0.15 0.24 -0.05 -0.39
0.33 0.15 0.24 -0.05 -0.39
0.27 0.13 0.16 0.24 -0.04
0.27 0.13 0.16 0.24 -0.04
0.23 0.13 0.09 0.27 0.24
0.20 0.13 0.04 0.22 0.40
0.17 0.11 0.01 0.14 0.37
0.33 -0.15 0.24 0.05 0.39</mpeg7:Matrix>
                            </mpeg7:BasisFunctions>
                                </mpeg7:AudioD>
                            </mim:ConstValue>
                        </mim:ResultArgument>
                    </mim:Parameter>
                </mim:SubAction>
                <mim:SubAction>
                    <mim:Semantics>
                        <mpeg7:FreeTextAnnotation>Door sides slide slowly to the sides.</mpeg7:FreeTextAnnotation>
                    </mim:Semantics>
                </mim:SubAction>
            </mim:Action>
            <mim:Semantics xml:lang="en">
                <mpeg7:FreeTextAnnotation>Door opens when approched by an object.</mpeg7:FreeTextAnnotation>
            </mim:Semantics>
        </Interaction>
        <Interaction>
            <mim:Event>
                <mim:InteractionSpace>
                    <mim:Viewpoint position="0 1.25 0" fieldOfView="0.785398" orientation="1 0 0 -2.355"/>
                    <mim:Semantics>
                        <mpeg7:FreeTextAnnotation>Door motion sensor monitoring range</mpeg7:FreeTextAnnotation>
                    </mim:Semantics>
                </mim:InteractionSpace>
                <mim:Parameter typeID="V-II.PositionCenter"/>
            </mim:Event>
            <mim:Condition>
                <mim:Expression>
                    <math:apply>
                        <math:and/>
                        <math:apply>
                            <math:eq/>
                            <math:csymbol>C</math:csymbol>
                            <math:cn>false</math:cn>
                        </math:apply>
                        <math:apply>
                            <math:eq/>
                            <math:csymbol>L</math:csymbol>
                            <math:cn>false</math:cn>
                        </math:apply>
                        <math:apply>
                            <math:geq/>
                            <math:apply>
                                <math:times/>
                                <math:csymbol>S.x</math:csymbol>
                                <math:csymbol>S.y</math:csymbol>
                                <math:csymbol>S.z</math:csymbol>
                            </math:apply>
                            <math:cn>0.82</math:cn>
                        </math:apply>
                        <math:apply>
                            <math:not/>
                            <math:apply>
                                <math:in/>
                                <math:csymbol>P</math:csymbol>
                                <math:csymbol>event.space</math:csymbol>
                            </math:apply>
                        </math:apply>
                    </math:apply>
                </mim:Expression>
                <mim:ExpressionArgument>
                    <mim:ParameterValue typeID="Aldoor-II.Closed" variableBinding="C"/>
                </mim:ExpressionArgument>
                <mim:ExpressionArgument>
                    <mim:ParameterValue typeID="Aldoor-II.Closed" variableBinding="L"/>
                </mim:ExpressionArgument>
                <mim:ExpressionArgument>
                    <mim:ParameterValue typeID="V-II.SizeBoundingBox" variableBinding="S"/>
                </mim:ExpressionArgument>
                <mim:ExpressionArgument>
                    <mim:ParameterValue typeID="V-II.PositionCenter" variableBinding="P" beforeOrAfter="after"/>
                </mim:ExpressionArgument>
            </mim:Condition>
            <mim:Action>
                <mim:SubAction>
                    <mim:Parameter typeID="A-II.PlayedSound">
                        <mim:Result>
                            <math:apply>
                                <math:csymbol>sound</math:csymbol>
                            </math:apply>
                        </mim:Result>
                        <mim:ResultArgument>
                            <mim:ConstValue typeID="A-II.PlayedSound" variableBinding="sound">
                                <mpeg7:AudioD xsi:type="mpeg7:AudioSpectrumBasisType" mpeg7:loEdge="62.5"
mpeg7:hiEdge="8000" mpeg7:resolution="1/4 octave">
                            <mpeg7:BasisFunctions>
                                <mpeg7:Matrix dim="10 5">0.26 -0.05 0.01 -0.70 0.44
0.34 0.09 0.21 -0.42 -0.05
0.33 0.85 0.24 -0.05 -0.39
0.33 0.85 0.24 -0.05 -0.39
0.27 0.83 0.16 0.24 -0.04
0.27 0.83 0.16 0.24 -0.04
0.23 0.83 0.09 0.27 0.24
0.20 0.83 0.04 0.22 0.40
```

```
0.17 0.81 0.01 0.14 0.37
0.33 -0.15 0.24 0.05 0.39</mpeg7:Matrix>
                                        </mpeg7:BasisFunctions>
                                   </mpeg7:AudioD>
                              </mim:ConstValue>
                         </mim:ResultArgument>
                    </mim:Parameter>
               </mim:SubAction>
               <mim:SubAction>
                    <mim:Semantics>
                         <mpeg7:FreeTextAnnotation>Door sides slide slowly back to the
center.</mpeg7:FreeTextAnnotation>
                    </mim:Semantics>
               </mim:SubAction>
          </mim:Action>
          <mim:Semantics xml:lang="en">
               <mpeg7:FreeTextAnnotation>Door closes when object leaves the sensor
range</mpeg7:FreeTextAnnotation>
          </mim:Semantics>
     </Interaction>
</Interactions>
```