Realistic Visual Speech Synthesis in WebGL

Abstract

This paper presents the work that has been done to develop a web application that shows the face of a virtual character pronouncing the sentences the user sets. The level of realism was high and the performance was fast enough. The application makes use of WebGL, speech processing, text to speech and co-articulation technologies to obtain the virtual pronunciation. The paper shows the implementation of the technologies and the optimization strategies that have been followed to ensure the efficiency of the application.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; H.5.3 [Information Interfaces and Presentation]: Group and Organization Interfaces—Web-based interaction;

Keywords: WebGL, facial animation parameters, visual speech synthesis

1 Introduction

3D virtual environments and characters, avatars, have become very popular since the first 3D films, like Toy Story or Shrek, were launched. Since then, the quality of the works has raised to a level where the 3D reproductions can be mistaken with reality. Specifically, the facial animation of virtual characters is increasingly realistic due to its importance in 3D applications, especially in those where the avatar talks to the user. In fact, Hodgins et al. [Hodgins et al. 2010] proved that an audiovisual content with virtual characters loses emotional value if the facial animation is not of high quality.

On the other hand, the widely spread utilization of web browsers generates a clear tendency to develop tools that work in web pages and without the need of any plug-in or stand-alone application. This way, the tools are accessible with any device that can run a web browser, computers, laptops, tablets or mobile phones, provided they can access the internet. This helps the developer, since he has not to care about incompatibilities.

The combination of these two popular technologies is the main purpose of the project SPEEP, partly funded by the Basque Government. The project creates a system for foreign language pronunciation learning where the key part is the visualization of a virtual character pronouncing the corresponding sentences in a web. Figure 1 shows the interface designed for the project integrated in the language learning system. Since the sentences that have to be synthesized in the virtual character are not predefined i.e. the user writes the desired sentence, the system cannot work with animations that have been generated in a previous phase of the project. Thus, we can define our project as a work in real-time Visual Speech Synthesis.

So, the two main challenges of the project SPEEP arise. On the one hand, creating new animations in real-time makes impossible to make use of most popular techniques for high-realistic facial animation: real-time motion capture and modeling by professional modelers. Nowadays, for highly realistic 3D applications or 3D visualizations, first a real person's motions are captured and translated to a virtual character. Then, a modeler corrects the errors the system could have done. Once the application is running, the animations are launched when they are needed. Nevertheless, in this project, the user asks the virtual face to pronounce a specific sentence and the system must be able to generate it automatically.

On the other hand, the utilization of web browser for the rendering of the virtual face. WebGL has been selected to render the virtual model via web. WebGL [Leung and Salga 2010] is a powerful Application Programming Interface (API) to present 3D graphics in a web page (nearly seven times faster than Flash because of GPU acceleration). It is based on OpenGL, which is a very widely used open source 3D graphics standard. Moreover, WebGL is compatible with different and most common browsers such as Google Chrome, Mozilla Firefox, or Safari. So, WebGL allows the use of web technologies, which are an easy way that non expert users have to access content. However, WebGL technology is slower than native OpenGL because it uses JavaScript for execution and applications usually need to be optimized to obtain a fast performance.

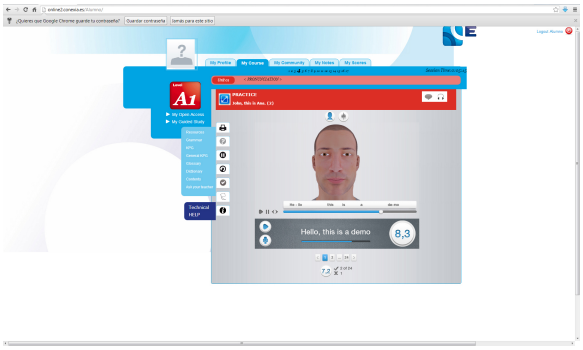

Figure 1: *Interface of the system.*

In this paper we describe how both challenges have been faced. In the reminder of the paper, we first make a short review of the state of the art in the section 2; in section 3 we make a short description of the system of the project SPEEP; in section 4 and 5 we detail the methods used for facial animation and speech generation and in section 6 we describe how all the modules were integrated. Finally, we conclude the paper discussing the results in section 7.

2 Related Work

According to Radovan and Pretorius [Radovan and Pretorius 2006] we can classify the different methods for facial animation in three groups: based in geometry, based in images and based in real movements.

The pioneer work [Parke 1972] in the field of facial animation can be classified in the first group. This and other primitive works transform the vertices one by one in each frame of the animation. On the other hand, more recent techniques can also be classified in geometry-based animation. Works that make use of pseudo-muscles [DeRose et al. 1998] i.e. that simulate geometrically the effects of muscles or directly simulate the physical behavior of the muscles and the skin [Wang 2010] exist nowadays.

Other methods, usually focused on the film industry, are based on images in-stead of geometry: morphing techniques [Su and

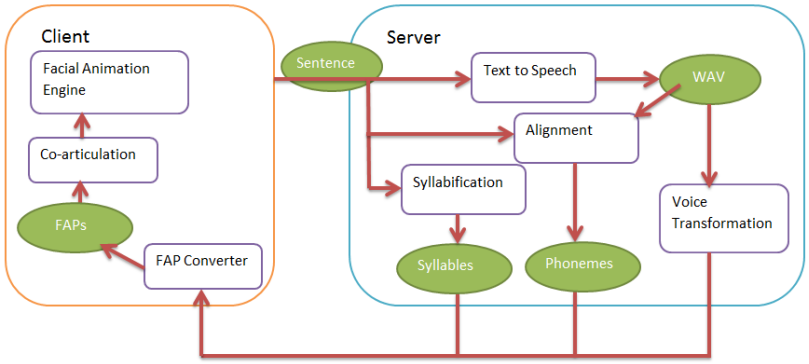

Figure 2: *Architecture of the system.*

Liu 2001] where different key images are interpolated to obtain the animation; texture manipulation methods [Fei 2001], where the changes in the texture of the facial mesh create the animation and works based on blend-shape interpolation [Huang et al. 2011], where the intermediate frames between two modeled faces are computed.

However, recently performance-driven techniques have replaced others because it is very difficult to obtain the realism obtained by capturing the movements of a real actor in such a short time. Beeler et al. [Beeler et al. 2011] analyze the captured data to recognize predefined motions and launch these predefined movements in the virtual face. In the work presented by Arghinenti [Arghinenti 2011] the captured data is only a layer of the facial animation engine. Other layers cope with expressions, phonemes and muscles. Deng et al. [Deng et al. 2006] combine a motion capture system with a face scanning system in order to obtain a realistic animation with low-level details, such as wrinkles. Once they obtain the minimum number of key expressions, blend-shape interpolation is used to generate the final animation.

In recent years, both for modeling animations and as a complement of other techniques, another type of facial animation has emerged, skeleton-driven facial animation [Moore 2001; Avrin 2011; van Straten 2011]. As in corporal animation, the face of the virtual character has a structure of joints and bones attached. The vertices of the facial mesh are transformed in concordance with the skeleton, since they are associated to certain bones. The work we present in this paper can be classified in this group. Unlike in corporal animation, the skeletons differ widely in skeleton-driven facial animation methods. Some have elastic bones, others have rigid bones; some use segments as bones, others use curves as bones; some have tree structure with its node in the neck, others have disjointed groups of bones.

In Visual Speech Synthesis, apart from the facial animation technique, one of the key aspects to obtain a realistic animation is the interpolation between the phonemes that have to be synthesized. One of the first works in this field and one of the main references is the one presented by Cohen and Massaro [Cohen et al. 1993]. They create exponential functions for facial parameters that rise until the phoneme's exact time and fall afterwards. The so called co-articulation is obtained by the combination of different phonemes and the functions for their associated facial parameters.

Since the first Works in co-articulation were published, several methods have been presented that take speech unities and convert them in a fluent and realistic facial animation: rule-based, Markov models [Yamamoto et al. 1998], neural networks [Massaro et al. 1999], to name but a few.

One of the key moments in the history of facial animation is the definition of the MPEG-4 standard [Bauer et al. 1999]. The standard sets three key features for facial animation: Facial Definition Parameters (FDPs), a set of points that define all the features of a human face; Facial Animation Parameters (FAPs), that are used to manipulate key feature control points, FDPs, in a mesh model of the face to produce the animation; and Facial Animation Parameter Units (FAPUs), that are defined in order to allow interpretation of the FAPs on any facial model in a consistent way. Since then, most studies are based in the parameters that were defined.

To finish this review of the state of the art, works that implement facial animation in web environments are studied. To the best of our knowledge, from the first systems [Alexa et al. 2000], that work with VRML, very few have been presented. The system presented by Gachery and Magnenat-Thalmann [Gachery and Magnenat-Thalmann 2001], for example, needed a Java applet to make use of MPEG-4 standard in a web. Krnoul [Krňoul 2011] presents a system based on WebGL that renders an avatar speaking both with her mouth and sign language. For the facial part, morphing, that is not the best technique for realism, is used. Finally, The work by Benin et al. [Benin et al. 2012] is very similar to ours, but, apart from the chosen language and differences in the methods for facial animation, we consider that the quality of our animation is higher.

To conclude, many methods have been presented that accomplish realistic facial animation, but very few have been developed to be used with the emerging WebGL technologies. Besides, high-level realism has not been achieved. Our work goes in this direction.

3 System Overview

The project SPEEP aims to develop a tool for foreign language pronunciation learning. Once the user of the application writes the sentence or word that he wants to practice, he has to take two steps. First, he can watch the virtual character pronouncing the sentence and afterwards, he pronounces the same sentence, so that the system renders both pronunciations and compares them for corrections. In this paper, we focus on the first part, where the user writes a sentence and the system automatically generates an animation of an avatar pronouncing it. Figure 2 shows the general architecture of the system.

In a web browser, the user will find a text field to write the desired sentence and once it is written and confirmed, it is sent to the server. In the server, the text is converted to speech with the desired voice

by the module for text-to-speech conversion and the module for voice transformation. On the other hand, the text is used to get the phonemes and the syllables of the sentence and align them with the audio file generated by the text-to-speech module. Once the three files needed (audio, phonemes and syllables) are generated, they are sent to the client.

The client receives the phonemes and via the FAP converter it gets the values of the FAPs that must be rendered in their exact time. Then, during the rendering of the virtual character, with the information about the syllables the co-articulation engine interpolates the values for the FAPs. The animation engine takes the audio file and the information generated by the co-articulation engine, synchronize them and generates the position of the avatar in each frame.

This way, the user will be able to watch the sentence he wrote properly pronounced almost in real-time. The time loss mostly comes from the connections between the client and the server, since the generation of the files and their parsing is almost immediate and the co-articulation engine is called in real-time. Besides, the user can reproduce the animation as in a music player, playing the animation from any point or pausing it in any frame.

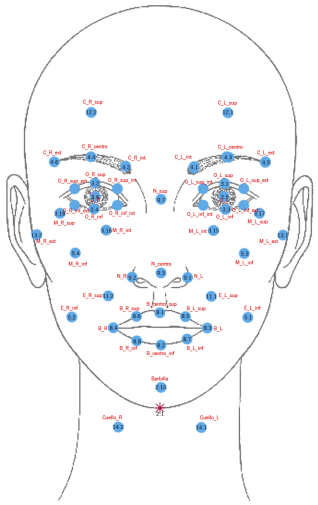

Figure 3: *Bone positions of the skeleton.*

4 Facial Animation

The facial animation module is completely programmed in Javascript and embedded in the web page that will be shown to the user. In the following we describe the different parts that compound such module.

Our facial animation approach can be classified as based on pseudo-muscles. We take a set of Facial Definition Parameters (FDPs), define a hierarchy for them and form a skeleton. Then, this skeleton guides the transformation of the vertices that compound the virtual face, as in skeletal animation for bodies. The vertices' transformations are computed combining the transformations of the bones of the skeleton that are near them. This way, in each frame, only the transformations of bones must be set, since the vertices will be updated automatically.

As the system has to work in WebGL, the number of bones has been lowered in order to overcome system's low power. Trying to maintain the balance between realism and efficiency, some sets of FDPs defined in the standard were replaced by a unique bone, mainly in the lips, ears, nose and jaw. The reduction of the number of bones has been done so that the main areas for the facial animation are not too affected. In the figure 3, final bone positions can be seen.

For the preprocess, we developed our own Collada loader. The XML format of Collada files permits to add new information in the hierarchy and this feature has been used to encode all the information needed for the rendering. Besides the usual information about the vertices that compound the virtual character, the bones of the skeleton and the skinning i.e. the process of attaching the vertices to the bones, the Collada loader is prepared to load all the information about FAPs and the correspondence between FAPs and bones. This way, the Collada loader permits the facial animation engine to trigger predefined animations, as always, and to generate facial animations once it gets a sentence to synthesize.

Once the geometry, the skeleton that will drive the geometry and the information needed for facial animation are loaded, the web page shows the text field to write the sentence. The client sends the text and receives three files: phonemes, syllables and audio. The first two files are those concerning this section. They contain an ordered list of phonemes (or syllables), with their starting time and finishing time.

The FAP converter module takes these two files and parse them. As the correlation between phonemes and FAPs is fixed and loaded in preprocess, the module only takes the needed values from the stored data and weights them with the input. For example, if the converter receives the phoneme *a*, the output will be the corresponding list of FAPs (*vertical top middle inner lip displacement*, *vertical bottom middle inner lip displacement*...) and the value attached to each FAP.

Once the system can access the value of each FAP in the exact moment a phoneme is said, the synthesis of the sentence begins. During the rendering, one of the key modules for realistic facial animation takes part, the co-articulation module. Co-articulation refers to variations in the articulation of speech segments depending on the surrounding ones.

So, in each frame the co-articulation engine is called to get the values of the FAPs in that exact moment. The co-articulation engine takes the phonemes that surround the actual time and interpolates the values of the FAPs following the method presented by Cohen and Massaro [Cohen et al. 1993]. The interpolation for each FAP, $F(t)$, is made in the following way:

$$F(t) = \frac{\sum_i w_i e^{\theta|t-1/2(t_{i_f}+t_{i_0})|}}{\sum_i e^{\theta|t-1/2(t_{i_f}+t_{i_0})|}} \quad (1)$$

In each frame of the animation, the phonemes that surround the current time, t, are taken into account. In the formula they are indexed by i. For each phoneme, the system receives the initial time, t_{i_0}, and the final time, t_{i_f}. In this first phase we consider that the point of maximum value of the phoneme is the central point of the interval, $1/2(t_{i_f} + t_{i_0})$. Therefore, the expression $|t - 1/2(t_{i_f} + t_{i_0})|$ is the distance between the current time and the maximum point of the phoneme. This value is balanced by the value θ, the same for all phonemes in this first phase, that is used to expand or contract the effect of each phoneme during the sentence. The combination of the exponential and the absolute value in the exponent creates functions that accelerate rapidly to the maximum value and decelerate in the same way. Finally, all the exponential functions are summed, with their corresponding weights, w_i, in the

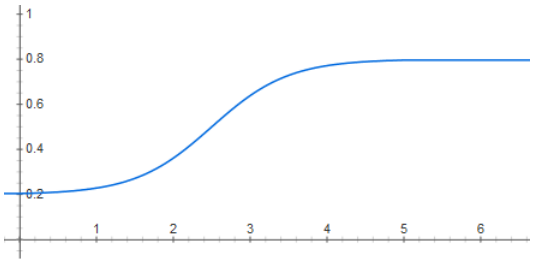

Figure 4: *Interpolation curve obtained by Cohen and Massaro's method.*

numerator and without them in the denominator. The weight is the value that a FAP must take in the maximum point of the phoneme. Figure 4 shows the type of transition between two values (0.2 and 0.8 in this case) obtained with this kind of interpolation.

When this type of interpolation was implemented, the engine was tested for efficiency. We noted that the system worked properly in powerful devices, but there was no margin for adding more computational cost in less powerful devices. Therefore, the utilization of syllables to enhance the quality of the animation must be of low computational cost. Thus, we decided that the same interpolation method will be used and the syllables will be used to make small changes.

Several rules have been defined that change the value of θ in equation 1 and the position of the maximum point of the phoneme in its interval, depending on the type of syllable and the phonemes involved. For example, the consonant of a syllable CV (consonant-vowel) is placed in a different position of its interval comparing to a consonant of a syllable VC. The value of θ can be different as well.

So, when equation 1 is computed, for each phoneme the system checks which type of phoneme it is (bilabial, dental, alveolar...), which type of syllable it belongs to (CV, VC, CVC...), which is its position in the syllable and what comes before or after this syllable if needed (the values differ if a silence comes after the syllable, for example). A rapid search through this rules gives the needed values and the facial animation engine keeps working as in the first phase.

Finally, the values of the FAPs are translated to the bones according to the equivalences that have been defined in the Collada file. Each FAP is associated to a set of bones and for each bone the type of transformation that will be applied (translation or rotation) is set and the axis and the limits that define the transformation are defined. For example, the FAP *vertical top middle inner lip displacement* is associated with the bone that is placed in the middle of the top lip and the bone can move from -0.5cm to 0.5cm through the axis (0,1,0). Then, the value of the FAP (from 0 to 1) will describe in which position of this axis the bone must be placed.

When the transformations are computed the skinning is called to translate the movements to the vertices of the virtual face. Note that, on one hand, the relation between phonemes and FAPs and on the other hand, the relation between FAPs and bones' transformations are separated. This way, the virtual face can be changed without changing anything related to phonemes.

5 Speech Generation

Several modules have been integrated in the server of our application in order to create the required files for the facial animation, as can be seen in figure2.

Firstly, the server takes the text written by the user and converts it into an audio file. The architecture of the system is prepared to work with any language. So, other languages, such as English, are planned to be implemented during the project, but Basque language is the first implemented one. So, Basque Loquendo TTS[1] is used to generate acoustic speech.

As shown in the previous section of this paper, in order to deal with co-articulation effect, speech and text alignment must be done in syllable level. Since we did not have access to the whole phoneset of Loquendo TTS system, an acoustic model was trained for Basque using HTK[2] tool in order to align automatically acoustic speech and input text.

The acoustic model for alignment was based on a triphoneme model. 7 hours of clean speech data was used to train acoustic models. The parameterization of the signal was based on Mel-frequency cepstral coefficients (MFCCs), delta and delta-delta coefficients. The Basque phonelist described below was used as phoneme set. Triphoneme models consisted of non-emitting start and end states and three emitting states (except from the short pause model) using Gaussian density functions, whose number of components was increased until no further recognition improvements were observed. The states are connected left-to-right with no skips. The models were trained iteratively using the embedded Baum-Welch re-estimation and the Viterbi alignment, while the resulting was tested using a Viterbi decoder.

In the first step, time codes are automatically obtained at triphoneme level using HTKs HVite decoder, based on Viterbi algorithm. Then, the input text is phonetically transcribed and mapped with the alignment output. Finally, the syllabification process allows us to obtain the syllables in addition to the time they were pronounced. The coarticulation model takes into account the current syllables in addition to the previous and the following ones.

On the other hand, having the whole phoneset is essential for building syllabification rules that can handle any phonetic representation. Our phoneset included all phonemes, allophones and diphthongs existing in Standard Basque. Even if more phones may exist in dialects, only the ones belonging to Standard Basque were chosen. The phoneset was coded in the international standard X-SAMPA [Gibbon et al. 1997].

Once phoneset was established, a rule-based Grapheme to Phoneme (G2P) system was built. The input of the G2P is the same as the speech synthesizer, and the output is a list of string of phonemes corresponding to the official spoken form [Euskaltzaindia 1998]. This translation is made with phonetic rules made ad hoc. This module automatically detects tokens with a different reading systems (such as acronyms), and provides the phonetic transcription of the natural spoken form. It also includes alternative pronunciations for words with single graphemes or combinations that may correspond to one or various phonemes.

Finally, in order to identify each syllable for a natural mouth articulation of the avatar, a syllabification module was built. Syllabification is a language-dependent process in which a string of phonemes is cut into syllables. In Basque each syllable is composed by nucleus that must appear compulsorily, an optional onset preceding the nucleus and an also optional coda following the nucleus. Depending on the phonological characteristics of Basque and the phonetic context, a full classification of potential phonetic environments was made ad hoc taking into account the standard pronunciation rules (Euskaltzaindia, 1998). This classification included phonemes, allophones and diphthongs; and describes the

[1]Loquendo. `http://www.loquendo.com`
[2]HTK. `http://htk.eng.cam.ac.uk/`

possible positions of each phone within a syllable. Departing from this classification, a set of boolean rules were designed. Since Basque phonology allows only one phoneme to be the coda and the nucleus can only be a vowel or a diphthong, the coda was set as the start point of the boolean rules. The syllabification module analyzes each string of phonemes starting from the end of the string, and building each part of the syllable until the beginning of the syllable is identified. The process repeats recursively until reaching the beginning of the string.

6 Implementation

One of the main advantages of WebGL is that it is supported in the majority of popular web browsers and so, it can be used in any device that can run these web browsers. Apart from the utilization of Javascript, which is slower than other languages, this means that the developed application should suit not very powerful devices. That is why every step in the project has taken into account that efficiency is the key of success. In the following, we describe some key aspects for the fast and correct performance of the system i.e. for a balanced performance between velocity and realism.

A study of the computational cost of the different methods that are called in each frame during the facial animation showed that the most time consuming function was the skinning i.e. the computation of the new position of the vertices according to the transformations of their neighbor bones. This high cost has two main causes. On the one hand, a big number of vertices are needed to obtain a realistic virtual face and although all the vertices are not transformed in all frames, the amount of vertices that are moved is big enough to become a problem. Specifically, the virtual face that is used in the project SPEEP is compound of 22802 vertices that form 43449 polygons and it can be seen in figure 5.

On the other hand, the skeletal method used for the animation works with matrices (transformations). The matrix multiplications appear when updating the skeleton first. As shown in section 4, the number of bones is less than the number of FDPs defined in the MPEG-4 standard, in order to make this process faster. Considering that the project's main interest is the realistic pronouncing, groups of FDPs that define the same area have been replaced by only one bone in non-relevant areas (e.g. ears and nose). Even in the mouth, each bone replaces two FDPs, the inner one and the outer one.

Besides, a unique matrix multiplication must be done to transform each vertex, what becomes a precious time loss because of the high number of vertices. Other techniques, such as morphing i.e. simple interpolation of key faces, are not so time-consuming, but our experience in the field indicates that the results are not realistic enough. So, we decided to make use of GPU's power and we implement the skinning using shaders.

As stated before, a custom loader for Collada files was created to load all the additional information that the system needs and the manipulation of the bones is based on o3d[3]. The skinning using shaders and the co-articulation engine are integrated in this engine.

Regarding the communication between the client and the server of the system, a communication protocol was developed for the system integration. Simple Object Access Protocol (SOAP) based Web Services were developed in PHP to solve this communication. Both Web Services have client and server profiles since both have to send and receive information. The visual information module sends the text to the speech generation module and waits until the acoustic speech and the syllable based information for lip synchronization are ready.

[3]o3d. `http://code.google.com/p/o3d/`

As the facial animation system will be used for foreign language learning, it has been tested in the type of devices that will use it, mainly desktop computers and laptops. Table 1 shows the frame rate obtained in three different devices using Google Chrome web browser.

Device	Frames per Second
Desktop computer with high-end GC	∼80 FPS
Laptop with high-end GC	∼70 FPS
Laptop with commodity GPU	∼35 FPS

Table 1: *Frame rates obtained with different devices.*

The best performance was obtained with a desktop computer. The test was held in an Intel Core2 Duo CPU (2,20 GHz) with a NVIDIA GeForce 9800 GT GPU and the average frame rate was about 80 frames per second. Besides, the application was run in two different laptops. The first one was a Intel Core Duo CPU (2,00 GHz) with a NVIDIA GeForce 8600M GT GPU and the second one was a Genuine Intel CPU (1,30 GHz) with its commodity GPU. The frame rate obtained by the laptop with the most powerful GPU doubles the other's frame rate (∼70 FPS vs. ∼35 FPS).

This shows that the key of a fast performance is the GPU. The utilization of shaders for the skinning makes the developed application not perfectly suitable for computers that do not have a powerful graphic card. Nevertheless, as the laptop we used for the tests was not actual and the performance was acceptable, we consider that the developed tool is useful in the majority of working laptops and desktop computers.

7 Conclusion and Future Work

The results obtained so far in the project SPEEP are satisfactory. As mentioned in section 1 two key challenges were identified for the project and we consider we accomplished them.

On the one hand, a realistic Visual Speech Synthesis system has been presented. Based on the classic method by Cohen and Massaro [Cohen et al. 1993] a fast rule-based method was incorporated to handle syllables. Figure 5 shows four captures of the virtual face when talking. Since the project is in its first steps, the validation by real users has not been done yet, but the first exploration shows that a high level of realism has been achieved.

Nevertheless, it has been detected that some phonemes do not seem natural enough. So, at the moment of writing this paper, a new skeleton with few more bones in the lips are being designed to solve this problem. This way, more natural shapes will be obtained and there will not be any noticeable increase in computational cost, because of the low number of new bones.

On the other hand, the problems that slower performance of web applications can cause have been overcome. Several strategies have been presented to make the application work faster and we showed that they worked, specially for computers with good graphic cards.

However, as stated before, the project SPEEP is taking its first steps and still, there is much work to do. The idea is to keep improving the level of realism of the Visual Speech Synthesis. Indeed, even though the level of realism achieved so far is high for people that can listen to the audio file, it is far from the ideal level, where a deaf person will be able to read the lips of the virtual face.

For that, we consider that skeletal animation is a good choice, but it can be enhanced by more sophisticated muscle simulation techniques. Besides, newer co-articulation techniques can be applied.

Figure 5: *Four captures of the talking head.*

Moreover, even if it is not part of the project, we plan to test the application in mobile devices. The utilization of WebGL makes the system can be run on smartphones and tablets, but the efficiency becomes even more crucial. We believe that the increasing power of such devices and the corresponding optimization strategies will be enough to render a high-realistic virtual talking face.

References

ALEXA, M., BEHR, J., AND MÜLLER, W. 2000. The morph node. In *Proceedings of the fifth symposium on Virtual reality modeling language (Web3D-VRML)*, ACM, 29–34.

ARGHINENTI, A. 2011. Animation workflow in killzone3: A fast facial retargeting system for game characters. In *ACM SIGGRAPH 2011 Talks*, ACM, 37.

AVRIN, J. 2011. Character development and rigging for facial animation part 1. In *http://www.jonasavrin.com/2011/08/12/character-development-and-rigging-for-facial-animation-part-1/*.

BAUER, S., KNEIP, J., MLASKO, T., SCHMALE, B., VOLLMER, J., HUTTER, A., AND BEREKOVIC, M. 1999. The mpeg-4 multimedia coding standard: Algorithms, architectures and applications. *J. VLSI Signal Process. Syst. 23*, 1 (Oct.), 7–26.

BEELER, T., HAHN, F., BRADLEY, D., BICKEL, B., BEARDSLEY, P., GOTSMAN, C., SUMNER, R. W., AND GROSS, M. 2011. High-quality passive facial performance capture using anchor frames. In *ACM Transactions on Graphics (TOG)*, vol. 30, ACM, 75.

BENIN, A., LEONE, G. R., AND COSI, P. 2012. A 3d talking head for mobile devices based on unofficial ios webgl support. In *Proceedings of the 17th International Conference on 3D Web Technology*, ACM, New York, NY, USA, Web3D '12, 117–120.

COHEN, M. M., MASSARO, D. W., ET AL. 1993. Modeling coarticulation in synthetic visual speech. *Models and techniques in computer animation 92*.

DENG, Z., CHIANG, P.-Y., FOX, P., AND NEUMANN, U. 2006. Animating blendshape faces by cross-mapping motion capture data. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, ACM, 43–48.

DEROSE, T., KASS, M., AND TRUONG, T. 1998. Subdivision surfaces in character animation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM, 85–94.

EUSKALTZAINDIA. 1998. *Euskara Batuaren ahoskera zaindua*. Euskaltzaindia.

FEI, K. 2001. Expressive textures. In *Proceedings of the 1st international conference on Computer graphics, virtual reality and visualisation*, ACM, 137–141.

GACHERY, S., AND MAGNENAT-THALMANN, N. 2001. Designing mpeg-4 facial animation tables for web applications. *MIRALab, University of Geneva*.

GIBBON, D., MOORE, R., AND WINSKI, R. 1997. *Handbook of standards and resources for spoken language systems*. Walter de Gruyter.

HODGINS, J., JÖRG, S., O'SULLIVAN, C., PARK, S. I., AND MAHLER, M. 2010. The saliency of anomalies in animated human characters. *ACM Transactions on Applied Perception (TAP) 7*, 4, 22.

HUANG, H., CHAI, J., TONG, X., AND WU, H.-T. 2011. Leveraging motion capture and 3d scanning for high-fidelity facial performance acquisition. In *ACM Transactions on Graphics (TOG)*, vol. 30, ACM, 74.

KRŇOUL, Z. 2011. Web-based sign language synthesis and animation for on-line assistive technologies. In *The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility*, ACM, New York, NY, USA, ASSETS '11, 307–308.

LEUNG, C., AND SALGA, A. 2010. Enabling webgl. In *Proceedings of the 19th international conference on World wide web*, ACM, 1369–1370.

MASSARO, D. W., BESKOW, J., COHEN, M. M., FRY, C. L., AND RODGRIGUEZ, T. 1999. Picture my voice: Audio to visual speech synthesis using artificial neural networks. In *AVSP'99-International Conference on Auditory-Visual Speech Processing*.

MOORE, G. 2001. Talking heads. In *http://www.gamasutra.com/view/feature/3089/talking_heads_facial_animation_in_php*.

PARKE, F. I. 1972. Computer generated animation of faces. In *Proceedings of the ACM annual conference-Volume 1*, ACM, 451–457.

RADOVAN, M., AND PRETORIUS, L. 2006. Facial animation in a nutshell: past, present and future. In *Proceedings of the 2006 annual research conference of the South African institute of computer scientists and information technologists on IT research*

in developing countries, South African Institute for Computer Scientists and Information Technologists, 71–79.

SU, M.-C., AND LIU, I.-C. 2001. Application of the self-organizing feature map algorithm in facial image morphing. *Neural processing letters 14*, 1, 35–47.

VAN STRATEN, A. 2011. Character rigging — pipeline tools. In *http://andy-van-straten.com/*.

WANG, W. 2010. *Muscle-based Facial Animation*. PhD thesis, Texas A&M University.

YAMAMOTO, E., NAKAMURA, S., AND SHIKANO, K. 1998. Lip movement synthesis from speech based on hidden markov models. *Speech Communication 26*, 1, 105–115.