A Facial Animation Interactive Framework with Facial Expressions, Lip Synchronization and Eye Behavior

Rossana B. Queiroz Marcelo Cohen Soraia R. Musse

Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Av. Ipiranga, 6681, Porto Alegre, RS, Brazil, 90619-900

Figure 1: *Example of characters performance in our prototype during an expressive talk.*

Abstract

In this paper we describe our approach of generating convincing and empathetic facial animation. Our goal is to develop a robust facial animation platform that can be scalable, usable and easily extended, in order to allow integration of research on the area and also the direct incorporation of the characters in interactive applications, such as Embodied Conversational Agents and games. We have developed a framework capable of animating MPEG-4 parametrized faces easily through high-level description of facial actions and behaviors. We also present a case study which integrates computer vision techniques in order to provide interaction between the user and a character, that interacts with different facial actions according to detected events in the application.

Keywords:: Facial Animation, MPEG-4, Facial Expressions, Lip Synchronization, Eye Behavior, Interactivity, Facial Animation Description Language

Author's Contact:

{rossana.queiroz, marcelo.cohen, soraia.musse}@pucrs.br

1 Introduction

Animated virtual human faces have been widely used in many applications, such as movies, games and Embodied Conversational Agents (ECAs). In all of these applications, characters' reactions should help in user immersion as well as to provide a believable interaction. Thus the facial behavior coherence is very important, to increase the credibility of the character.

Recent research on facial animation has lead to models that explore expressiveness, communication and interactivity. Many of these research are focused on ECA development [de Rosis et al. 2003; Smid et al. 2004; Cosi et al. 2007; Huang et al. 2008] and provide us a range of studies that correlate human psycho-social behavior and facial animation. Together with a facial animation platform development, we find several studies about face parameterization and scripting languages for assisted and automatic animation generation [Perlin 1997; Cassell et al. 2001; Rutledge 2001; Arafa and Mamdani 2002; Byun and Badler 2002; Carolis et al. 2004; Not et al. 2005; Arya and DiPaola 2007; Vilhjálmsson et al. 2007]. Those studies have provided ways to the higher-level description of a character's face actions. Most of these languages are based on XML and try to describe face actions in different levels, such as attributing values of some specific parameters, or specifying more complex behaviors which implicate the animation of various facial attributes in a synchronized way.

We observe that in some works focused on complex facial models, such as the physically-based animation by [Sifakis et al. 2005], the implementation of the facial animation platform is constrained to a single character model where all tests are performed on. In fact, if a facial animation prototype is focused only on its main research subject, it becomes difficult the addition of new functionalities or its use in other environments or applications. Concerning the project of a facial animation platform, we can emphasize some requirements in order to obtain a robust and usable tool:

- A consistent set of face animation parameters, which allows us to get a satisfactory control of face attributes in different faces;
- Minimal need of animator manual work, such as preparing a set of animation keyframes to be interpolated;
- Mesh-deformation algorithms which produce realistic representation of facial muscle actions;
- An interface which can be understood both by computer and humans, enabling them to determine or edit characters' actions;
- The real possibility of easy incorporation of the animated faces in other applications;
- A framework which enables interactivity with real time animation generation, without the need of previously recorded animations.

In this context, our work presents an interactive facial animation framework which considers facial expressions, synchronized speech and eye behavior generation where the users define the characters' actions by high-level description and can use the produced animations on different face models. Our goal is to develop a robust facial animation platform that can be easily extended, scalable and usable in order to allow integration of other research on the area and also the direct incorporation of the characters in interactive applications such as ECAs and games. Our main contribution is the methodology of building an animation platform using free or open source tools, integrating some known animation models and capable of generating a substantial amount of different face actions in good quality animations.

This paper is organized as follows: the next section presents some related work. Section 3 describes the architecture of our model and the main technologies and tools that we have used for development of the framework. In Section 3.4 we describe a case study: an interactive application where a character reacts according to detection of the user's face. Finally we make some final remarks and suggest future work in order to improve and evaluate our model.

2 Related Work

Parameterization techniques for facial animation have been an area of active research since Parke [Parke 1982]. We can identify in the

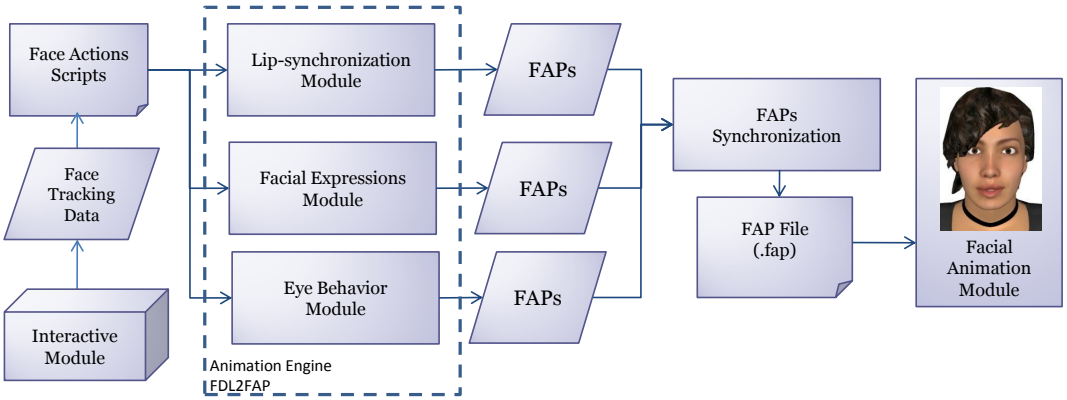

Figure 2: *Overall architecture diagram of the framework*

literature two standards of parameter sets that are widely used in facial animation works: Facial Coding System (FACS) [Ekman and Friesen 1978] and the MPEG-4 Facial Animation (MPEG-4 FA) [Pandzic and Forchheimer 2003]. However, these standards provide a rather low-level basis for animators, as they describe the parameters as a set of pseudo-muscles which can be activated in a given moment, producing a facial expression (Subsection 3.3 overviews the MPEG-4 FA). Hence there are several efforts to produce a more high-level parametric approach to generate facial animations in a effective way [Perlin 1997; Cassell et al. 2001; Rutledge 2001; Byun and Badler 2002; Carolis et al. 2004; Not et al. 2005; Arya and DiPaola 2007].

In the bibliography, we can also find some works that describe their facial animation frameworks, which aim to provide a desirable platform for facial animation research or the development of talking heads applications. Wang [Wang et al. 2007] describes a methodology to build an expressive facial animation system with lip synchronization using affordable off-the-shelf components. This include the *FaceGen Modeller*[1] software for face and key meshes generation, and the Microsoft Speech SDK [2] as the speech API.

Cosi [Cosi et al. 2005] proposes a facial animation toolkit implemented in MATLAB[3] created mainly to speed up the procedure for building the LUCIA talking head [Cosi et al. 2007] through motion capture techniques, translated to MPEG-4 parameters. Although it seems to be a promising approach, it was built over proprietary software.

DiPaola and Arya [DiPaola and Arya 2007] propose a facial animation framework compatible with the MPEG-4 standard called iFace, whose binaries are available on the web[4]. iFace allows interactive non-verbal scenarios through a XML-based scripting language called FML (Face Modeling Language) [Arya and DiPaola 2007], which allows both parallel and sequential description of face actions. Face actions include talking, expressions, head movements and low-level MPEG-4 parameters. Actually, iFace has been used for behavioral animation studies in order to reach a comprehensive association of facial actions to personality types, creating a higher-level facial parameter set in a personality-space [Arya et al. 2006].

Balci *et al.* [Balci et al. 2007] designs Xface, a set of open source tools for creation of talking heads using MPEG-4 and keyframe-based animation driven by the SMIL-Agent scripting language. The toolkit is freely available [5] and aims to supply the lack of free and open source tools for research as mentioned above. However, Xface uses the SMIL-Agent scripting language for its keyframe-based animation module. In other words, for each face model it is necessary a set of corresponding key-meshes with the different facial expressions and visemes.

Our approach uses Xface as the facial animation engine and integrates a model to generate MPEG-4 animation automatically through facial actions scripts with lip synchronization [Rodrigues 2007] and eye behaviors [Queiroz et al. 2007], without the need of keyframing meshes. Moreover, in our framework external controls are possible in order to interactively define characters' actions. As a case study of the framework capabilities, we have developed a simple application where the character eye movements are driven by detection of the user's face, in an approach slightly similar to [Courty et al. 2003].

[1] http://www.facegen.com/
[2] http://www.microsoft.com/speech/download/sdk51
[3] http://www.mathworks.com/
[4] http://ivizlab.sfu.ca/research/iface/
[5] http://xface.itc.it/

3 Model

This section presents the overall architecture of our framework, as well the scripting language for the description of facial actions. We also present the approach to animation synthesis and interactivity.

3.1 Architecture

The overall architecture of our framework is presented in Figure 2. The input for the *Animation Engine* is a script file containing the description of one or more facial actions. Our scripting language is called FDL (Face Description Language) hence we name our script files as FDL files. The *Animation Engine* interprets the facial actions within a FDL file and generates the animation according to them. In the current stage of our research, the FDL files can describe a sequence of three types of high-level face actions: talking, facial expressions and eye behaviors. Each of these types of face actions are independent, and the processing of them in our *Animation Engine* is performed by three different modules: *Lip-Synchronization Module*, *Facial Expression Module* and *Eye Behavior Module*. This provides the values of each MPEG-4 facial animation parameter (FAP) corresponding to the desired facial actions through time. After this step, the *FAPs Synchronization Module* receive the FAP values and resolves possible conflicts among them, such as making the combination of facial expressions and visemes. The output is a FAP file containing the final animation, which can run on any MPEG-4 compliant player and with different face models. Our *Facial Animation Module* uses Xface for the facial animation synthesis (described in Subsection 3.3).

The *Eye Behavior Module*, whose architecture is illustrated in Figure 3 contains the implementation of [Queiroz et al. 2007] a model for automatic generation of eye animation. This provides a set of eye behaviors which can be used in combination with different affective states. The model uses a known statistical model [Lee et al. 2002] (*Default Model*) as a saccadic eye movement engine and creates differentiated behaviors (*Behavioral Database*) through changes in the gaze parameters, such as direction, magnitude and interval between movements. Saccadic movements (or saccades) are rapid movements of both eyes from one gaze position to another [Lee et al. 2002].

Eye behaviors are described as high-level actions in FDL scripts. When combined with facial expressions, they contribute for increasing both the expressiveness and engagement in communica-

tion. The *Expressive Gaze Generator* receives descriptions of face actions and returns the FAP values for each gaze generated according to the specified eye behavior, also providing head and eyelids movements according to the model rules. The *Interactive Module*, described in Subsection 3.4, can provide the input eye behavior sequence (e.g. following the user face) in a interactive way.

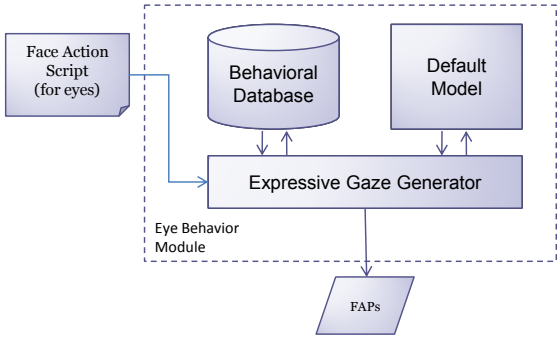

Figure 3: *Diagram of the Eye Behavior Module*

The job of the *Facial Expression Module* (Figure 4) is simpler: currently, it only produces facial expressions predefined as a set of FAP values in the *Emotional Database*, representing "pure" emotions. But the framework can be easily extended, in order to allow new facial expressions and algorithms to generate derived expressions. The facial expressions are also described as facial actions in the FDL language.

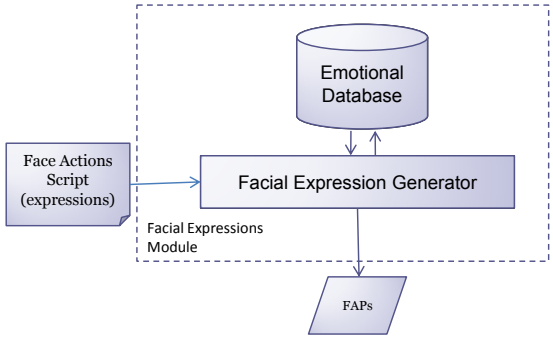

Figure 4: *Diagram of the Facial Expressions Module*

The *Lip Synchronization Module* follows the methodology described in [Rodrigues 2007]. Basically the module receives a sound file (.wav) as input, containing the character speech and an auxiliary file containing its textual description. For the generation of lip animation, we use the CSLU Toolkit [6] which aligns the sound file with the textual description and provides a file with the timeline of the phonemes through time. This file is the input for the *Phoneme-Viseme Mapping Module*, which connects the phonemes to their visual representation (visemes), according to [Rodrigues 2007]. The CSLU toolkit alignment script generates 41 English phonemes that are mapped to the 14 visemes specified in the MPEG-4 standard. There is also an auxiliary "viseme" we have called *silence* to represent the pauses in speech. After this step, we have the complete timeline of visemes, whose mapping to FAPs are carried out in the *Viseme-FAP Mapping Module*. In this stage the blending of visemes (i.e. the transition between two visemes) is also performed. For this transition, we apply a linear interpolation between them. Figure 5 illustrates the architecture of this module.

After the three modules of the *Animation Engine* have finished, the *FAPs Synchronization Module* receives the output FAP values and resolves possible conflicts among them. Meanwhile, our framework solves three types of conflicts:

[6]http://cslu.cse.ogi.edu/toolkit/

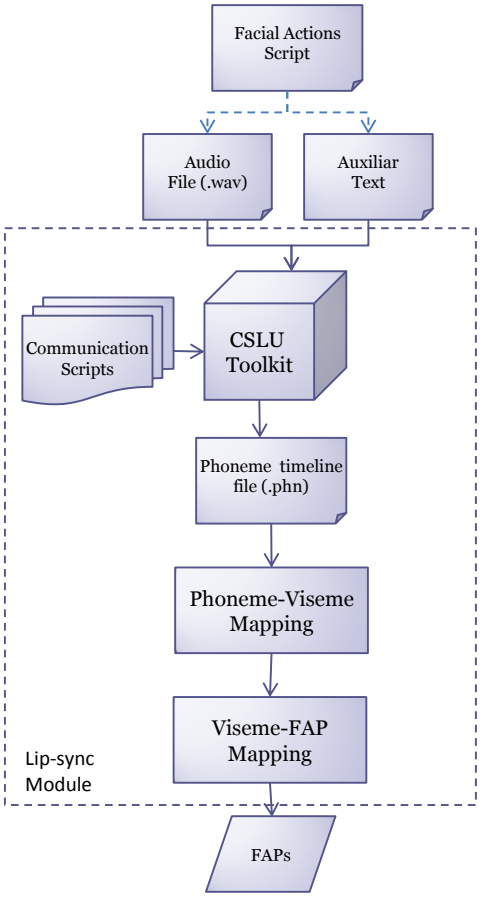

Figure 5: *Diagram of the Lip Synchronization Module*

1. Viseme-Expression conflict: occurs mainly in mouth region FAPs, when the facial expression includes mouth movement. To solve this conflict, we blend the mouth FAP streams generated by the *Facial Expression* and *Lip Synchronization* modules, in a weighted sum of them. Pyun [Pyun et al. 2003] proposes that the weights for different visemes and expressions should variate according to the "importance" of them. However, in the current stage our framework uses the same weights for all combinations of expressions and visemes, as shown in equation 1. Therefore, each final FAP (F_i) will be generated through the weighted sum of both expression (E_i) and viseme (V_i) FAPs.

$$F_i = \frac{E_i}{4} + \frac{3V_i}{4} \quad (1)$$

2. Eyes-Expression conflict: the *Eye Behavior Module* generates eyelid movements related with the gaze direction and shift [Queiroz et al. 2007]. But some facial expressions also include the eyelids conformation. In such cases, we also make a weighted sum, as in the Viseme-Expression conflict (25% for eye behavior and 75% for expression FAPs)

3. Head control: during FDL processing, some eye behaviors lead to head movements. If that happens, the *Eye Behavior Module* signals the *FAPs Synchronization Module*, which in turn preserves the generated head FAPs. If there are no head movements, then our framework provides an implementation of Perlin Noise [Perlin 1985] to generate subtle head movements instead of none, in order to approach a more realistic head behavior.

We also include in our framework a more generic module we named *Interactive Module* which is a *hotspot* to enable the inclusion of interactive events detection and automatic generation of animation with their respective characters' responses in runtime. Through this module, we can incorporate Virtual Reality approaches (e.g.

through Computer Vision techniques), emotional models which lead to facial expressions, Natural Language Processing techniques to verbal or textual communication (e.g. a chat), and such. We can also define rules of the interactivity of two or more virtual agents in a simulation or game application.

3.2 Facial Description Language

In our previous work [Queiroz et al. 2007], we have encoded our face actions as a high-level parameterized set of commands in Lua[7] scripts we called Facial Description Language (FDL). We opted to use Lua as auxiliary scripting language instead of XML-like notations (as SMIL-Agent and FML) for two reasons: *i)* Lua syntax is clearer and its structure helps intuitively to understand the sequencing of actions; and *ii)* it is a powerful way to easily incorporate new functionalities in the framework.

Our approach describes each expression, viseme or eye behavior as a facial action which is triggered by script files supplied by an interactive application, or simply selected by an user. These script files describe one or more face actions, which are interpreted, processed and translated to low-level MPEG-4 Facial Animation Parameters (FAPs). The main script file contains the information about the speech (sound and text file names) and the scripts with the eye behavior and expressions "storyboard", as shown below. If there is no speech in the desired animations, the *speechSound* and *speechText* fields should be filled with the `"none"` string value.

```
FDL = {
   speakSound = "hastalavista.wav",
   speakText = "hastalavista.txt",
   expressionScript = "expressions.sb",
   head = "default",
   eyesScript = "eyes.sb",
   output = "animation",
}
```

A sequence of facial expressions can be described as the script below, which informs the desired expressions through time, with their respective duration times in frames. The example shows some of the facial expressions which are part of our *Emotional Database*.

```
expressions = {
   {"joy", 100},
   {"sadness",100},
   {"surprise", 100},
   {"anger", 50},
   {"disgust",50},
   {"fear",100},
   {"trust",100},
   {"tongueout",100},
   {"inlove",100},
   {"worry",100},
   {"sleepy",100},
}
```

Similarly, the eye behavior sequence can be described as the example below. The set of eye behaviors description and its parameters are the same of GDL (*Gaze Description Language*) files of [Queiroz et al. 2007] adding the "pursuit" behavior we describe in Section 4 in order to generate the eyes pursuit movement.

```
eyes = {
   {"lookTo","left",17.0,"yes",50},
   {"lookTo","upleft",10.0,"yes",150},
   {"lookTo","up",15.0,"no",150},
   {"lookTo","upright",7.0,"yes",50},
   {"lookTo","right",15.0,"yes",50},
   {"lookTo","downright",12.0,"no",50},
   {"lookTo","down",12.0,"yes",50},
   {"lookTo","downleft",12.0,"yes",50},
   {"default", "talking", 0.9, 500},
   {"concentration", 0.01,50},
   {"discomfort", 0.99, 0.6,500},
   {"distress", 0.5,400},
   {"ironic", 0.5,50},
}
```

As we show above in the first FDL script, we generate mouth speech animation automatically through a sound file and its textual description. After the phoneme-viseme mapping, our system generates a FDL script such as example below, which is processed by the *Viseme-FAP Module*. This type of script can be alternatively used for, e.g. the output of other speech-processing system, or even edited manually. Following the same syntax of the other FDL scripts, it describes the sequence of visemes through time.

```
Sentence = {
   {"silence", 11},
   {"A", 6},
   {"kg", 1},
   {"Q", 1},
   {"nl", 3},
   {"sz", 3},
   {"I", 6},
   {"A", 2},
   {"silence", 27},
}
```

3.3 Animation Platform

Our framework follows the MPEG-4 Facial Animation standard for face and animation parameterization. The MPEG-4 FA describes the steps for creation of an animated face by definition of a set of parameters in a standardized way. First, MPEG-4 defines 84 feature points (the FPs - see Figure 6) placed on a character head, which in turn define animation parameters, as well as calibrating the models when they are exchanged between different players.

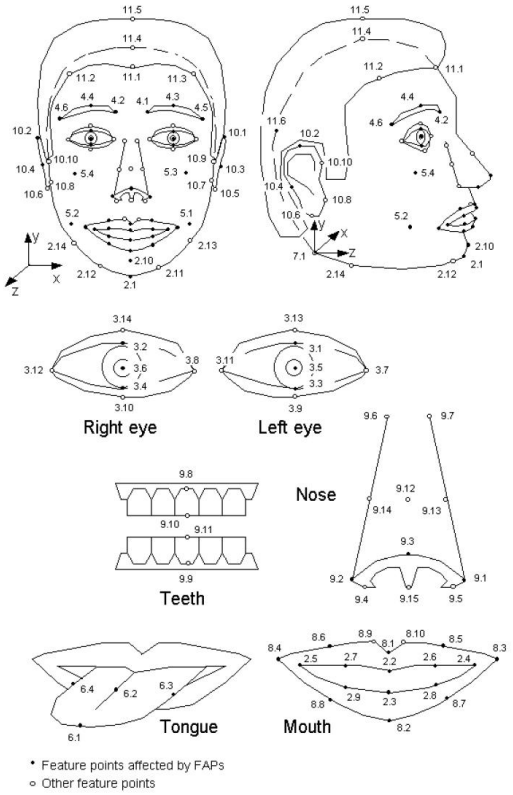

Figure 6: *MPEG-4 Feature Points*

In the standard, 68 values (the *Facial Animation Parameters*, or FAPs) define the deformation between two frames of animation. The first two suit a framework with high level parameters, representing visemes and the six basic emotions defined by Ekman [Ekman 1999]. The next ones deal with specific regions on the face, as left eyebrow, right corner lip, tongue tip, etc. FAP values are independent of model geometry. For this reason, FAPs have to be calibrated prior to use on a face model. This is done using *Face Animation Parameter Units* (FAPU, illustrated in Figure 7) which are defined as fractions of distances between key facial features. Moreover, the information about the 3D model is provided through the *Facial Definition Parameters* (FDPs) which allow one to configure the 3D facial model to be used at the receiver, either by adapting a previously available model or by sending a new model. The new or the adapted model is then animated by means of FAPs [Abrantes and Pereira 1999; Balci 2004].

Each model geometry has its own FAPU and every FAP value is

[7] http://www.lua.org

calibrated by a corresponding FAPU value as defined in the standard. Together with FPs, they serve to achieve independence of the face model for MPEG-4 players [Balci 2004]. Figure 8 illustrates the FAP file format, which can be read by any MPEG-4 compliant player.

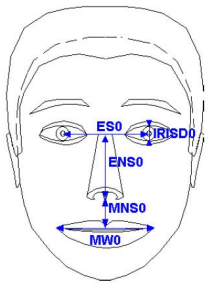

Figure 7: *MPEG-4 FAPU*

We use Xface as the MPEG-4 FA engine which plays the FAP animation generated by the *Animation Engine Module*. The Xface toolkit is implemented in C++ using the OpenGL API and incorporates four pieces of software [Balci et al. 2007]: *i)* the Xface core library, which enables the developers to embed 3D facial animation into their applications; *ii)* XfaceEd, an authoring tool to generate MPEG-4 parameterized meshes; *iii)*; Xface Player, a sample application that demonstrates the toolkit in action; and *iv)* XfaceClient, which allows remote network control of XfacePlayer. Our Facial Animation framework module is implemented over the XfacePlayer, adding a FDL loader function and integrated to the *Interactive Module*, as we describe in Subsection 3.4. Our 3D face models were generated in *FaceGen Modeller*, and body and hair was modeled by artists. Our faces were parameterized according to the MPEG-4 standard using the XfaceEd tool. XfaceEd generates a configuration file containing FAPU and FDP data, which are read by XfacePlayer to open a face model.

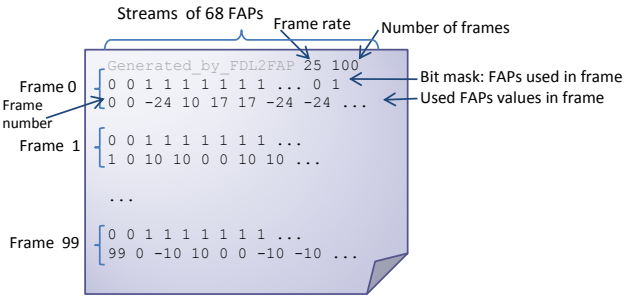

Figure 8: *FAP file scheme*

Figure 1 illustrates the visual results during the performance of some scripted expressive talks. The 3D model independence of our generated animations is illustrated in Figure 9, which shows our two different characters performing the same facial expressions (i.e. playing the same generated FAP file for both characters).

3.4 Interactivity

In our prototype, we can currently generate new animations by two ways:

- loading FDL files, which calls the FDL processing routines and then runs the generated FAP file; or
- enabling our *Interactive Module* to detect events and generate runtime specific FDL/FAP files with the respective character responses for these events.

The next section presents a case study in which our framework was extended to an interactive application using a Computer Vision technique to drive the animation of a character's eye.

Figure 9: *Performance of facial expressions by two different characters in our system.*

4 A Case Study: Following Faces

As we described in the previous sections, a robust facial animation platform should be extensible enough to allow the developers to build its applications in an effective way. When referring to facial animation, we see that many applications need automatic runtime generation of animation generation, in order to produce coherent characters' reactions throughout the interaction with other agents or users.

In order to provide support for these features, we have extended our framework with the *Interactive Module* and created an interactive application that generates eye motion driven by Computer Vision techniques. Specifically, our interactive application presents a virtual character, which follows the user's face through gaze behavior. The module that detects and tracks the user's face employs the face

detection method provided by the OpenCV[8] Computer Vision library as a starting point [Viola and Jones 2001].

The face tracker runs on a separate thread, hence face detection data is readily available. The OpenCV detection function returns the location and the face radius (in pixels) of all recognized faces, but at this moment we consider just the face which is closest to the virtual character, i.e. the face with the largest radius. Based on this information, the direction and magnitude (angle of eyeball rotation) of the virtual character's eyes are calculated as follows:

1. Obtain x and y coordinates from the face tracker for the closest face. These coordinates are converted to the range $[-1, 1]$ (zero being the center of the camera image).

2. Determine gaze direction independently for x and y:

   - If $x < -0.1$ we assume that the face is towards the right side of the camera image. This is 10% of the right half size, so very small face movements will not cause any change in the eye direction. Note that the meaning is reversed, as the camera image is normally mirrored.

   - Likewise, if $x > 0.1$ then the face is towards the left side of the camera image. The process is carried out similarly for y.

3. Now to determine the direction, we just combine the vertical and horizontal information. This produces one out of eight discrete directions (up, up and left, left, etc).

4. Finally, the magnitude is computed through the Euclidean distance of the face to the center of the camera image, scaled by a factor of 20. This produces roughly 12-15 degrees of horizontal and vertical rotation, as the face tracker does not return coordinates for faces that are very close to the edges of the screen, as there is not enough data.

After that, these parameters are used to generate an eye behavior FDL script file, which is processed by the *Animation Engine Module*. Note that there is a single computation for both eyes: we assume, for the purposes of this test application, that the user will never get too close to the camera, thus the virtual character eyes will never need to converge inwards.

In order to make the virtual character's gaze follow the user's face, a new gaze behavior was implemented. This behavior (called *pursuit*), allows eyeball rotation without returning to a central position, allowing continuity between eye's face actions. Pursuit movements occur when the eyes follow a moving object, either voluntarily or involuntarily. They are quite different from saccadic movement: they are smooth, slower and have a smaller latency [Lee et al. 2002]. After each face detection, a FDL script is generated, as shown in the example below:

```
eyes = {
   {"pursuit","right",17.0,"yes",50},
   }
```

The parameters recognized by the *pursuit* behavior are the following:

- gaze direction (2D rotation axis), described in a textual representation as eight discretized directions (such as 0°, 45°, 90°as "left", "upleft","right"). The use of this discretized values is inherited of the Default Model implementation of the model developed by [Lee et al. 2002];

- magnitude angle of eyeball rotation;

- a yes/no value indicating if the head should follow eye movement or not;

- duration (in frames).

Figure 10 illustrates our "Following Faces" application.

[8] http://sourceforge.net/projects/opencvlibrary/

5 Final Remarks

This paper presents an interactive facial animation framework which considers facial expressions, synchronized speech and eye behavior generation where the users define the characters' actions by high-level description. For animation generation, we follow the MPEG-4 Facial Animation standard. Consequently, the generated animations can be used in different face models. We also present a case study that incorporates Computer Vision techniques in a simple Virtual Reality application, in which character's interacts with user following his/her face with gaze behavior. This application shows that our *Interactive Module* is promising for the incorporation of different events and runtime interactive issues.

As future work, we aim to:

- improve the *Facial Expression Module*;

- provide head control, also allowing independent head motion as facial actions (head behaviors);

- support visemes from other languages;

- incorporate a GUI tool for editing facial actions in an easier way;

- evaluate some framework features with subjects.

In summary, our visual results show that in the current stage of development, our approach is promising, since our framework allows the integration of other research on the area and also provides interactive control of characters for applications such as ECAs and games.

Acknowledgements

This work was developed in collaboration with HP Brazil R&D. Thanks to Prof. Avelino Zorzo for providing his pictures.

References

ABRANTES, G., AND PEREIRA, F. 1999. Mpeg-4 facial animation technology: survey, implementation, and results. *Circuits and Systems for Video Technology, IEEE Transactions on 9*, 2 (Mar), 290–305.

ARAFA, Y., AND MAMDANI, A. 2002. Multi-modal embodied agents scripting. *Multimodal Interfaces, IEEE International Conference on 0*, 454.

ARYA, A., AND DIPAOLA, S. 2007. Face modeling and animation language for mpeg-4 xmt framework. *Multimedia, IEEE Transactions on 9*, 6 (Oct.), 1137–1146.

ARYA, A., JEFFERIES, L. N., ENNS, J. T., AND DIPAOLA, S. 2006. Facial actions as visual cues for personality: Research articles. *Comput. Animat. Virtual Worlds 17*, 3-4, 371–382.

BALCI, K., NOT, E., ZANCANARO, M., AND PIANESI, F. 2007. Xface open source project and smil-agent scripting language for creating and animating embodied conversational agents. In *MULTIMEDIA '07: Proceedings of the 15th international conference on Multimedia*, ACM, New York, NY, USA, 1013–1016.

BALCI, K. 2004. Xface: Mpeg-4 based open source toolkit for 3d facial animation. In *AVI '04: Proceedings of the working conference on Advanced visual interfaces*, ACM Press, New York, NY, USA, 399–402.

BYUN, M., AND BADLER, N. I. 2002. Facemote: qualitative parametric modifiers for facial animations. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ACM, New York, NY, USA, 65–71.

CAROLIS, B. D., PELACHAUD, C., POGGI, I., AND STEEDMAN, M. 2004. Apml, a mark-up language for believable behavior generation. *H. Prendinger, ed., Life-like Characters.*, 65–85.

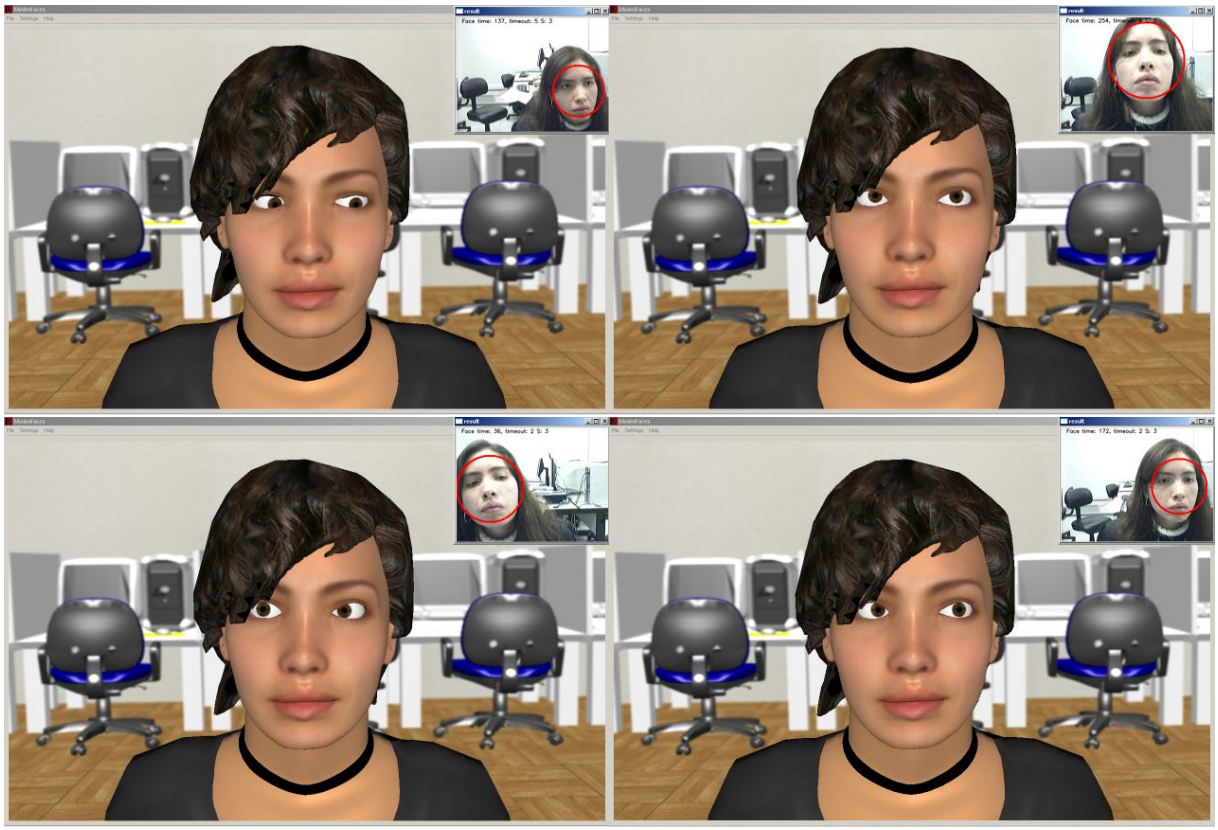

Figure 10: *Snapshots showing our application following user faces. The top right window shows the detection of the user's face.*

CASSELL, J., VILHJÁLMSSON, H. H., AND BICKMORE, T. 2001. Beat: the behavior expression animation toolkit. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 477–486.

COSI, P., DRIOLI, C., TESSER, F., AND TISATO, G. 2005. Interface toolkit: a new tool for building ivas. 75–87.

COSI, P., MAGNOCALDOGNETTO, E., AND TISATO, G. 2007. Emotional talking head: The development of "lucia". In *CD Proceedings Workshop "Toni Mian"*. Available in `http://www2.pd.istc.cnr.it/Papers/PieroCosi/cp-TONI2007.pdf`.

COURTY, N., BRETON, G., AND PELÉ, D. 2003. Embodied in a look: Bridging the gap between humans and avatars. In *IVA*, 111–118.

DE ROSIS, F., PELACHAUD, C., POGGI, I., CAROFIGLIO, V., AND CAROLIS, B. D. 2003. From greta's mind to her face: modelling the dynamics of affective states in a conversational embodied agent. *Int. J. Hum.-Comput. Stud. 59*, 1-2, 81–118.

DIPAOLA, S., AND ARYA, A. 2007. A framework for socially communicative faces for game and interactive learning applications. In *Future Play '07: Proceedings of the 2007 conference on Future Play*, ACM, New York, NY, USA, 129–136.

EKMAN, P., AND FRIESEN, W. 1978. *Facial Action Code System*. Consulting Psychologists Press, Inc., Palo Alto, CA.

EKMAN, P. 1999. Facial expressions. In *Handbook of Cognition and Emotion*, Dalgleish and M. Power, Eds. John Wiley & Sons, ch. 16.

HUANG, H.-H., NISHIDA, T., CEREKOVIC, A., PANDZIC, I. S., AND NAKANO, Y. 2008. The design of a generic framework for integrating eca components. In *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 128–135.

LEE, S. P., BADLER, J. B., AND BADLER, N. I. 2002. Eyes alive. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 637–644.

NOT, E., BALCI, K., PIANESI, F., AND ZANCANARO, M. 2005. Synthetic characters as multichannel interfaces. In *ICMI '05: Proceedings of the 7th international conference on Multimodal interfaces*, ACM, New York, NY, USA, 200–207.

PANDZIC, I. S., AND FORCHHEIMER, R., Eds. 2003. *MPEG-4 Facial Animation: The Standard, Implementation and Applications*. John Wiley & Sons, Inc., New York, NY, USA.

PARKE, F. 1982. Parameterized models for facial animation. *IEEE Computer Graphics and Applications 2*, 9, 61–68.

PERLIN, K. 1985. An image synthesizer. *SIGGRAPH Comput. Graph. 19*, 3, 287–296.

PERLIN, K. 1997. Layered compositing of facial expression. In *ACM SIGGRAPH - Technical Sketch*.

PYUN, H., KIM, Y., CHAE, W., KANG, H. W., AND SHIN, S. Y. 2003. An example-based approach for facial expression cloning. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 167–176.

QUEIROZ, R. B., BARROS, L. M., AND MUSSE, S. R. 2007. Providing expressive gaze to virtual animated characters in interactive applications. In *SBGames 2007*, vol. 1, 197–206.

RODRIGUES, P. S. L. 2007. *Um Sistema de Geração de Expressões Faciais Dinâmicas em Animações Faciais 3D com Processamento de Fala*. PhD thesis, Pontifícia Universidade Católica do Rio de Janeiro.

RUTLEDGE, L. 2001. Smil 2.0: Xml for web multimedia. *Internet Computing, IEEE 5*, 5 (Sep/Oct), 78–84.

SIFAKIS, E., NEVEROV, I., AND FEDKIW, R. 2005. Automatic determination of facial muscle activations from sparse motion capture marker data. *ACM Trans. Graph. 24*, 3, 417–425.

SMID, K., PANDZIC, I., AND RADMAN, V. 2004. Autonomous speaker agent. *Proceedings of Computer Animation and Social Agents Conference (CASA'04)* (July).

VILHJÁLMSSON, H., CANTELMO, N., CASSELL, J., CHAFAI, N. E., KIPP, M., KOPP, S., MANCINI, M., MARSELLA, S., MARSHALL, A. N., PELACHAUD, C., RUTTKAY, Z., THÓRISSON, K. R., VAN WELBERGEN, H., AND VAN DER WERF, R. J. 2007. The behavior markup language: Recent developments and challenges. In *Intelligent Virtual Agents*. Springer Link.

VIOLA, P., AND JONES, M. 2001. Rapid object detection using a boosted cascade of simple features. *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on 1*, I–511–I–518 vol.1.

WANG, A., EMMI, M., AND FALOUTSOS, P. 2007. Assembling an expressive facial animation system. In *Sandbox '07: Proceedings of the 2007 ACM SIGGRAPH symposium on Video games*, ACM, New York, NY, USA, 21–26.