MPEG-4 SDK: From Specifications to Real Applications

OCTAVIAN FOLEA, MARIUS PREDA, FRANCOISE PRETEUX
ARTEMIS Project Unit
GET-INT
9, rue Charles Fourier, 91000 Evry,
FRANCE

Abstract: - Nowadays, multimedia contents are invading the digital world through a wide family of distribution channels, and are played on terminals with various capabilities and resources. In such hybrid and heterogeneous environments, the availability of international standards enabling application interoperability becomes a strong requirement. MPEG-4 is a powerful multimedia standard in terms of media representation, scene composition, and user interactivity. However, building an MPEG-4 application demands an in-depth knowledge of MPEG-4 specifications which currently limits the world wide deployment of the standard. In order to facilitate MPEG-4 usability for non-expert developers, this paper addresses the technical issues related to the implementation of an MPEG-4 Software Development Kit (SDK). By focusing on the MPEG-4 scene composition level and on the graphics representation features, we build a low-level MPEG-4 SDK (scene graph access, media and stream processing) compliant with scene and media specifications. The more specific issue of virtual character animation is addressed by developing an intuitive API; it is referred to as MPEG-4 VC API and supports high-level functionalities (data-based semantic access, hierarchical object processing and stream control). We demonstrate the relevance of this toolkit for the easy design and creation of real applications by implementing a plug-in providing MPEG-4 content importing/exporting. No particular knowledge of the MPEG-4 standard is therefore required. The SDK toolkit is extensively evaluated within the OLGA IST FP6 European Project.

Key-Words: multimedia systems, MPEG-4, graphics content creation, virtual characters, 3dsmax

1 Introduction

Starting with the nineties, the boom of the computer graphics field has been determined by the migration of the 3D software platforms from SGI workstations to PCs. Increased availability and demand, advanced technologies, and improved hardware were the keywords that convinced the major market actors to create powerful authoring tools for 2D/3D content processing. Maya (Alias|Wavefront), 3dsmax (Discreet), Animation Master (Hash Inc.), XSI (SoftImage), Poser (Curious Lab) are popular and functionality-rich software solutions spread all over the world. But the "jungle" of the proprietary data formats developed on each of these platforms raised fundamental problems for end-user software integrators: poor performance of format conversion software, inadequacy of these formats for specific applications and difficulty to use undocumented (or not sufficient documented) proprietary solutions.

These reasons motivated the computer graphics community to start developing open standards in order to achieve two key requirements: interoperability and access to the specifications. The former ensures the common bases for applications and solutions provided by independent parties and the later prevents the situation in which a unique company can maintain and control the updates and the development process of the standard. Recently, Alias released a file format called FBX, with the goal of providing the reference format for exchanging 3D content between authoring tools. Alias FBX is a platform-independent 3D authoring and interchange format that provides access to 3D content from most 3D vendors. The FBX file format supports all major 3D data elements, as well as 2D, audio, and video media elements [8]. Despite its advanced features, FBX represents a proprietary solution. CyberVRML [6] is a low-level API, wrapped on VRML nodes, allowing easy integration of a VRML parser into 3D applications. Among existent or on-going multimedia standards, MPEG-4 [1] is one of the most complete in terms of media representation, compression, 2D and 3D graphics primitives, user interaction and programmatic environment. As a member of the MPEG family, the MPEG-4 standard inherits and improves all the features of its predecessors, offering the possibility of efficient transmitting and/or storing a huge amount of digital audio / video. Furthermore, the standard jumps a great step forward with key-techniques, as advanced audio coding, video compression-based on visual object, wavelet deployment and mesh-based representation. The most promoting feature of MPEG-4 is probably the definition of BInary Format for Scenes (BIFS) as a part of system information, which lies on the top of

all media data and acts as an intermediate layer between media data and the final displayed content. It makes possible to manipulate various types of media in an MPEG-4 scene, such as scheduling, coordinating in temporal and / or spatial domain, synchronizing, processing interactivity, etc. BIFS is based widely on Virtual Reality Modeling Language (VRML) [2]. It is a binary encoded version of an extended subset of VRML, which can represent roughly the same scene as with VRML in a much more efficient manner. The wide range of functionalities supported by MPEG-4 makes this standard one of the most complete and advanced solution, and companies are slowly deploying MPEG-4 technologies inside their applications. As for example, let us mention the Keynote toll from Apple which allows exporting MPEG-4 presentations [2]. However, the main drawback of integrating MPEG-4 solutions is the complexity. Today, in order to include MPEG-4 content within an application, a software developer has to implement: (1) MPEG-4 codec for each of the elementary stream that composing the MPEG-4 scene, and (2) MPEG-4 importer/exporter for his own data structures. This requires an advanced MPEG-4 expertise over the standard and an expensive development effort. Moreover, the developed MPEG-4 modules are application-specific, and difficult to re-use. In order to ease the understanding of the standard, the MPEG group also released a special part called Reference Software [3] which implements the specifications (usually in a not-optimized manner). The Reference Software can be used as a starting point in developing applications or as a reference for implementing optimized software. However, good knowledge of the standard is required in both cases. Unfortunately, the MPEG group does not provide any Software Development Kit (SDK) that can be used to solve decoding and formatting the MPEG-4 data.

In this paper we propose such MPEG-4 SDK. The flexible framework proposed makes it possible to support the MPEG-4 functionalities to ease the integration phase by modulating and reuse of code (Fig.1).

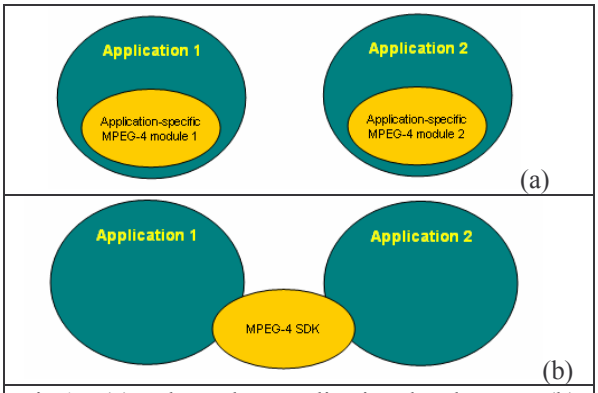

Fig.1 – (a) Independent application development (b) Development using the MPEG-4 SDK.

The structure of this paper is as follows. Section 2 presents the basic concepts of the MPEG-4 SDK, a low-level API that wrap the scene graph and the elementary streams. Then, a high-level API derived from the MPEG-4 SDK, called Virtual Character SDK (VC SDK) is described. VC SDK deals with the definition and animation of a skinned model as specified in MPEG-4 Part 16 – Animation Framework eXtension (AFX) [4]. The implementation principles of the MPEG-4 SDK and VC SDK are reported. Finally, Section 3 demonstrates the relevance of these APIs for developing applications. The presented application is an MPEG-4 importer/exporter plug-in for virtual characters, which allows transforming 3dsmax into a fully-compatible MPEG-4 authoring tool. The final section summarizes and discusses the experimental results and opens the perspectives for future work.

2 MP4SDK concepts

Defining the SDK for a data format can be performed at two levels: a low-level for wrapping the data representation and a high-level for accessing according to their semantic meaning. The first level is generic, while the second one is specific to the type of data. Both approaches are used in our implementation: the low-level API gives access to the entire scene graph and to the media layer. The high-level API, defined for a specific part of the standard, namely the representation of a VC, allows semantic-based access on the definition and the animation of a skinned model.

2.1 Low-level API: MPEG-4 SDK

The MPEG-4 SDK is designed to offer an access to the MPEG-4 fundamental elements - the BIFS scene and the elementary streams – transparently with respect to the compression layer. The user of this SDK must know the interface of the nodes defined by the standard but no knowledge is

required concerning how this information is compressed. Fig.2 presents the MPEG-4 SDK architecture.

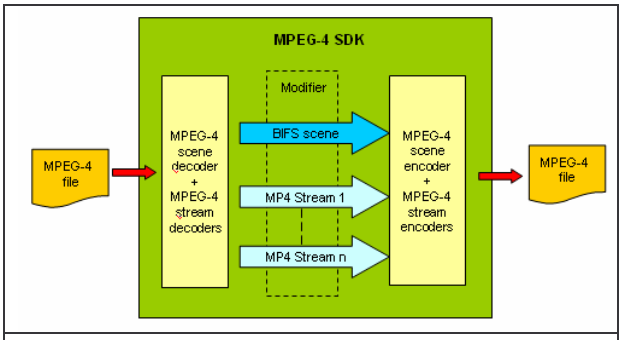

Fig.2 –Architecture of MPEG-4 SDK.

The MPEG-4 SDK main functionalities refer to (1) encoding/decoding MPEG-4 scene and elementary streams; (2) browsing and modifying the attributes of MPEG-4 scene nodes and the content of MPEG-4 elementary streams, and (3) creating/deleting MPEG-4 scene nodes and elementary streams.

The MPEG-4 fundamental concepts split the design of the low-level API into two components, namely **SceneAPI**, which handles the access to scene information and scene nodes, and **MediaAPI**, which manipulates elementary streams. Fig.3 illustrates the user scenario when using the SDK, for both SceneAPI and MediaAPI.

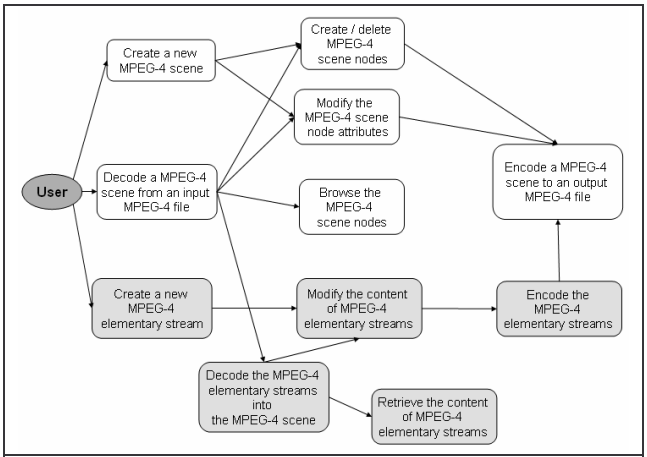

Fig.3 – User scenario for MPEG-4 SDK: in white the components of the SceneAPI and in gray the components of the MediaAPI.

Implementing an MPEG-SDK supporting all the scene nodes and elementary streams would be a useless task since it does not exists any application that can use all the nodes. A real MPEG-4 application is likely to use only a few of the approximately 200 scene nodes and 10 elementary streams defined by the standard. This is why MPEG-4 defines the concept of *profile* [7] which represents a subset of tools that can be used for a large class of applications and services. The "SDK Generator" software component uses MPEG-4 profiles as input data to produce a reduced-size MPEG-4 SDK that fits the chosen profile (Fig.4).

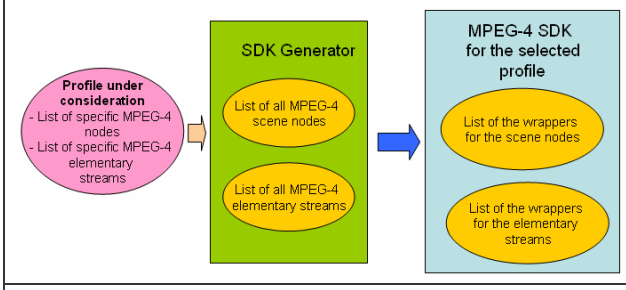

Fig.4 – MPEG-4 SDK generator.

2.2 High Level API: Virtual Character SDK

The **high-level APIs** add new processing functions and simplify the access to the MPEG-4 objects. These APIs are designed for the developers who are not familiar with MPEG-4 concepts and need to rapidly integrate the MPEG-4 content into their software platforms. We designed such API for the specific case of defining and animating a VC. The main concept used by MPEG-4 for representing a virtual character is the *skeleton*, defined as a hierarchical graph of bones. By attaching the skeleton to the VC shape (referred to as the *skin*) and by allowing to apply geometric transformations to the bones, deformations of the skin are possible. The animation framework, called Bone-based Animation (BBA), offers high visual quality of the representation and the animation, and is a generic approach suitable for any kind of articulated figure (animals, humans and plants). The features provided by the Virtual Character SDK, defined on the top of BBA specifications, are described below:

Data	**Operations**
Scene	- encode / decode the entire scene - access the VC model list - encode / decode / modify the animation stream
Model	- get / set the model name - access the bone and root bone hierarchy - add / remove skin shapes
Bone	- access the parent model - get / set the bone name - retrieve the bone ID - access the parent bone - get / set the transformation matrix - access the list of influenced points with corresponding weights - access the children bone list
Skin shape	- access the parent model - retrieve the appearance of the shape (material and/or texture) - access the list of polygons - access the list of edges
Skin polygon	- retrieve the parent shape - access the list of edges - access the list of neighborhood

	relationships: polygons, edges
Skin edge	- access the parent shapes - access the list of neighborhood relationships: polygons, edges
Skin vertex	- retrieve the list of parent shapes - access the list of bones that influence it with corresponding weights - access the list of neighborhood relationships: polygons, edges

2.3 Implementation issues

The MP4SDK is a C++ library based on the MPEG-4 Reference Software (IM1). Our current implementation is generated by using the "Animated Character" profile. This profile, currently under consideration within the MPEG community, selects 11 nodes in the scene graph and 2 streams from the media layer. The components of this profile (the list of scene nodes and elementary streams) and the corresponding SDK classes are described in Fig.5.

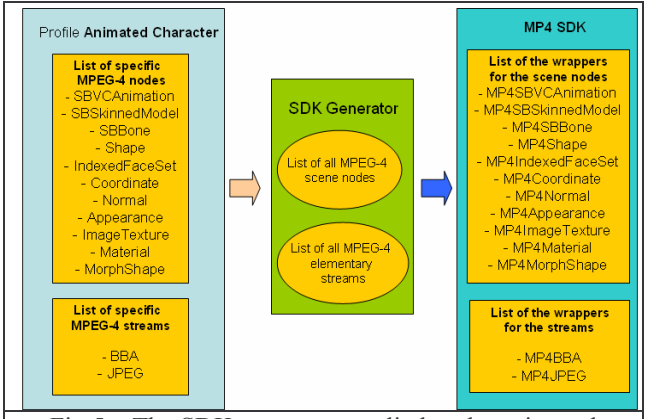

Fig.5 – The SDK generator applied to the animated character profile.

2.3.1 MPEG-4 SDK

The main class of the MPEG-4 SDK is *MP4Engine*. It is in charge of: loading/saving a MPEG-4 file, decoding/encoding the MPEG-4 scene, creating/deleting MPEG-4 wrapper nodes, filtering the access to MPEG-4 nodes and to elementary streams. It is derived from the IM1 *Application* class (Fig.6).

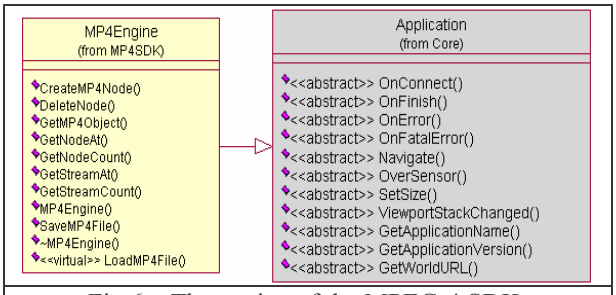

Fig.6 – The engine of the MPEG-4 SDK.

SceneAPI represents the wrapper over the scene nodes declared in the MPEG-4 Reference Software (IM1). The wrapping mechanism is the following: the SDK core class, *MP4ProxyNode*, contains a reference to the core class of IM1 scene node, *MediaObject*. Then, every wrapper node is created by deriving the *MP4ProxyNode* class and casting the *MediaObject* reference to the specific node. The example below shows the creation of the *MP4SBSkinnedModel* wrapper node (Fig.7).

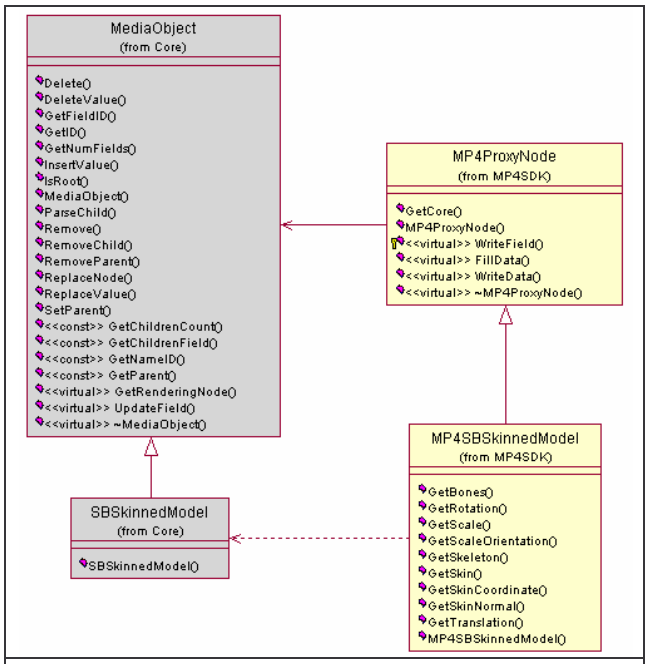

Fig.7 – An example of node wrapping mechanism.

MediaAPI represents the wrappers over the MPEG-4 elementary streams. The wrapping mechanism is described as follows: the SDK core stream class, *MP4Stream*, contains a reference to the IM1 core stream consumer class, *StreamConsumer*. Every wrapper stream is created by deriving the base class *MP4Stream* and overwriting its methods. An example of creating such a stream wrapper is presented below (Fig.8).

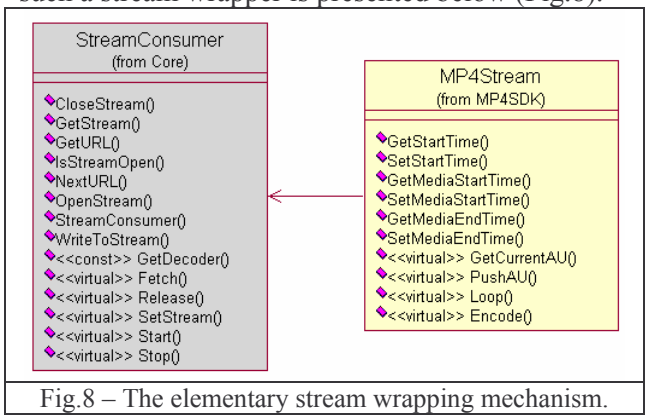

Fig.8 – The elementary stream wrapping mechanism.

2.3.2 VC SDK

The VC SDK adds an extended layer over the MPEG-4 SDK by providing virtual character specific functionalities. New classes for skin edge

and skin polygon are integrated into the API. An example of virtual character wrapper over the MPEG-4 SDK class (*MP4SBSkinnedModel*) is shown in Fig.9.

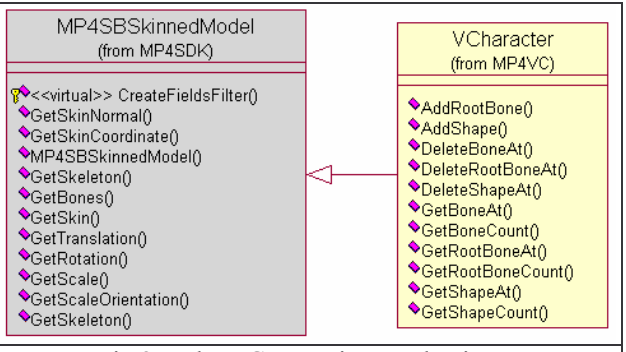

Fig.9 – The VC wrapping mechanism.

3 MP4SDK applications

As an example of using the MP4SDK, loading an MPEG-4 file is performed by *MP4Engine* class, calling the method *LoadMP4File(fileName)*. Then, one can retrieve all the nodes of a specific type, by calling *GetNodesByType(nodeType)*. After updating the scene nodes, a new MPEG-4 file can be created by *SaveMP4File(fileName)*.
In order to validate the developed SDKs, we implement a 3dsmax to MPEG-4 import export module.

3.1 MPEG-4 plug-in for 3dsmax

One of the most popular authoring tools (AT) for games and for 3D content in general is 3D Studio Max published by Discreet. This software package is an ongoing platform, enriched at each version with new features. Moreover, its architecture makes possible the development of plug-ins by independent parties. By using high-level and interactive tools, this software packages afford the creation and the animation of complex 3D objects and scenes. Although very advanced with respect to specialized tools and ergonomic interfaces, 3dsmax does not include mechanisms for the compression of the geometry and the animation parameters. On the other hand, the MPEG-4 standard had provided, in its first version, -1998 - the compression of the geometry based on indexed face set. Later on, in 2003, the MPEG community published the MPEG-4 Part 16 which includes specifications for modeling and animating of virtual characters, including the compression of the animation parameters. By building an import/export module, we are focusing on mapping between the high level tools provided by 3dsmax and the low level representation provided by MPEG-4.

Using a set of seven 3dsmax models, we have validated the exporting of static attributes (geometry, materials and textures) and dynamic attribute (the animation stream), performing a comparative visualization using 3dsmax versus the MPEG-4 Player [5]. Similar results have been obtained. The importer module was validated by using two models previously exported from 3dsmax. Finally, the relevance of the animation was validated by playing the imported and the original content in the 3dsmax environment and the exported content by using the MPEG-4 Player. No differences were noticed between the two playbacks. Some frames from the animation sequence, obtained using the MPEG-4 Player and 3dsmax, are shown in Fig.10.

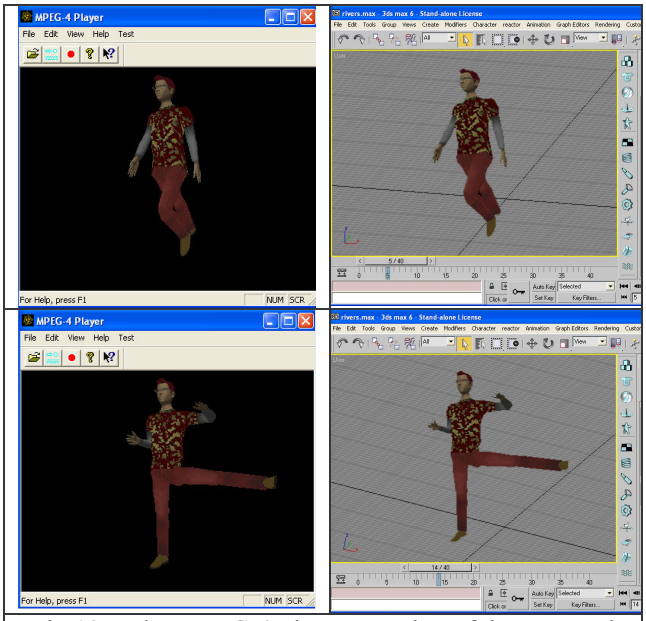

Fig.10 – The MPEG-4 Player snapshot of the exported model[1].

4 Conclusion

In order to facilitate MPEG-4 usability for non-expert developers, this paper addressed the technical issues related to the implementation of an MPEG-4 Software Development Kit (SDK). By focusing on the MPEG-4 scene composition level and on the graphics representation features, we described the main concepts of a low-level MPEG-4 SDK (scene graph access, media and stream processing) compliant with scene and media specifications. The more specific issue of virtual character animation was addressed by developing an intuitive API referred to as MPEG-4 VC API and supporting high-level functionalities (data-based semantic access, hierarchical object processing and stream control). We demonstrated

[1] With courtesy of EPFL-VRLab for the initial model.

the relevance of this toolkit for the easy design and creation of real applications by implementing a 3dsmax plug-in for importing/exporting MPEG-4 content and a virtual character scalability engine. In developing these applications, no particular knowledge of the MPEG-4 standard was required. Our future work will deal with the development of a streaming client/server system based on MP4SDK which is able to transmit and play MPEG-4 scalable content.

References

[1] ISO/IEC 14496-1:2001 Information technology: Coding of audio-visual objects, Part 1: Systems, *International Standard Organization (ISO)*, Switzerland, 2001.

[2] ISO/IEC 14772-1:1998, Information technology: Computer graphics and image processing, The Virtual Reality Modeling Language, Part 1: Functional specification and UTF-8 encoding, *International Standard Organization (ISO)*, Switzerland, 1998.

[3] ISO/IEC 14496-5:2001 Information technology: Coding of audio-visual objects, Part 5: Reference Software, *International Organization for Standardization*, Switzerland, 2001.

[4] M. Bourges-Sevenier, F. Moran, M. Steliaros, M. Preda, M. Han (Eds), ISO/IEC 14496 [MPEG-4] Part 16: Animation Framework eXtension (AFX) *International Standard Organization (ISO)*, Switzerland, 2003.

[5] MPEG-4 3D Player, http://www-artemis.int-evry.fr/~preda/MPEG-4/.

[6] Cyber VRML 97, http://www.cybergarage.org/vrml/cv97/cv97cc/index.html.

[7] F. Pereira, T Ebrahimi, *The MPEG-4 book*, IMSC Press Multimedia Series/Andrew Tescher, 2002.

[8] Alias FBX, Universal 3D Asset Exchange, http://www.alias.com/eng/products-services/fbx/

[9] Apple Keynote, http://www.apple.com/iwork/keynote/