A MPEG-4 Virtual Human Animation Engine for Interactive Web Based Applications

Mario Gutiérrez, Frédéric Vexo and Daniel Thalmann
Virtual Reality Lab (VRlab)
EPFL (Swiss Federal Institute of Technology)
1015 Lausanne, Switzerland
tel: +41-21-693-5216 fax:+41-21-693-5328
e-mail: Mario.Gutierrez, Frederic.Vexo, Daniel.Thalmann@epfl.ch

***Abstract*— This paper presents a novel, MPEG-4 compliant animation engine (body player). It has been designed to synthesize virtual human full-body animations in interactive multimedia applications for the web. We believe that a full-body player can provide a more expressive and interesting interface than the use of animated faces only (talking heads). This is one of the first implementations of a MPEG-4 animation engine with deformable models (it uses the MPEG-4 Body Definition Parameters and Deformation Tables). Several potential applications are overviewed. This software tool was developed in the framework of the IST-INTERFACE European project [18].**

I. Introduction

This paper describes a new tool designed to improve the human-computer interaction by means of non-verbal communication (body gestures). The developed software is a multiple platform, real-time tool, aimed to produce realistic full-body animation of 3D virtual humans on web applications. Using body gestures, the virtual human will be able to express natural human reactions, such as emotions, giving the illusion that the user is interacting with a real person inside the computer.

One of the most important features we tried to include in the developed tool was the capability to synthesize full-body animation instead of limiting it to face animation.

The ease of integration in a web-based application and the user-friendly installation procedure were also important goals to achieve.

In the next section we overview some of the existing implementations to provide a human-like character interfaces on web applications (talking heads, virtual characters) and justify the innovations of our tool. After, we detail the technical requirements for the body animation engine and justify the technology selected for the implementation of our tool.

In the third section, we describe the general architecture of the MPEG-4 player developed. The last part presents a demonstration application which provides the synthesis of body emotional gestures. The conclusions discuss the potential applications of the tool and the future improvements.

II. State of the Art

It has been stated by several authors that the ultimate human-computer interface would include audio/video analysis and synthesis in combination with artificial intelligence (AI) techniques, dialog management and face/body gestures to allow an ”intelligent” and expressive dialog with the user [23], [34]. In other words, a full autonomous or semiautonomous software agent which will look, move, listen, speak and make conversation as a real person. This simulation of a human being inside the computer (human-like character interface) would serve as an assistant to the real human users. The natural target for this new interface paradigm are web based applications where the computer generated humans will be used to present information, help in the search and interaction with on-line services, etc. The following is an overview of some representative implementation examples of this approach.

A. Human-like Character Interfaces on the Web

The most common implementations of this kind of user interface consist mainly of animated faces (talking heads) which use some variation of text-to-speech and dialog manager software to drive a somehow ”intelligent” conversation with the user or to present some information in an oral way. Examples of talking heads which help in delivering their web-site’s information include the Ananova web site which features a video news report with a computer generated character (face animation) using text-to-speech synthesis [1]. Nevertheless, the user interaction is very limited. A similar application used to present and search contents can be found on the KurzweilAI web site [21]. This site features a more dialog-oriented interactive talking head. Even though the interactive level is higher, the character expressiveness is limited to facial gestures. There are other examples of the talking head approach: [22], [35], [31], [11], [37]. Many of them feature good animations and synthesized or prerecorded voices which help in reaching the objective of creating a ”human inside the computer”. However, they still have many limitations in the areas of speed and visual appearance, intelligent dialog management and interactivity in general. Among the areas to improve, a very important missing detail is the capability to show full body expressions and gestures. We propose to extend the talking-head model to a full-body virtual human which will be able to complement the text-to-speech voice synthesis, dialog management and facial animation with body ges-

tures to enhance the expression and give a more human touch to the virtual character through the use of non-verbal communication. [12], [36], [30]

The following section deals with the technical requirements of the full body animation engine.

B. System Requirements

In this section we define the system requirements for the tool (henceforth called *the player*). We also justify the software tools selected to develop the player.

The system requirements are specified as follows:

-System characteristics: the new player must be executable across multiple platforms. There should be no need for plug-in installation or additional downloads. Expected performance: the player should be able to render realistic 3D virtual humans at a minimum speed of 10 frames per second, to provide 'natural' animation.

- Data representation and communication: The player requires a 3D Geometric description of the virtual human, and a set of animation parameters. Both of them should be sent over the network to the player, the animation parameters must be continuously updated in order to keep the virtual human 'alive'. The 3D Geometric description should include information to calculate anatomical deformations in real-time. The required high quality results should be obtained with a low bit rate data transfer. It is mandatory to use a very efficient representation of the data (the 3D content).

The two main standard representations of 3D content for web applications are the VRML language and the MPEG-4 specification [6]. MPEG-4 adopts a VRML based standard for virtual humans representation: H-Anim, and provides an efficient way to animate virtual human bodies [7], [8].

MPEG-4 is a complete solution because it does not only standardize the definition of the the shape and surface of a model (geometry representation) and anatomic deformations, but also defines an efficient and flexible way to animate virtual humans with low bit rate data transfers: the body animation parameters [7], [8], [26]. Since it has been accepted as the new ISO-IEC standard to provide interactive multimedia to new networks, including those employing relatively low bit rate, and mobile ones [24], being MPEG-4 compliant will allow the player to be a standard, efficient 3D application [19].

To avoid the development of a new software system, we could have adapted an existing MPEG-4 compliant player. However, the offer of available web-enabled MPEG-4 players is limited. The existing solutions from Philips, and other developers provide either stand-alone players or plug-ins for specialized browsers [28], [25] [5]. All of them are oriented to display video and are not designed to support interactive applications. One of the latest products is the full MPEG-4 compliant player developed in the framework of the IST-SONG project [5], [15]. But this player does not fulfil our system requirements. Its main drawback is that it has been developed using C/C++ based components which avoids its direct execution in multiple platforms, and has to be implemented as a plug-in [16]. Moreover, this player does not implement anatomical deformations, it works only with rigid non-deformable bodies. The anatomical deformations are a very important characteristic to include if the goal of a realistic graphical representation is to be achieved.

There are some other applications aimed to produce an MPEG-4 player for interactive applications, but usually they focus on face animation, as we have already mentioned [32], [27].

Since there were no MPEG-4 players which seem to fulfil our needs, we decided to develop our own MPEG-4 body player.

In order to reach the multi platform execution and to avoid the need of plug-in installation, the best choice was to make use of the java language, which has proved to be one of the most reliable solutions. Java adds complete programming capabilities plus network access. Most of the web browsers for the main operative platforms are java enabled [6]. There are applications that demonstrate the possibilities of the java language to synthesize animation of 3D virtual humans on the web [3], [4], such as the player for H-Anim bodies developed by C. Babski [2]. Although this developments have not implemented an MPEG-4 body player ready for general use.

For the new player to be a java-based application it is essential to have a pure java 3D rendering engine. Since the development of a new 3d rendering engine for web applications was out of our scope, we decided to use an existing one: shout3D [33]. The shout3D rendering engine has been written in pure Java. It effectively avoids the plug-in installation and provides a real-time 3D animation at the required frame rate. The render performance can be improved by means of installing a plug-in to use openGL hardware acceleration. Although it's a commercial product, it was chosen due to its and good performance and low price (US$100 per year, to be paid by the content provider, and only if the shout3D logo is to be removed, otherwise it can be used for free) [33].

In conclusion, we defined the tool to be developed as follows: the body player will be a pure java applet which will use MPEG-4 compliant data streams to represent the 3D content. The 3D rendering of the scene will be done using the shout3D java engine. Such a system will be able to execute on multiple operative platforms without any plug-in installation or additional download, providing a minimum performance of 10 frames per second of 3D realistic virtual humans animation. In the following section, we describe the MPEG-4 streams used by the body player, and after we present the system architecture.

III. MPEG-4 STREAMS FOR BODY ANIMATION

In this part we describe in detail the MPEG-4 streams used by the player.

The developed body player uses the MPEG-4 FBA (Face and Body Animation) object to describe the geometry of the virtual human and animate it.

MPEG-4 defines two sets of parameters: The first one specifies the geometry of the FBA model: FDPs (Face Definition Parameters) and BDPs (Body Definition Parameters). These parameters allow the decoder to create an FBA model with the specified shape and texture. The second set defines the animation of the face and body: FAPs (Face Animation Parameters) and BAPs (Body Animation Parameters). The body animation parameters are a set of rotation angles to be applied to the different parts of the body in order to modify the posture [8], [7], [24].

In our player we consider the face as part of the body and don't use the specific parameters for definition and animation of the face. However the design of the tool allows the integration of a face animation module. The figure 1 shows a schematic view of the FBA object.

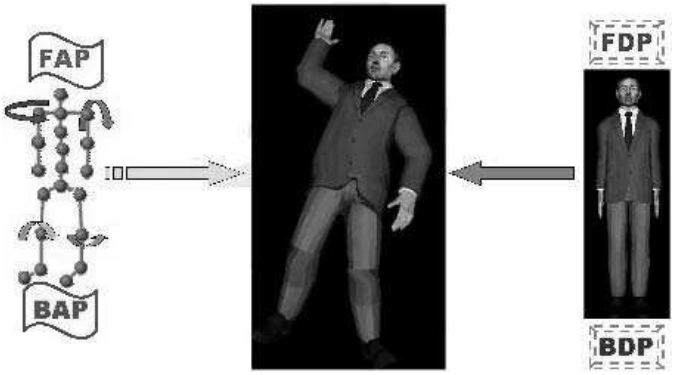

Fig. 1. The MPEG-4 Face and Body Animation object

Now we present a description of the body definition and animation parameters:

A. MPEG-4 Body Definition Parameters

The MPEG-4 BDPs used to define the body of the virtual human, are directly related to the VRML/Web3D H-Anim 1.1 specification [7]. According to the H-Anim standard, the human body consists of a number of segments (such as the forearm, hand and foot) which are connected to each other by joints (such as the elbow, wrist and ankle). A H-Anim file contains a set of Joint nodes that are arranged to form a hierarchy (figure 1). Each Joint node can contain other joint nodes, and may also contain a segment node which describes the body part associated with that joint. Each segment is a normal VRML transform node describing the 3d shape of the body part [13]. The Segments can also have a number of site nodes, which define locations relative to the segment. The Sites can be used for attaching clothing and jewelry, and can also work as end-effectors for inverse kinematics applications. They can also be used to define eye points and viewpoint locations. A Segment node may contain a number of displacer nodes, that specify which vertices within the segment correspond to a particular feature or configuration of vertices. The H-Anim file contains a single Humanoid node which stores human-readable data about the humanoid, such as author and copyright information. That node also stores references to all the joint, segment and site nodes, and serves as a "wrapper" for the humanoid. In addition, it provides a top-level Transform for positioning the humanoid in its environment. In the developed tool we are not using site nodes, nor displacers, in order to simplify the BDP file, although, the stream is still compliant with MPEG-4.

B. MPEG-4 Body Animation Parameters

The MPEG-4 BAP (Body Animation Parameters) are used for the synthesis of body movements. In order for the tool to animate the virtual human, it obtains access to the H-Anim joints and alters the joint angles to match those defined in a BAP stream. A BAP file/stream can contain up to 296 parameters describing the topology of the body skeleton. For most joints in the skeleton there are 3 different angle parameters to describe in which way the joint is oriented (yaw, roll, pitch). The BAP's are grouped into 24 different categories, dividing the human body into segments [7], [9].

The BAP unit is defined as:

$BAP = angleInRadians * 100000/\pi$

A BAP file has the following structure:

BAP mask: 0 or 1 for each of the possible 296 BAPs (masked or unmasked) indicating whether the BAP is used. The BAP mask precedes each BAP data line on a separate line. BAP data description: each frame is described as one line of integer BAP values starting with the frame number. The BAP mask in the header indicates which BAP groups are included. The 296 parameter values are defined in MPEG-4 Version 2 visual specification. A 0 value for any BAP indicates the default position [7], [9].

This is as example of a bap file called new01.bap with a frame rate of 25 fps and 1 frame of animation:

```
3.2 new01.bap 25
1 1 0 1 0
1542 1542 -3483
```

When the virtual human body defined in the BDP is animated by a BAP stream we most consider that the segment's geometry of the affected joints needs to be deformed. This special consideration is solved by means of the anatomical deformations algorithm, defined as follows.

C. Anatomical deformations

Since the BDP model is made out of separated segments, defined by a mesh of polygons, when the rotation angles of the joints are modified by the applied BAPs, the children segments are rotated from its original position. The result is a discontinuous surface over the rotated joint (low visual quality). The set of vertices that form the polygon mesh of the segment, need to be modified to keep an anatomically realistic, continuous surface over the modified joint.

MPEG-4 defines the use of preset deformation tables, which contain the coordinates of the segment vertices to be deformed in the key postures of each joint. The deformations of the segments on intermediate postures (not contained in the preset tables) are calculated using linear interpolation.

The BodyDefTable node specification is defined in the MPEG-4 version 2 standard, and is an extension of the FaceDefTable node. Multiple BAPs can affect the same vertex without equal coefficients (for example, the upper arm segment vertices may be deformed, based on elbow and shoulder BAPs). It is not enough to add the displacements from different tables. Therefore, the BodyDefTables use multiple BAPs to control vertices. A BodyDefTable contains 'key deformations', each key consisting of a combination of selected BAPs. An example of the data contained in a BodyDefTable is shown in figure 2, [7].

BAPs				**Vertices**			
BAP_1	BAP_2	...	BAP_k	$Vertex_1$	$Vertex_2$	...	$vertex_N$
0	0	.	0	0	0	.	0
0	0	.	100	D_{11}	D_{21}	.	D_{N1}
0	100	.	0	D_{21}	D_{22}	.	D_{N2}
...							

Fig. 2. Body Deformation Table data.

Each entry in the BodyDefTable corresponds to a combination of BAPs. If the current body posture is not one of these BAP combinations, the vertex deformations have to be interpolated based on these BAP combinations. A linear interpolation technique solves this problem. BAP keys (rows in Table) are represented as k-dimensional points, where k is the number of BAP columns. Given several BAP keys $P_1, P_2, P_3, ...P_n$, computing linear interpolation at BAP point P. n represents the number of key positions to be used for interpolation, specified in a field of the BodyDefTable node [7].

Let $d_1, d_2, d_3...d_n$ be respective distances from P to keys. Let $v_1, v_2, v_3, ..v_n$ be tabular displacement values of a vertex at the keys.

For any key P_i, the deformation contributed by P_i should be inversely proportional to distance d_i from point P.

Let this proportionality factor be f_i.

Thus,

$DEF_i = f_i * v_i$ (deformation due to P_i)

$DEF = f_1 * v_1 + f_2 * v_2 + ... + f_n * v_n$ (Total deformation at P)

With the condition that $f_1 + f_2 + ... + f_n = 1.0$,

$D = d_1 + d_2 + ... + d_n$ (total distance)

$f_i = (1 - d_i/D)/(n - 1)$

We have specified the MPEG-4 streams used in our player to produce the animation of a virtual human. The next part will describe the architecture.

IV. System architecture

Figure 3 gives a general view of the system architecture. The player has two main input files, the MPEG-4 streams that define the geometry of the virtual human (BDP file), which includes the information for the anatomic deformations (BDT), and the body animation parameters (BAP file) which contain information to alter the virtual human joints and display the animation.

The core of the system is a java package (set of classes) constituted by two main sections: the MPEG-4 data management and animation classes (BAPReader and HAnimBAP), and the graphics rendering classes, which are directly related to the shout3D engine (BDPLoaderPanel and BDPLoaderApplet). The last class, BDPLoaderApplet is the external component that must be embedded in the web-based application, for the player to be used.

This is a more detailed description of the data management and animation classes:

- BAPReader: this is the data management class. It's used for BAP files management. Its function is to read and parse BAP streams from a local file system or from a network location. It incorporates as well the capability to alter the BAP values of a stream according to an intensity factor (this will be explained in the applications section).

- HAnimBAP: this class is responsible for the synthesis of the body animation using the BAP streams parsed by the BAPReader class. It receives the scene graph (BDP) loaded by the BDPLoaderPanel class and alters the positions of the segments to provide animation. The routines for anatomical deformations, are included in this class.

The BDPLoaderPanel and BDPLoaderApplet classes are derived from prototype classes implemented in the Shout3d library. The BDPLoaderPanel class derives from the Shout3dPanel class. It allows to implement scene navigation and is the place where the scene graph (BDP stream) is loaded and rendered, once it has been modified (animated) by the HAnimBAP class.

The BDPLoaderApplet class derives from the basic Shout3dApplet class and implements the applet that will be loaded by the browser, it constitutes the container of the body player. This is the only class that must be explicitly called from a web application in order to use the player. It implements public methods which may be called from the user interface to control the animation and the load of models.

The set of data management and animation classes constitutes the so called motion management library, whose main objective is to provide the functionality to blend and control the execution of different BAP streams (motions). This library provides the capability of processing BAP streams either from local files or from a given URL, thus the tool is ready to work in cooperation with a server which could send it new BAP streams in real time.

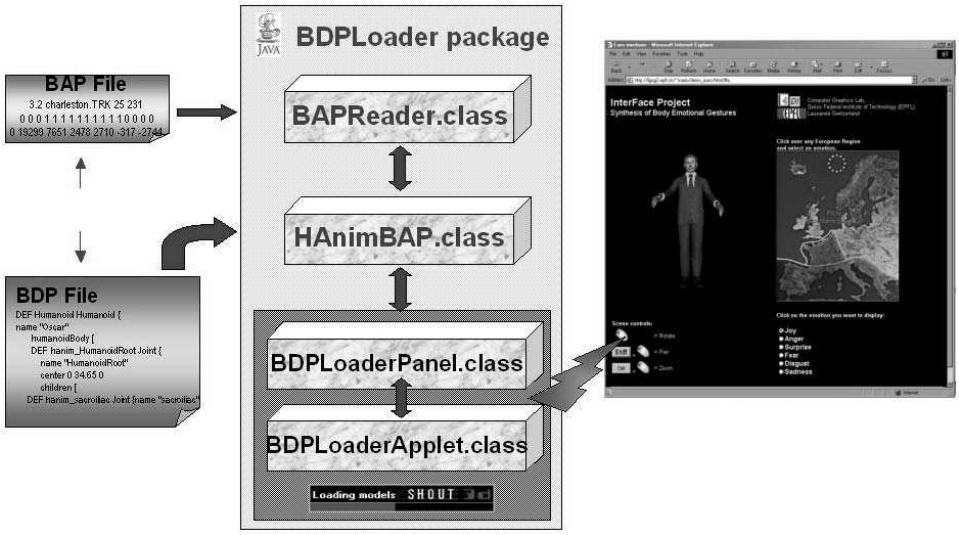

Fig. 3. System architecture of the body player.

The motion management library requires a front-end component to provide user interface controls, if required by the application, such as scene navigation (zoom and panning of the virtual human scene) and controls to select the virtual human to display and the specific animation (BAP file) to execute.

The front-end interface can be a html file that implements a graphical user-interface and serves as a container for the body player implemented in a java applet. The communication with the applet can be done by means of JavaScript calls to the public methods of the applet container (the BDPLoaderApplet class).

In the following part, we describe a specific real application of the MPEG-4 body player.

V. Applications

The described MPEG-4 body player has been integrated in a demonstration application developed in the framework of the IST-INTERFACE project [18]. One of the main goals of the project is the synthesis of emotions. The demonstration is a web-based application whose main objective is to synthesize human body emotional gestures corresponding to the 6 basic universal emotions: joy, sadness, anger, surprise, disgust and fear. The idea of the demonstration is that the emotional behavior of people from each region of the world can be distinguished by the intensity levels of their emotional gestures. In particular, the aim of the demonstration is to present the different intensity levels of emotions that can show European people from the north, center and south. Assuming that the northern people are less expressive than their southern counterparts. The emotional gestures are performed by three different virtual humans (one for each European region).

Figure 4 show the differences in the intensity level for the same emotion: anger.

This demonstration application can be viewed on any java-enabled web browser and is available on the interface project web page [17], under the on-line demo link, or in the following address: http://vrlab.epfl.ch/~mario/demo_euro/html.

Fig. 4. Different intensity levels for the same emotion.

The performance of the body animation depends on the complexity (number of polygons) of the model, we have tested two models with different levels of complexity. According to the built-in monitoring routines in shout3D, the table in figure 5 shows the performance in terms of frames per second (fps) that the player is capable to animate. The tests where done on a notebook PC (HP OmniBook XE3) with an Intel PIII processor running at 900 MHz, 256 Mb of RAM, S3 Graphics Savage/IX graphics card with 8 Mb of memory, using MS-Internet Explorer 6 on MS-Windows 2000.

Numb. polygons	fps	fps with OpenGL HW acceleration
10794	30	40
119018	10	15

Fig. 5. Performance measured in frames per second of animation.

VI. Conclusions

In this paper we have presented a real-time MPEG-4 virtual human body animation engine (body player) designed to be integrated as a user interface in web-based applications. It uses MPEG-4 streams to animate 3D human models with anatomical deformations at a minimum rate of 10 frames per second (highly detailed model) on a standard PC (see Applications section).

This tool provides a new way to implement a user interface by means of a virtual human that can establish non-verbal communication using body gestures and work in coordination with text-to-speech, dialog managers and other interaction methods as well. Early tests to integrate the tool with the above cited technologies and MPEG-4 facial animation are in progress and will be reported in future papers.

The potential uses for which the player is ready include: interfaces and assistance systems for mobile applications for Personal Digital Assistants (PDA) such as the new generation of java enabled [20] PocketPC devices [14], [10]. For this, a light-model (less than 5000 polygons) will be used.

An application developed in the framework of the IST-INTERFACE project was presented as well. This work is still in progress.

Acknowledgements

This work has been supported by the Swiss Federal Office for Education and Science in the framework of the European project IST-INTERFACE.

References

[1] Ananova Ltd, http://www.ananova.com

[2] C. Babski, *MPEG4 Player for HANIM 1.1 Compliant VRML Body*, http://ligwww.epfl.ch/~babski/StandardBody/mpeg4

[3] C. Babski and D. Thalmann. *3D on the WEB and Virtual Humans*. Software Focus, Wiley, Vol. 1, No. 1, 2000. http://ligwww.epfl.ch/~thalmann/papers.dir/wiley.pdf

[4] C. Babski and D. Thalmann. *Real-time animation and motion capture in Web Human Director*. Proc. Web3D & VRML2000 Symposium, Monterey CA, february 2000.

[5] *MPEG-4 player*, Blaxxun interactive AG. http://www.blaxxun.com/solutions/research/song.shtml

[6] D. Brutzman. *The Virtual Reality Modeling Language and Java*. Communications of the ACM, Vol. 41, No. 6, june 1998.

[7] T. K. Capin, E. Petajan and J. Ostermann. *Efficient modeling of virtual humans in MPEG-4*. IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA AND EXPO (ICME), NEW YORK, NY, Volume: 2, 2000.

[8] T. K. Capin, E. Petajan and J. Ostermann. *Very Low Bitrate coding of virtual human animation in MPEG-4*. IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA AND EXPO (ICME), NEW YORK, NY, Volume: 2, 2000.

[9] T. K. Capin and D. Thalmann. *Controlling and Efficient Coding of MPEG-4 compliant avatars*. International Workshop On Synthetic - Natural Hybrid Coding And Three Dimensional Imaging (IWSNHC3DI'99), Santorini, Greece, 1999.

[10] *Casio Inc. PDA devices.* http://www.casio.com

[11] Extempo Systems Inc., http://www.extempo.com

[12] Grammalidis, N.; Goussis, G.; Troufakos, G.; Strintzis, M.G. *Estimating body animation parameters from depth images using analysis by synthesis.* Proceedings of the Second International Workshop on Digital and Computational Video, 2001.

[13] Web3D Working Group on Humanoid Animation, *Specification for a Standard Humanoid*, Version 1.1, August 1999. http://h-anim.org/Specifications/H-Anim1.1

[14] *HP-Compaq handheld devices.* http://www.hp.com

[15] Information Societies Technology Programme, *IST-SoNG project web page* http://www.octaga.com/SoNG-Web

[16] Information Societies Technology Programme, *Project SoNG: Portals of next generation*, Annex 1 - Description of Work, January 11th 2001.

[17] Information Societies Technology Programme, *IST-ITERFACE project web page*, http://www.ist-interface.org

[18] Information Societies Technology Programme, *Project INTERFACE: Multimodal Analysis/Synthesis System for Human Interaction to Virtual and Augmented Environments*, Annex 1 - Description of Work, October 15th 1999.

[19] Jang, E.S. *3D animation coding: its history and framework.* IEEE International Conference on Multimedia and Expo, Volume: 2 , 2000.

[20] *Jeode Java Virtual Machine* Insignia Solutions Inc. http://www.insignia.com/content/products/otherjeode.shtml

[21] KurzweilAI.net, http://www.kurzweilai.net

[22] Lexicle Limited, http://www.lexicle.com

[23] Magnenat-Thalmann, N.; Kshirsagar, S. *Communicating with Autonomous Virtual Humans.* Proceedings of the Seventeenth TWENTE Workshop on Language Technolgy , Enschede, Universiteit Twente, October 2000, pp 1-8.

[24] *MPEG-4 Overview*. ISO/IEC JTC1/SC29/WG11 N4030, March 2001, http://mpeg.telecomitalialab.com/standards/mpeg-4/mpeg-4.htm

[25] *NetFront web browser.* ACCESS CO.,LTD. http://www.access.co.jp/english/products/pdf

[26] J. Ostermann and E. Haratsch. *An animation definition interface: Rapid design of MPEG-4 compliant animated faces and bodies*. International Workshop on synthetic - natural hybrid coding and three dimensional imaging, Rhodes, Greece, Spetember, 1997.

[27] Petajan, E. *The communication of virtual human faces using MPEG-4 tools*. Proceedings of the IEEE International Symposium on Circuits and Systems, Volume: 1 , 2000.

[28] *WebCine MPEG-4 player*. Philips Electronics North America Corporation. http://www.mpeg-4.philips.com/downloads/player/index.asp

[29] *PocketPC, Software for mobile devices.* Microsoft Corporation. http://www.microsoft.com/mobile/pocketpc/

[30] Prakash, E.C. *A human touch for Internet training: the Virtual Professor*. TENCON 99. Proceedings of the IEEE Region 10 Conference , Volume: 2 , 1999.

[31] Sensory, Inc., http://www.sensoryinc.com

[32] Shin, M.C.; Goldgof, D.; Kim, C.; Jialin Zhong; Dongbai Guo *Customizable MPEG-4 face player using real-time 2D image sequence*. Fifth IEEE Workshop on Applications of Computer Vision, 2000.

[33] *Shout3D rendering engine.* Eyematic Interfaces Incorporated. http://www.shout3d.com

[34] Smid,K; Pandzic, I. *A Conversational Virtual Character for the Web*. Computer Animation 2002, Geneva, Switzerland, June 19-21, 2002.

[35] TELEVIRTUAL Ltd., http://televirtual.com

[36] Todesco, G.; Araujo, R.B. *MPEG-4 support to multiuser virtual environments*. Proceedings of the 20th International Conference on Distributed Computing Systems, 2000.

[37] *W Interactive SARL*, http://www.winteractive.fr