XfaceEd: Authoring Tool for Embodied Conversational Agents

Koray Balcı
ITC-irst
Cognitive and Communication Technologies Division
Trento, Italy
balci@itc.it

ABSTRACT

In this paper, XfaceEd, our open source, platform independent tool for authoring 3D embodied conversational agents (ECAs) is presented. Following MPEG-4 Facial Animation (FA) standard, XfaceEd provides an easy to use interface to generate MPEG-4 ready ECAs from static 3D models. Users can set MPEG-4 Facial Definition Points (FDP) and Facial Animation Parameter Units (FAPU), define the zone of influence of each feature point and how this influence is propagated among the neighboring vertices. As an alternative to MPEG-4, one can also specify morph targets for different categories such as visemes, emotions and expressions, in order to achieve facial animation using the keyframe interpolation technique. Morph targets from different categories are blended to create more lifelike behaviour.

Results can be previewed and parameters can be tweaked real time within the application for fine tuning. Changes made take into effect immediately, which in turn ensures rapid production. The final output is a configuration file XML format and can be interpreted by XfacePlayer or other applications for easy authoring of embodied conversational agents for multimodal environments.

Categories and Subject Descriptors

I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*virtual reality*; H.5.1 [**Information Interfaces and Presentation**]: Multimedia Information Systems—*artificial, augmented, and virtual realities*

General Terms

Design, Standardization, Performance

Keywords

Talking heads, 3D facial animation, MPEG-4, Open Source, Embodied Conversational Agents

ICMI'05, October 4–6, 2005, Trento, Italy.

1. INTRODUCTION

The use of 3D talking heads has gained wider popularity in various fields including the entertainment industry, video games, customer service, human computer interaction and virtual reality. Unfortunately, there is a lack of free and open source tools for creation of animated agents. With this in mind, we have started an open source initiative, which will provide the research community with a stable and extensible platform for generating and animating 3D talking agents, namely Xface [1]. Xface toolkit relies on MPEG-4 Face Animation (FA) standard [12] and targets researchers working on similar topics as well as developers in software industry. In this paper, we present XfaceEd, our editing tool for authoring MPEG-4 ready faces.

XfaceEd is primarily designed for tagging and testing of muscle models on the face based on MPEG-4 FA. One can import one or more 3D models in various formats, define the regions and parameters to deform the model according to the standard, then test the quality of animation immediately. The resulting information is stored in a configuration file in XML, so that it can be used by other possible future MPEG-4 face player implementations, as well as Xface core library and XfacePlayer of our toolkit. In addition, XfaceEd is open source as the entire Xface toolkit, thus one can easily derive her own application according to her needs when necessary.

XfaceEd also supports keyframe animation as an alternative facial animation technique by using various morph targets representing visemes, emotions and expressions and let more freedom to people working in ECA development field.

The paper is organized as follows. In the following section, an overview of MPEG-4 FA standard is given. Next, XfaceEd is described in detail. First, we overview the basic capabilities of the editor. Following this, MPEG-4 based ECA generation process is presented. Finally, keyframe animation preparation is explained. Before conclusion, we discuss our future plans for the project.

2. MPEG-4 FACIAL ANIMATION

Moving Pictures Experts Group recently released MPEG-4 as an ISO standard [8] [9]. The standard focuses on a broad range of multimedia topics including natural and synthetic audio and video as well as graphics in 2D and 3D. Contrary to former MPEG standards that focus on efficient coding of different contents, MPEG-4 mainly concerns communication and integration of multimedia content. It has been widely accepted in the academia, while gaining attention from in-

dustry as new applications relying to it emerges. Being the only standard incorporating facial animation, MPEG-4 defines the guidelines for authoring and driving an ECA in a standardized way. In other words, with MPEG-4 Facial Animation (FA) standard [12], we have a unified way of parameterizing the animation of the face models [13]. To achieve this, one has to first set Feature Points (FP) and Face Animation Parameter Units (FAPU) that describe the geometry, in order to let Facial Animation Parameters (FAPs) drive the animation. In following subsections, we present these key concepts in detail.

2.1 Feature Points (FP)

MPEG-4 defines 84 Feature Points (FPs) that describe the shape of a standard face and should be defined for every face model. These points are used for defining animation parameters as well as calibrating the models when exchanged between different players. Figure 1 shows the set of FPs.

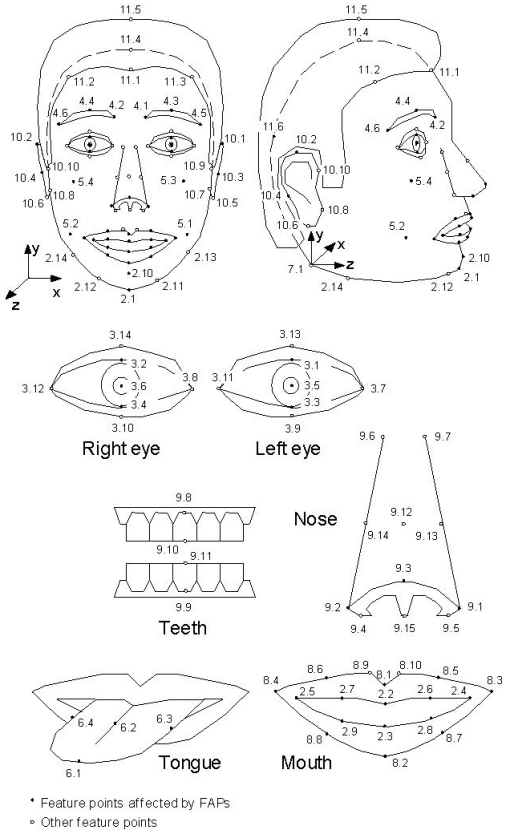

Figure 1: MPEG-4 Feature Points.

2.2 Facial Animation Parameters (FAPs)

In the standard, 68 values define the deformation between two frames of animation and they are called Facial Animation Parameters (FAPs). The first 2 of these are high level parameters representing visemes and emotions. Visemes are the visual counterpart of phonemes in speech and emotions are the set of six basic emotions defined by Ekman [4]. One can drive a face model using only the first two FAPs, interpolating between prototypical emotions and superimposing visemes, possibly by keyframe animation. However, for better quality and control, use of low level parameters are encouraged by the standard. Remaining 66 low level FAPs deal with specific regions on the face, e.g. right corner lip, bottom of chin, left corner of left eyebrow, etc. Most FAPs correspond to an FP, and define translation or rotation on that FP along an axis in three dimensions, while some of the FAPs represent rotation of the head and eyes. Using FAPs, application developers have a standard set of input for animation and yet are free to decide on how they interpret FAPs and handle deformation of the model. However, since FAPs are low level, non-trivial parameters, having a tool that generates FAPs from a human readable script incorporating speech, emotions and expressions is necessary. Examples of such scripting languages are SMIL-AGENT [11], VHML [6], APML [3] and BEAT [2] among others [14].

2.3 Face Animation Parameter Units (FAPU)

FAPs are unitless values and independent of model geometry, which means they can drive different face models, regardless of geometry. For this reason, FAPs have to be calibrated prior to use on a face model. In MPEG-4 FA, this is done using face animation parameter units (FAPU). FAPU are defined as fractions of distances between key facial features like eye-nose separation, as shown in Figure 2. They are specific to the actual 3D face model that is used. While streaming FAPs, every FAP value is calibrated by a corresponding FAPU value as defined in the standard. Together with FPs, FAPU serve to achieve independence of face model for MPEG-4 compliant face players. As a result, by coding a face model using FPs and FAPU, developers can freely exchange face models without the problem of calibration and parametrization for animation.

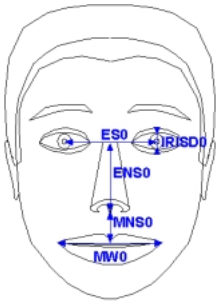

Figure 2: MPEG-4 FAPU description.

3. XFACEED

In this section, we will review the capabilities and uses of XfaceEd in more detail. XfaceEd is being developed using standard C++ and operating system independent libraries such as wxWidgets for GUI, Xerces for XML serialization. It is distributed as an open-source tool with Mozilla Public License v1.1 and anonymous access to the source code repository is available from our SubVersion server repository.

Before proceeding with the uses, we should point out that final output of XfaceEd is a configuration file in XML format. This ensures the definition of deformation rules and parameters externally, so that nothing is hard coded inside Xface library. One can crate or modify an ECA with little work, without doing any programming. The configuration file also permits the library to use multiple 3D mesh files for the head. For example, one can use a single 3D model for the

teeth and different models for the face. The configuration file is parsed and interpreted by the library and the respective deformation rules are generated automatically. Note that the configuration file is not a replacement for the original MPEG-4 binary format, but is a simpler way of description. For this reason, we use the configuration file to define our proprietary face models as suggested in MPEG-4 standard. Note that, one can also use XfaceEd output along with her own face player application without using other components from Xface toolkit.

3.1 MPEG-4 Operations

In order to create a 3D talking agent, the first step is creation of a static 3D face mesh using a 3D modeling package or 3D scanner. However, it is not so clear how to define the animation and deformation rules in order to have a talking face. As we have discussed, MPEG-4 defines a standard way to parameterize the facial animation, but one has to set FAPU and FP on the 3D model manually. XfaceEd simplifies this process by providing an easy way of defining the FAPU, FP regions, weights, and parameters for manipulating and animating the static face models.

Basically, the workflow follows these steps:

- Import one or more static 3D meshes.
- Define FAPU.
- Define FP regions, weights and deformation functions.
- Preview results and fine tune parameters defined in previous step.

3.1.1 *Importing 3D Models*

As a first step in ECA creation process, one or more static 3D models representing the face should be imported to the editor. One can use more than one mesh for the face, e.g. one for the hair, another for teeth, and compose them in order to get a more realistic character. Current 3D file formats supported are VRML1, VRML97 and Wavefront OBJ formats, which are well known in the community and they are supported by most of the 3D modeling packages. Addition of support for other file formats is trivial and can be plugged in to the editor easily.

The editor also supports basic 3D editing capabilities such as rotation, zoom, panning, and vertex selection. Initial gaze direction of the ECA is also set, once the configuration file is saved as shown below.

```
<!--Global Translation info for the face-->
<translation x="0" y="-1" z="-659"/>
<!--Global Rotation info for the face-->
<rotation axis_angle="0.407589"
          axis_x="-0.999632"
          axis_y="-0.0154467"
          axis_z="0.0223226"/>
```

3.1.2 *Defining FAPU*

After importing the models, FAPU has to be defined, as illustrated in Figure 3. The user must select seven points on the surface using the editing tools provided. Note that, this step is crucial and should be done before any of the other operations as normalization of FAP data that drives the animation are done using the distances defined by these points. FAPU data is saved in the head portion of FDP file as follows;

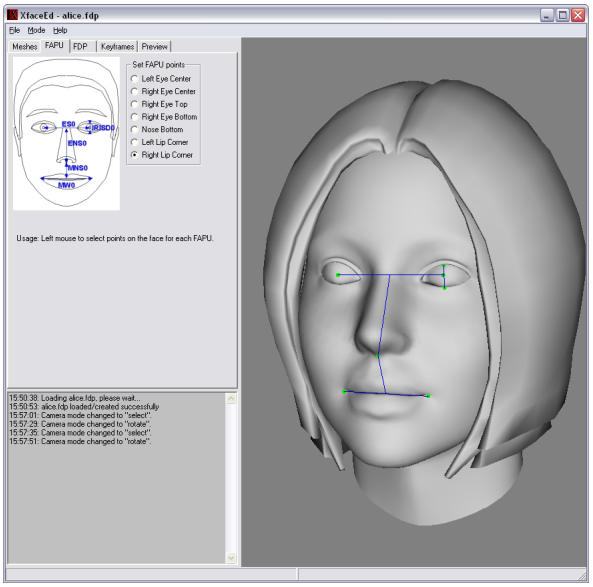

Figure 3: Setting FAPU.

```
<!--FAPU (Facial Animation Parameter Units)-->
<fapu ENS0="51" ES0="69"
    IRISD0="16" MNS0="30" MW0="50"/>
```

3.1.3 *Defining FPs*

The Xface core library is responsible for loading 3D models and corresponding MPEG-4 parameters, as well as streaming animation parameters (FAPs) and handle deformation of the face mesh for facial animation. MPEG-4 FPs are defined using XfaceEd as shown in Figure 4. As discussed previously, there are 84 FPs on the head. However, not all of the FPs are affected by FAPs. Therefore, currently we only take into account those FPs that are affected by FAPs. For each FP to be animated, corresponding control point on the model and the vertices in the zone of influence of this FP, together with one or more deformation functions, and a weight factor are set. An FP can be affected by multiple FAP values, therefore one can assign a deformation function for each FAP influence. To summarize, for every FP;

- A single control point (CP) defining the feature point (FP) itself.
- Neighbouring vertices which are in the zone of influence of the CP.
- One or more deformation functions for affecting FAPs to this FP.
- A weight value for every deformation function for fine tuning purposes.

should be defined manually.

The output in the configuration file is as follows:

```
<fdp name="2.3" index="627" affects="alice.wrl_3">
  <indices>29 87 591 592 634 795 1491 1646 1647
           1648 1649 ... </indices>
  <influence weight="0.33" fap="4"
      type="RaisedCosInfluenceWaveY" />
```

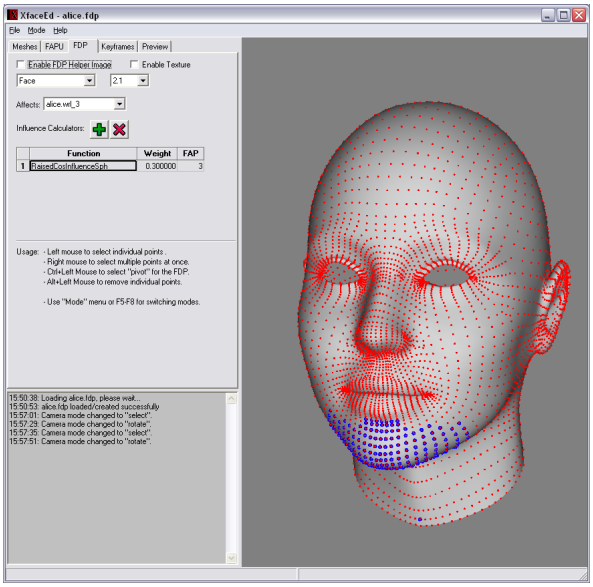

Figure 4: Setting parameters for chin area.

```
  <influence weight="-0.1" fap="15"
      type="RaisedCosInfluenceSph" />
</fdp>
```

Presently, the deformation function is defaulted to variations of raised cosine function as shown in Equation 1, where Δv_p denotes the euclidean distance that the point p in FP zone of influence should be translated in FAP direction, while d_p is the p's and d_{max} is the farthest point's distance in region to the FP. w is a weight value defined for every deformation and fap is the FAP value.

$$\Delta v_p = \left(1 + \cos\left(\pi \times \frac{d_p}{d_{max}}\right)\right) \times w \times fap \qquad (1)$$

Raised cosine achieves satisfactory results, yet the library can be extended to use different deformation strategies [10] such as RBF (Radial Basis Functions) [7] and FFD (Free Form Deformations) [5]. Extension is done via pluggable factories design pattern, which enable the XfaceEd tool to automatically recognize and add new deformation functions to its GUI without recompilation.

3.1.4 *Preview and Fine Tuning*

In the final step, one can preview the correctness and quality of animation by loading external FAP files in XfaceEd together with a speech file as shown in Figure 5. As one can expect, every FAP file should be synchronized by its accompanying speech file. This is ensured during FAP generation process, which is done externally. Note that, Xface toolkit does not provide any means (yet) to provide FAP generation and depends on FAP files created externally. Every FAP file has a predefined frames per second (FPS) value. For a given FPS value, how many steps of deformation should be processed in every second is known *a priori*. During playback, the speech file is started and the animation waits or chases the voice depending on the performance of the rendering in realtime. With a powerful hardware, the animation is smooth and fast, therefore desired number of frames can be rendered. If the animation is too fast, rendering waits for the speech to catch up. In the other extreme where the setup is less powerful and target FPS cannot be achieved, rendering skips some of the frames and this time, chases the speech.

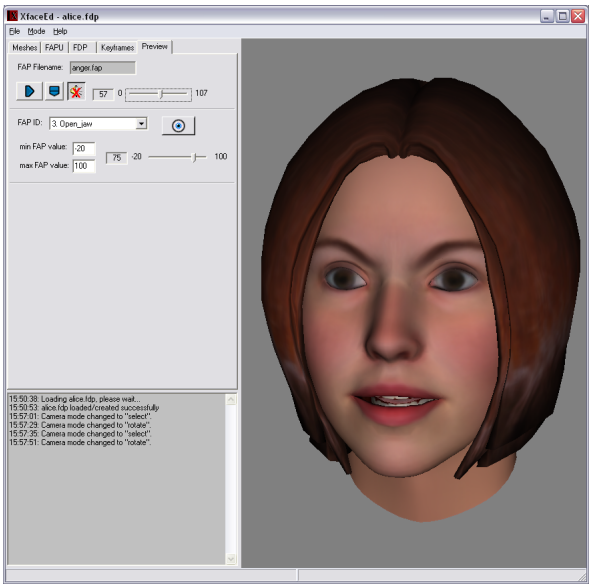

Figure 5: FAP streaming preview.

Alternatively, one can choose to preview the effect of a single FAP on an FP, and the limits of the deformation can be tested by varying its value (see Figure 6). In the first mode mentioned above, overall quality is assessed while in this mode, user can understand the contribution of each FP region and the coresponding parameters better for fine tuning.

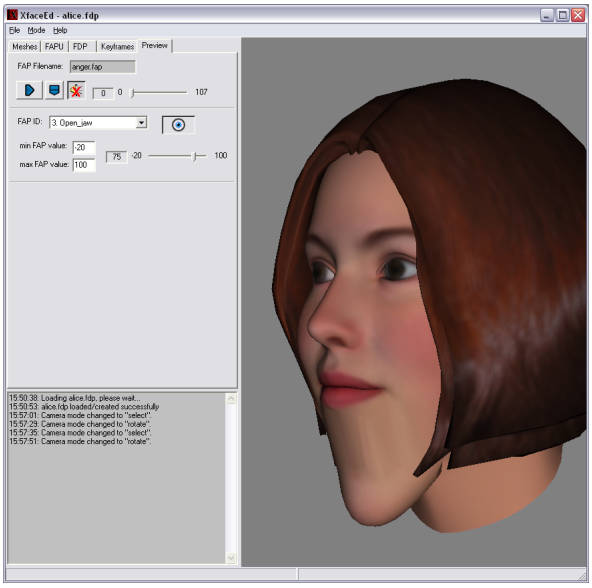

Figure 6: Single FAP value preview.

Note that, this final step is an iterative process done by tweaking and fine tuning all the parameters set previously.

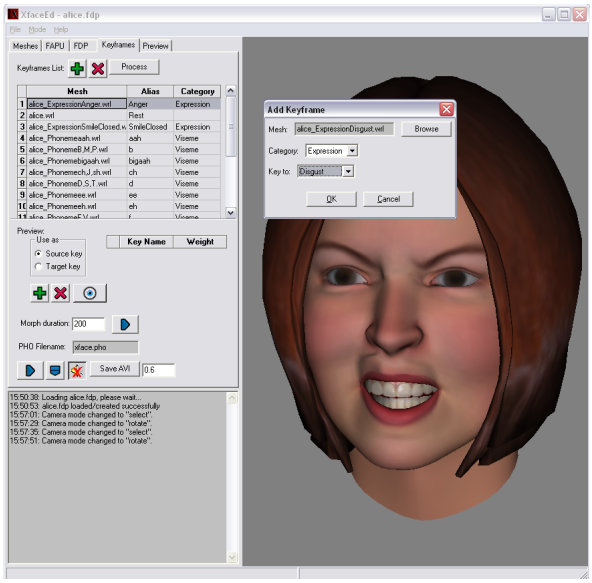

Figure 7: Addition of a morph target.

The changes made to the parameters take into effect immediately and speeds up the workflow of the user.

3.2 Keyframe Animation

In addition to MPEG-4 based facial animation, Xface also supports keyframe animation technique. In this mode, animation system relies on effort from 3D artists, since for every viseme and expression, a separate 3D model as a morph target has to be supplied. However, once these morph targets are ready, creating an ECA is much simpler since there are less parameters to control compared to MPEG-4. In our project, for morph target creation, we use the FaceGen [16] tool, which automatically generates a set of morph targets for a given static face model.

As shown in Figure 7, first step is to create a dictionary of morph targets and categorize them into three classes as *Viseme* for visual speech generation, *Emotion* for emotions and *Modifier* for other expressions such as eye blinking, frowning, etc. These morph targets in the dictionary are saved in the configuration file as follows:

```
<entity alias="q" category="Viseme">
  <mesh file="alice_PhonemeQ.wrl" format="wrl"/>
</entity>
```

During animation, morph targets from different categories are blended as desired for every frame of animation as explained in [15]. Once the keyframes for each time frame is created on the fly by blending process, they are interpolated, synchronized with the timing information coming from the speech file and phoneme durations. For the interpolation, the core library relies on linear and cubic interpolation functions shown in Equations 2 and 3 in the current version.

$$k(t_u) = k(t_0) \times (1-u) + k(t_1) \times u \quad (2)$$

$$k(t_u) = k(t_0) \times (2u^3 - 3u^2 + 1) + k(t_1) \times (3u^2 - 2u^3) \quad (3)$$

where u is a value between 0 and 1, $k(t_0)$ and $k(t_1)$ are the vertices of previous and next keyframes respectively, and $k(t_u)$ designates the morph target interpolated using these two.

Note that, in order to preview the animation in realtime, phoneme durations designating how long each phoneme lasts during speech should be provided by the user. These durations can be extracted easily by either a speech synthesizer or a speech recognizer module, depending on the user's own platform. Xface toolkit does not incorporate a recognizer or a synthesizer but supports externally calling them from within its programming environment. Once the speech file and the phoneme durations are supplied, various emotions and modifiers from the keyframe dictionary can be tested for blending to arrive at a realistic, lifelike behaviour. In Figure 8, visual speech blended with anger keyframe is shown.

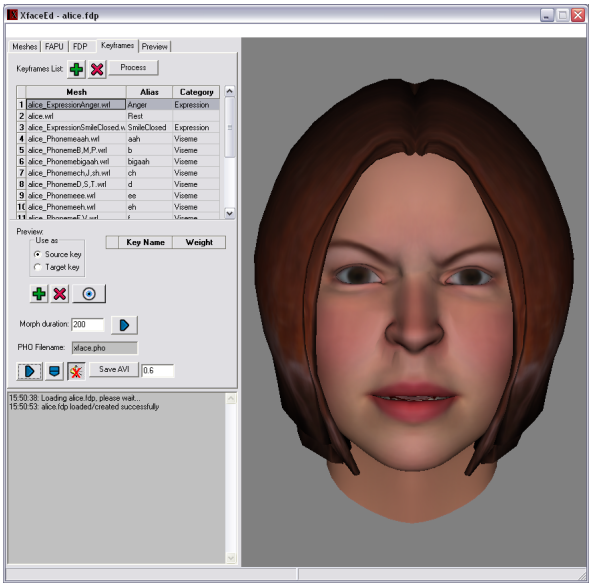

Figure 8: Blending: Angry speech.

4. CONCLUSION AND FUTURE WORK

In conclusion, XfaceEd, our open source, 3D embodied conversational agent authoring tool for multimodal environments is presented in this paper. Development of new features and improvements are still in progress, while a beta version of the tool is available for download and testing.

In the near future, we plan to focus on two issues. First, automatic generation of FP regions in order to achieve rapid development of talking agents. Note that, at the current version, users should specify the vertices in the regions manually. With a clever region selection tool using the connectivity analysis among neighbouring vertices, users will be able to select regions of predefined shapes and modify them according to their needs by adding and removing individual vertices.

Next, we will improve the blending approach and interpolation technique for the keyframe animation in order to account for morph targets from various categories specified by the user and more testing options for keyframe interpolation previewing as well as addition of new interpolation functions for enhanced quality.

5. REFERENCES

[1] K. Balcı. Xface: MPEG-4 based open source toolkit for 3d facial animation. In *Proc. Advance Visual Interfaces*, Italy, May 2004.

[2] J. Cassell, H. Vilhjlmsson, and T. Bickmore. BEAT the behavior expression animation toolkit. In *Proc. of SIGGRAPH 01*, 2001.

[3] N. DeCarolis, V. Carofiglio, and C. Pelachaud. From discourse plans to believable behavior generation. In *Proc. Int. Natural Language Generation Conf.*, New York, July 2002.

[4] P. Ekman. *Handbook of Cognition and Emotion*. Wiley, New York, 1999.

[5] P. Faloutsos, M. van de Panne, and D. Terzopoulos. Dynamic free-form deformations for animation synthesis. *IEEE Transactions on Visualization and Computer Graphics*, 3(3):201–214, September 1997.

[6] InterFace. *The Virtual Human Markup Language*. Curtin University of Technology, 2001. Working Draft [Online] Available: http://www.vhml.org.

[7] N. Kojekine, V. Savchenko, M. Senin, and I. Hagiwara. Real-time 3d deformations by means of compactly supported radial basis functions. In *Short papers proceedings of Eurographics*, pages 35–43, Saarbrucken, Germany, 2-6 September 2002.

[8] I. J. N1901. Text for CD 14496-1 Systems. Fribourg Meeting, November 1997.

[9] I. J. N1902. Text for CD 14496-1 Visual. Fribourg Meeting, November 1997.

[10] J. Noh and U. Neumann. A survey of facial modeling and animation techniques. Technical Report 99-705, USC, 1998.

[11] E. Not, K. Balci, F. Pianesi, and M. Zancanaro. Synthetic characters as multichannel interfaces. In *Proc. of ICMI05*, October 2005.

[12] I. Pandzic and R. Forchheimer. *MPEG-4 Facial Animation: The Standard, Implementation and Applications*. Wiley, New York, 2002.

[13] F. Parke. Parameterized models for facial animation. *IEEE Computer Graphics App. Mag.*, 2(9):61–68, 1982.

[14] H. Prendinger and M. Ishizuka, editors. *Life-Like Characters: Tools, Affective Functions, and Applications*. Springer, 2004.

[15] H. Pyun, W. Chae, Y. Kim, H. Kang, and S. Y. Shin. An example-based approach to text-driven speech animation with emotional expressions. Technical Report 200, KAIST, July 2004.

[16] FaceGen Modeller from Singular Inversions Inc. http://www.facegen.com.