Using Virtual Characters as TV Presenters

David Oyarzun[1], Maider Lehr[1], Amalia Ortiz[1], Maria del Puy Carretero[1], Alejandro Ugarte[1], Karmelo Vivanco[2], Alejandro García-Alonso[3]

[1]VICOMTech research centre. Paseo Mikeletegi, 57. 20009 San Sebastián, Guipuzcoa, Spain
{doyarzun, mlehr, aortiz, mcarretero, augarte}@vicomtech.es
[2]Baleuko. Askatasun Etorbidea, 23. 48200, Durango, Vizcaya, Spain
kvivanco@baleuko.com
[3]Artificial Intelligence and Computer Sciences Department, University of the Basque Country
agalonso@si.ehu.es

Abstract. Research in virtual characters and techniques that involve them, started several years ago. Nowadays, they can be used in environments which require a high level of interactivity, such as edutainment. A good example of this scenario is the television. This paper presents the application of several techniques related to corporal animation and natural voice recognition in order to achieve a whole body real time virtual presenter for television. Two main goals have been targeted within the application: a high quality of realism of the virtual presenter and real time interaction in a non intrusive way. A first prototype of this work has been tested in a TV program aim at children in the Basque Country television, and currently it is broadcasted daily.

Keywords: Virtual TV presenter, facial animation, corporal animation, phoneme recognition

1 Introduction

Despite having started several years ago, virtual characters and techniques that involve them are a field of research that continue having an increasing interest.

A virtual character as interaction tool can play different roles depending on the application. It can act representing users such as in games or chats, or act autonomously, such as virtual guides, assistants or teachers, etc.

The techniques required for the development of their real time animations has been widely studied and proof of it is the amount of research and documentation about them. Good overviews about general animation of virtual characters can be found in [1, 2].

This current maturity of animation technologies can improve the current scenario of modeling and producing TV content and extend the use of virtual characters to other kind of applications which require real time and high level of interactivity, such as live TV programs.

This paper is focused on the implementation of an interactive real-time virtual presenter for television. This idea comes from the necessity of using a virtual 3D model designed for a cartoon film [3] in an interactive environment. The goal is to create a virtual puppet live show based on this "cartoon star", where the user can participate and speak with the virtual character in real time. The virtual character is controlled by actors using techniques of real time phoneme recognition and lips synchronization, and real time corporal animation. The animation of the lips is generated in real time by means of the voice of the actors and the rest of animations are launched by other input devices.

This is the way the article is structured. Section 2 shows an overview of the state of the art. System requirements are explained in section 3. An overview of the system architecture is given in section 4. Section 5 is focused in facial animation, section 5.1 deals with phoneme recognition and section 5.2 deals with facial animation and lips synchronization. In section 6 the corporal animation techniques are shown. Section 7 gives an approach of the results and finally, the conclusions and current work are shown in section 8.

2 Previous Work

There are few developments using virtual characters in live TV environments. Some of the more famous are Ananova [4] and Meteosam [5]. Both of them have been developed with various animation and speech techniques.

The technical challenge for Annanova's developers has been to develop a fully-animated 3D character capable of creating dynamically-generated news bulletins in a style and tone appropriate to specific pieces of content. To create Ananova's voice, text-to-speech synthesis software from Lernout & Hauspie (L&H) has been integrated with Ananova's real-time information systems [6]. As it will be explained later, these techniques are not suitable for our approach since we are using a 3D "cartoon star" and it is necessary to use the same voice than in the film.

In the case of Meteosam, it is driven by human actors using motion capture systems and their natural voice in order to achieve the real time animation. In the work presented in this article, the idea is that the virtual presenter animation control should be easy to use and less intrusive for the actor that in the Meteosam case. The reason is that the virtual character should be driven daily by the same actor who dubbed it in the film. These imply that the actor probably is not used to manage with motion capture systems and, since it is a daily TV show, the time needed for the preparation should be the minimum possible in order to reduce the cost.

Other project that is applicable in TV environments is Virtual Human [7]. In this case, nowadays, the virtual characters are used in Augmented and Mixed Reality environments.

3 System Requirements

As it is said, in this work, the animation of the lips is generated in real time by means of the voice of the actors and the rest of animations are launched by other input devices.

The animation of the virtual character has to be very realistic because of people should be able to do the association between the cartoon in movies and the virtual character in TV programs. In order to achieve a natural body language it is necessary a lot of flexibility and the animation engine should be extensible and with low computational cost. The flexibility is said in terms of using any kind of input devices and being 3D model and graphics API independent.

A first prototype was developed with this idea. It was tested in a TV program aim at children and it was very successful. The children could call to the TV program and they had the illusion of speaking with their favorite cartoon. In Fig. **1** a screenshot of this prototype in the TV program is shown.

Fig. **1**: First prototype of the virtual presenter.

Due to its success, nowadays, this prototype is being broadcasted in a Basque Country television program daily [3].

It was developed with a commercial graphics engine –3DGameStudio [8]- and a home-made phoneme recognition application.

Although the 3DGameStudio is a powerful animation engine, it was designed for games and it can not provide a high level of flexibility in applications that requires a complete control of the virtual character in real time. With the first prototype the following limitations were found:

- *Flexibility in the animation input*. 3DgameStudio is oriented to standard input devices for games, as joysticks, with a limited number of commands. In the presented work exist a necessity of more control over the virtual character by means of real time motion capture systems or similar.
- *Scalability in the animation*. With these standard input devices, there is a limited number of animations that humans are able to memorize. It is due to these animations are launched by means of button combinations, In this

project it is needed to launch a large number of motions and they have to be expandable.

- *The allowed polygonal weight is around 30000 polygons*. It is not enough for more than one virtual characters and their immersion in virtual environments with high level of realism.

Therefore, an animation engine that fulfil the initial requirements without having these limitations is being developed. In this article an advanced stage of development is presented.

4 System Architecture

A modular architecture shown in Fig. 2 has been designed for solving the limitations of high realism, flexibility and scalability.

The system is divided in two main parts, the phoneme recognition module and the animation engine, and some additional modules.

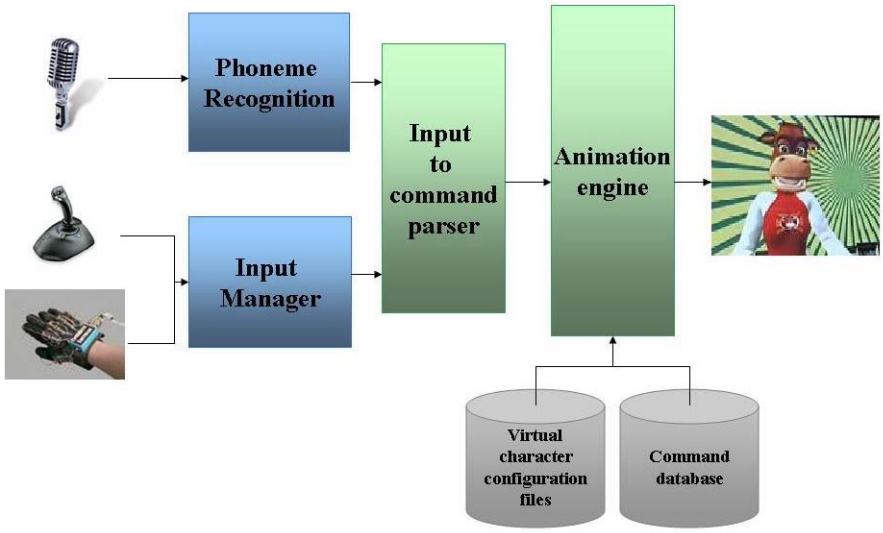

Fig. 2: Simplified system architecture.

The *phoneme recognition* module allows the real time animation of the lips through the voice of the actor. After the training stage of the recognition module, the actor only needs a microphone in order to generate de lip animation. The performance of this module is explained in section 5.1.

The *input manager* is an interface between any kind of input device and the animation system and an *input to command parser* transforms all the devices data in understandable commands for the animation engine.

The *input manager* and the *input to command parser* modules allow flexibility for using any kind of input devices.

The *animation engine module* generates the animation using a command database that translates the command in the animation stream and some autogenerated files – see below- that store the virtual character configuration.

These commands reach the animation engine in a common syntax, name of the command and parameters, and these animations are stored with the same format in the database. The animation engine obtains the command and searchs for it in the database. If it is found, then the corresponding motion is executed. It allows the increasing of animations and the modifications on the existents without changes in the engine. The animation engine core uses low level APIs and animation algorithms with low computational cost for solving the high realism requirements.

An internal module translates the stored animations to the transformations understandables by the graphics API, creating a separation between the animation algorithms and the graphics API functions. In addition, one file stores the virtual character hierarchy without use of the geometrical information.

In this manner, the engine is graphics API and 3D model independent.

5 Virtual Character's Facial Animation

The facial animation is generated by means of morphing techniques [9, 10] in order to obtain a realistic enough animation for real time with low computational cost. Before the animation generation, a process of phoneme recognition is done. All of this occurs in real time and the animation is corresponded with the actors speech.

In the next sections, facial animation modules are explained in more depth. Section 5.1 deals with the phoneme recognition system, and section 5.2 explains the facial animation engine.

5.1 Phoneme Recognition

The phonetic content is needed, that is to say, phonemes and their duration, with the aim of synchronizing the character's lip animation with its speech.

In this case, the animation is generated with the voice of a real person in real time, therefore, the character has a much more natural appearance. An audio analysis has been performed to obtain the phonetic information needed. Most of the commercial applications in this field are English-based and the use of virtual characters in minority language contexts is very limited. This is obviously due to the lack of both resources and tailored technology.

In this work, the development of a system capable to produce the suitable data to animate faces from a Basque natural voice using open source technologies has been studied. We believe that using open source technologies such as HTK [11] or Sphinx [12] may help diminish the digital gap between majority and minority languages.

Specifically, HTK Toolkit [11], based on Hidden Markov Models (HMM) methods [13], has been used to discover the best approach to obtain a synchronous animation in real time from speech in Basque, using the minimum amount of resources possible.

The output of the speech analysis is the phonetic data set, suitable to perform the lip animation of the virtual character for each frame of the animation in real time. More concretely, the extracted phonemes are correlated whit the set of possible visemes. To obtain these data, a set of tests has been performed using HTK Toolkit.

Resources for Basque language, such as corpora, are scarce. Nowadays, no open source oral database is available. Therefore, in order to train and test the HMMs, 250 sentences in Basque have been recorded, 150 for training and 100 for testing. Recordings were done using a Sennheiser desktop microphone (16 kHz/ 16bits/ mono). Recording conditions were not optimal in regarding noise level. The audio files were recorded in a working room with people speaking and with other types of noises, such as steps or street sounds.

The application captures the speech signal from the input through a sound card and identifies the appropriate phonemes. As phonemes are recognized, they are sent to the animation platform. The application developed consists of three modules (Fig. 3):

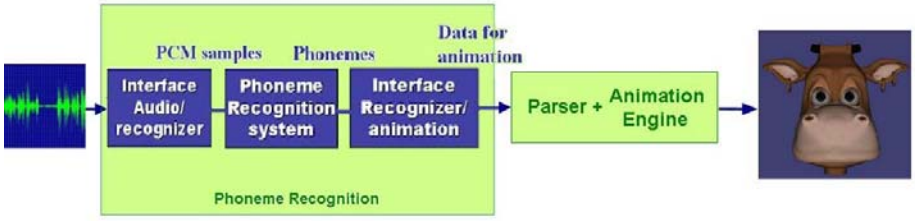

Fig. 3: Diagram of the voice capture module

The phoneme recognition system, has been implemented with HTK Toolkit. During training, feature extraction was performed over 25 ms segments every 10 ms. The Basque version of SAMPA was used as phoneme set for the recognizer. Monophone models were created, which consisted of non-emitting start and end states and 3 emitting states (except from the short pause model) using single Gaussian density functions (due to the small amount of training data). The states are connected left-to-right with no skips. For training, we initially set the mean and variance of all the Gaussians of all the models to the global mean and variance of the complete data set. These models were then trained iteratively using the embedded Baum-Welch re-estimation and the Viterbi alignment. The short pause model was added only after a few training cycles. The resulting single Gaussian monophone system was tested using a Viterbi decoder.

Once the off-line recognition system has been developed, it has to be connected with on-line audio, so that the recognizer generates the phonemes corresponding to the speech waveform in real time (Interface Audio/Recognizer Module). To develop this interface we used the ATK API [14].

Once the phonemes corresponding to the audio samples are recognized, they are sent to the animation module. For this, an interface with sockets was developed, based on the TCP/IP communication protocol (Interface Recognizer/Animation Module).

Through this module the animation module is fed with the recognized unit for realistic animation.

It should be noted, though, that due to the lack of sufficient speech data in Basque, the end user must train the system. A minimum of 150 training sentences are needed. In order to facilitate this task, an interface has been developed. This interface allows the user to select the sound recording device as well as the format of the audio files.

5.2 Facial Animation Engine

As it is said, facial animation is generated by means of morphing techniques. These techniques obtain the animation creating a continuous interpolation between a reduced group of key expressions. The key expressions are the visemes, or visual representations of the phonemes recognized in the Phoneme Recognition Module, and the basic facial expressions. For example the interpolation between the face with the expression 'A' and the face with expression 'happy' give us a face with an 'A happy'. More information about these techniques can be found in [9, 10].

An internal module of the engine allows to synchronize in real time the different commands for speech and facial expressions resulting from the input devices – microphones, joystick, etc…-.

A minimum and constant delay –about 100 milliseconds are enough- is applied with the aim of smoothing the transitions between visemes and it is compensated by means of applying the same delay to the output audio.

6 Body Animation of the Virtual Character

These animations are launched by means of any kind of input devices such as joysticks or data gloves.

The goal of the engine is to obtain realistic motions of virtual characters with low computational cost.

In order to achieve a realistic body motion, the rotations in the joints should be produced without breaks in the mesh, also called *cracks*. Thus, a technique that deforms the mesh on the joints is required.

So, with the aim of solving cracks in the joints, some algorithms use a mesh for each rigid part and merge them with deformable patches or spheres [15]. Others use different layers in the whole body or techniques as animation based on metaballs [16].

The work that is shown here uses only one complete mesh for representing the whole virtual character, and the transformations are applied to this mesh.

The base of the algorithm consists on subdividing the rigid parts of the body in virtual sections depending on the nearness from a joint in a hierarchical way. A weight is assigned to each vertex, and it is rotated depending on this weight. Using this technique, we obtain independence of the kind of mesh modeled and independence in the way of storing the model, and in addition, few operations per frame and a realistic result. In Fig. 4 a visual representation of the weight values for each vertex is shown.

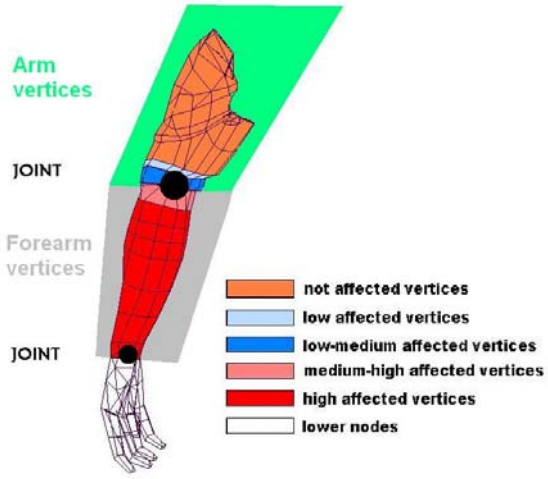

Fig. 4. Vertices affected by the movement of the elbow. The limit distances between each group of vertices and the joint of the elbow are user-defined.

In order to store the hierarchical information a new file format has been created. The format of this file is called *simple hierarchical format* (shf). In Fig. 5 a fragment of a shf file is shown. This file contains fields for the information required for each node, such as the name, the rotation point, model part geometry indices and the children on the hierarchy.

```
Neck
0 0 0
0 0 0
1986 2038 1987 2037 1988 1989 1990 1991 1992 1993 1994 1995 1996 ...
{
      Head
      0 -0.011 0.683
      0 0 0
      0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
      814 815 816 817 818 819 820 821 822 823 824 825 826 827 82 ...
      {
            .
            .
            .
```

Fig. 5: SHF file example.

This file permits independence of the model format from the information that it stores, and it does not contain specific information about the format of geometry. In addition, it stores less information than the standard hierarchical file formats, since it is not necessary in this concrete way of animation, and it would increase the file size. With the aim of developing web applications or applications for special devices, it is important that the file is as small as possible.

So, the necessary information for the animation is stored in two files. One contains the geometrical information of the mesh that is generated with the modeling tool and the other is the shf file, which contains the information of how a rotation affects the vertices of the mesh.

6.1 Animating the Avatar

Two types of corporal motions are distinguished in this research: predefined movements and real-time generated movements.

The first ones are obtained using motion capture systems and adapting them to the skeleton of the virtual character. Their main feature is that they are iterative motions which are not affected by the relative position of the virtual character in the virtual world.

The second ones have to be generated in real-time because a motion of this type has infinite postures depending on the relative position of the virtual character in the virtual world. An example is '*pointing'*. If the virtual character is pointing to a specific position, the motion of the arms will be different when the virtual character changes its position or orientation.

Predefined movements. Using a motion capture system, the spatial coordinates of each part of the body per frame can be obtained. Having these points, a simple preprocess allows to obtain the angles corresponding to each joint in each frame. This information, generated previously to the animation process is stored in a file within the correspondence between the joints detected by the motion capture system and the hierarchy of the shf file.

Real-time generated movements. Normally these kinds of movement are generated using either kinematics or dynamics techniques or real time capture systems. In our case, for motion algorithms, we use inverse kinematics because the dynamics approximations have a very high computational cost –they include physical laws-, and forward kinematics need to know what angles to move and, in motions such as *pointing motion*, only the final position of the movement will be known.

We have decided to implement an adaptation of the Cyclic-Coordinate Descent Method [17]. It is fast and immune to problems of other algorithms as the problem of proximity to singularities with the Jacobian Method [17, 18]. This algorithm makes a heuristic iterative search that gives us the possibility of improving either the accuracy of the end result or the speed of the resolution. With this method the final rotations of each joint are obtained, and then they can be applied by means of interpolations between the initial and final angles.

But only with this approach, the avatar can take unreal poses and, although they can be useful showing exaggerated characters as cartoons, it is not desired for a realistic application. It is solved giving restrictions to the joint's angles that avoid them. For example, in a default pose with the z positives toward the user, the elbow should move by the x-axis in the counter-clock wise.

The processing of the real time capture animation is done in the same way that the predefined animation due to the engine reads motions per frame independently of the way they have been generated.

Execution of the animation. As the two cases obtain the angles per frame of each joint, and they are applied directly over the virtual character information stored in the mesh obtaining step. The low level animation is done by means of the body animation algorithm in both cases. As in previous developments [19], the final vertex position is obtained transforming the vertex with the angle multiplied by the weight.

In a conceptual way, the equation 1 is applied for each vertex in order to obtain the animation.

$$v_f = v + w_i \cdot M_r \cdot v_i \,. \qquad (1)$$

where v_f is the final vertex, v is the vertex with the previous transformations in the hierarchy, w_i its assigned weight, M_r the rotation matrix corresponding to the joint and v_i the vertex in its default position.

7 Results

Next, the obtained results in phoneme recognition and visual quality are shown.

Referred to phoneme recognition, Table 1 illustrates the results obtained for the two types of parameterization. These results represent the number of visemes recognized by the system. In the case of the phoneme clusters this result is obtained directly. However, in the case of the complete phoneme set we mapped the phone transcriptions of the reference text (the text to recognize) and the text recognized to obtain the viseme recognition rate. The phonemes that have the same visual representation were mapped to the same symbol. More information about the phoneme recognition and its results can be found in [20].

Table 1. Experimental results (viseme recognition rate)

Parametrization	All phoneme set	Phoneme clusters
Mel-frequency Cepstrum coefficients	76.45% (Acc :70.76%)	71.32% (Acc: 65.28%)
Reflection coefficients	65.33% (Acc: 58.64%)	60.56% (Acc: 54.14%)

The visual quality obtained is shown in Fig. 6 comparing it with the film cartoon and the 3DGameStudio prototype.

Fig. 6. Screenshots of tests. Left image shows the film cartoon. Medium image shows the results using 3DGameStudio and right image shows the presented animation engine.

8 Conclusions and Future Work

The presented work shows an engine for animating a virtual character in real time oriented to applications with the needed of a great visual quality. The requirements of this engine are real time animation with high realism, scalability and flexibility in order to use it in live TV programs.

There are not developments in the state-of-the-art that fulfill these requirements using non intrusive techniques at the same time.

The developed engine allows a high level of realism with low computational cost obtaining real time animation and being 3D model format and graphic API independent.

Thanks to the phoneme recognition system developed, the virtual character can speak through the actor's voice. The animation reacts to the caller's answers, therefore needs to run in real time. Lip animation runs as the actress who dubs the virtual character in the program speaks into the microphone. This voice is sent in real time to the software developed in this work. The software analyses the stream and generates the data to synchronize the lips with the audio. This information is interpreted by the animation engine.

In addition, with the architecture presented in this paper we can achieve the following aspects:

- *Flexibility in the animation input.* With the architecture proposed the user can use any kind of input devices
- *Flexibility in the animation engine*. It can use any kind of 3D model and animating it with any graphics API. The animation engine is also extensible and with low computational cost.
- *Scalability in the animation*. Animations can be added easily and without need of modify the animation engine.
- *High realism*. The use of low level API's and low cost animation algorithms allows animate complex cartoons in real time.

The next steps are the inclusion of path-finding and static and dynamic collision avoidance techniques. It will allow the complete integration of the virtual character in virtual worlds, opening a new way of visualizing TV virtual presenters and entertainment TV programs.

In addition, a plugin for commercial modeling tools that automatizes the *shf* file generation will be developed.

Acknowledgments. This article is the result of the collaboration in R&D project with Baleuko and Talape (Basque companies in television and film production). We want to express our acknowledgements for the opportunity of evaluating our research in a live daily TV program and also in public events with children.

References

1. Giang, T., Mooney, R., Peters, C., O'Sullivan, C.: Real-Time Character Animation Techniques. Trinity College Dublin, Computer Science Department Tech Reports (2000)
2. Collins, G., Hilton, A.: Models for Character Animation. Software Focus (2001) 44-51
3. Betizu. Retrieved in November 2006 from: www.betizu.com.
4. Ananova. Retrieved in November 2006 from: www.ananova.com
5. Meteosam. Retrieved in November 2006 from: www.meteosam.com.
6. Ananova. Retrieved in November 2006 from: http://www.thescreamonline.com/technology/technology10-01/.
7. Virtual Human. Retrieved in November 2006 from: http://www.virtual-human.org/
8. 3DGameStudio. Retrieved in November 2006 from: www.3dgamestudio.com.
9. Alexa, M., Behr, J., Müller, W.: The Morph Node. Proc. Web3d/VRML, Monterey, CA (2000) 29-34
10. Glauert, J., Kennaway, R., Elliott, R., Theobald, B.-J.: Virtual Human Signing as Expressive Animation. In Symposium on Language, Speech and Gesture for Expressive Characters, University of Leeds (2004) 98-106
11. Young, S., Kershaw, D., Odell, J., Ollason, D., Valtchev, V., Woodland, P.: The HTK Book. Retrieved in November 2006 from: htk.eng.cam.ac.uk.
12. Sphinx. Retrieved in November 2006 from: http://cmusphinx.sourceforge.net/html/cmusphinx.php.
13. Rabiner, L.R.: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE (1989) 257-286
14. Young, S.: The ATK Real-Time API for HTK. Retrieved in November 2006 from: htk.eng.cam.ac.uk.
15. Petriu, M.D., Georganas, N.D., Whalen, T.E.: Development of a Humanoid Avatar in Java3D. Proc. IEEE International Workshop on Haptic, Audio and Visual Environments and their Applications, Ottawa, Canada (2003)
16. Aubel, A.: Anatomically-Based Human Body Deformations. Ph.D. Thesis No 2573, EPFL, (2002)
17. Welman, C.: Inverse kinematics and geometric constraints for articulated figure manipulation. B. Sc. Simon Fraser University (1989)
18. Tolani, D., Goswami, A., Badler, N.I.: Real-Time Inverse Kinematics Techniques for Anthropomorphic Limbs. Computer and Information Science Department, University of Pennsylvania, Philadelphia, 19104-6389
19. Anderson, E.F.: Real-Time Character Animation for Computer Games. National Centre for Computer Animation, Bournemouth University, Internal report (2001)
20. Lehr, M., Arruti, A., Ortiz, A., Oyarzun, D., Obach, M.: Speech Driven Facial Animation using HMMs in Basque. Proceedings of 9th International Conference, TSD 2006, Brno, Czech Republic (2006)