JAVA BASED TALKING FACES FOR ADVANCED WEB USER INTERFACES

Carlo Bonamico[1], *Roberto Pockaj*[1]
[1]DIST, University of Genova
Via Opera Pia 13, 16145 GENOVA, Italy, email: {charlieb,pok}@dist.unige.it

ABSTRACT

Parameter-based facial animation is now a mature technology, and has also been included in the MPEG4 standard. To make this technology available on the Web, the authors have developed a Java port of the FAE facial animation software created at DIST. While showing an acceptable performance reduction with respect to the original FAE, the Java version is considerably more flexible, since it can be used either as a stand-alone Java applet and as part of more complex, multi-user on-line worlds. This paper describes JFAE architecture, based on the Java3D object-oriented graphics library. The results of several tests that compare performances of the Java and native code versions are also presented.
JFAE allows for the use of virtual consultants, clerks or testimonials to create easy to use user interfaces for e-Commerce sites and Web call centers. As a sample, a prototype Web-based virtual news service is described.

1. KEYWORDS

Java multimedia applets, Java3D, facial animation, MPEG-4, Web user interfaces, Web call centers, e-commerce.

2. INTRODUCTION

With the exponential diffusion of electronic commerce on the Web, site developers are always looking for new types of interactive content, not only to distinguish their virtual shop from the competition, but also to create easy to use user interfaces, in order to attract a wider customer audience.
On the side of machine-to-man interaction, computer-animated characters, acting as virtual consultants, clerks or testimonial are obviously of great interest to online businesses.
Parameter-based facial animation is now a mature technology: the inclusion in the MPEG4 standard eases interoperability and integration with other multimedia content [13]. However, while high quality results have been obtained with stand-alone facial animation software, only a small number of very simple talking heads are presently available on the Web.

Starting from our experience in the European Community funded *VIDAS* and *InterFace* projects, we developed a Java port of the FAE facial animation software created at DIST's DSP-Lab. While showing an acceptable performance degradation with respect to the original FAE, the Java version is considerably more flexible. Not only it can be inserted as an applet in static web pages, but it may also be integrated in more complex, interactive web applications where several faces interact with each other and with the user. This would allow the creation of user friendly (and possibly fun) Web sites, especially targeting non-technical users and children.

This paper is organized as follows: section 2 presents a range of facial animation systems for the Web, and describes the animation algorithms used in the DIST's Facial Animation Engine. Section 3 analyzes the Java version of this engine. Section 4 describes animation and performance results. Section 5 suggests some possible applications of this technology, while describing the virtual news service that was realized as a proof-of-concept. Section 6 suggests some future developments of this technology.

3. STATE OF THE ART

Since the early work of Parke [12] and Waters [17], facial animation techniques have evolved significantly, while progressively gaining more relevance in the development of multimedia systems.
This has lead to the inclusion of detailed specifications for facial animation modules in the ISO/IEC MPEG-4 standard. While traditional video coding frameworks are based on the idea of a rectangular window of pixels, with an associated audio stream, MPEG-4 organizes multimedia data in *scene graphs* composed by individual objects. This allows the adoption of optimized coding techniques for each individual class of objects: natural video, text, high and low quality audio, 3D meshes, and so on.
The Synthetic Natural Hybrid Coding (SNHC) ad hoc group operates in MPEG since 1996, with the mandate of defining efficient techniques for the coding of synthetic objects. Its choice for coding synthetic faces was to define a standard set of parameters for animating and reshaping virtual heads.

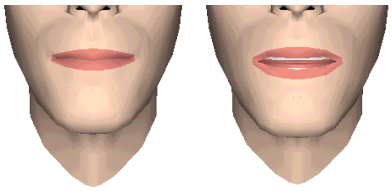


Figure 1. Effects of varying FAPs 5, 10, 11, 52, 57, 58.

68 Facial Animation Parameters (FAPs) are responsible of describing the movements of the face, both at low level (i.e. displacement of specific characteristic points of the face, like lip corners, as in Figure 1), and at high level (i.e. reproduction of a facial expression, like joy or anger).

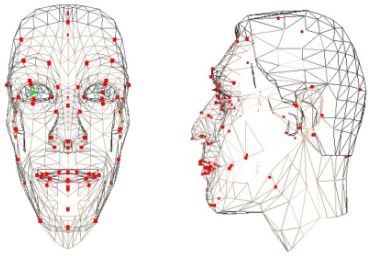


Figure 2. The 84 *feature points* defined by MPEG4.

Other parameters may be used to modify the geometry (through a set of feature points) and the appearance (through the mapping of texture images) of the face.

It should be noted that the standard defines only the displacement of 84 *feature points*, while leaving to developers of the decoding software the duty of moving any other point of the face model in order to create realistic animations (Figure 2).

As an example, we will shortly describe the algorithm used in the MPEG-4 face decoder developed at DIST. The Facial Animation Engine is an Open-GL based application, written entirely in C, which runs on Windows PCs and Silicon Graphics workstations.
The engine is divided in two blocks (Figure 3).

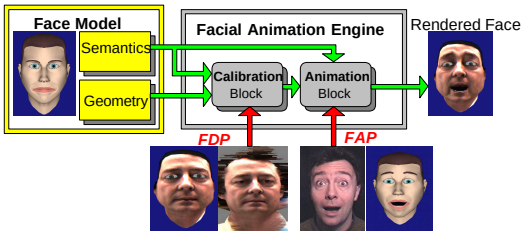


Figure 3. Structure of the Facial Animation Engine.

The animation block is able to perform the optional calibration phase, where a generic face model is reshaped to better represent a given real face. A second animation block, given a VRML 2 file containing a face model, an MPEG-4 parameter stream, and an audio stream, produces a realistic animation of the entire face.
The algorithms used in FAE are based on the concept of a semantic file, which contains model-specific information about the correlation between the displacement of the feature points and the displacement of adjacent points [8].

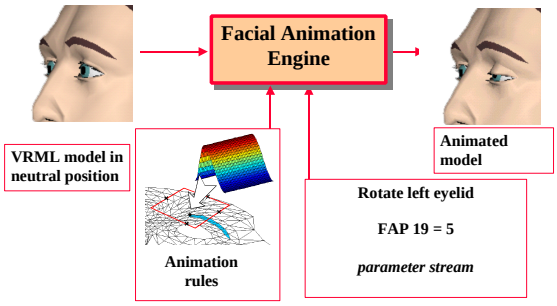


Figure 4. Example of application of FAE algorithms to the model Oscar.

To give an example, let's consider the 19th FAP ("close top left eyelid"), which describes the 1-D movement of feature point 3.1. From the semantic file, the animation block knows which vertex corresponds to the feature point, and has a list of all the vertices that must be moved, and the weight associated to each of them. The animation block also knows that the predefined movement associated to FAP 19 is the "weighted rotation along a line parallel to the X axis", and can therefore start rotating the vertices, each with an angle whose amplitude is function of the weight associated with it. Figure **4** shows the result of the described procedure.

The parameter streams can be produced either with motion capture techniques, using dedicated cameras, or with a text-to-speech engine, by deriving mouth movements from phonemes.

Many other universities and research laboratories have developed MPEG-4 compliant facial animation software. A quite complete list is present in [16]. However, while high quality results have been obtained with stand-alone facial animation software, only a small number of very simple talking heads are available on the Web.
Apart from sprite-based characters derived from Microsoft's Agent Development Kit [1], existing web-based facial animation systems can be divided into 3 categories, according to the underlying technologies:

- VRML+EAI Java applets (Tinky);
- Self contained Java applets (RedTed's JavaHead, W Interactive);

- ActiveX controls for Internet Explorer and plugins for Netscape (RedTed, NTU university's VRTalk).

Tinky is a face player implemented in JAVA by GMD-IPSI. It is an applet which can control a VRML model of a face displayed externally by a VRML plugin like SGI CosmoPlayer, using the External Authoring Interface (EAI). Tinky permits the visualization of various expressions, and the reproduction of MPEG4 streams. However, the face models used are really simple [6].

A pure Java solution is the one choosen by the developers of W Interactive [19]. The applet displays texture-mapped 3D models derived from real faces. Actually, 2D images are rendered off-line and stored on a server. The player can then download and reproduce a sequence of frames synchronized with an audio stream. Animation is based on the reproduction of MPEG-4 *visemes* (high level parameters which describe the mouth position corresponding to the emission of a given phoneme).

RedTed's JavaHead [14] uses texture mapping over very simple 3-D models, which can move mouth and eyes. It is not MPEG-4 compliant. RedTed also provides an ActiveX version of its software, for use with the Internet Explorer browser.

Internet Explorer is also required by the VRTalk Player developed by Dr. Chen at NTU university (Taiwan). It is speech-driven, in the sense that it obtains mouth movements from the analysis of an audio stream. It is not based on MPEG4 [2].

4. JFAE

The Java version of the Facial Animation Engine is based on the same animation algorithm as the native version, with minor changes in the audio/video synchronization mechanism due to the use of the `javax.sound.sampled` library instead of DirectSound.

The main obstacles to porting an animation engine (or any other multimedia software) to the web are the management of streams coming from remote servers, the peculiarities of browser plugin APIs, and the limited graphic APIs available to Java applets. This situation has changed with the introduction of the Java 2 platform, which offers good support for stream handling, complete browser integration, and high level graphics manipulation through the Java3D visualization library.

Java 3D is an *Application Programming Interface* for the creation, visualization and animation of three-dimensional graphics [15]. The API is specified by Sun, which offers also implementations for PC/Windows and Solaris. There are compliant implementations for SGI, HP-UX, MacOS and Linux. It offers a platform-independent high level interface to a native library (like OpenGL/Mesa or DirectX). The main advantages of using Java3D come from its object-oriented structure. A complex 3D scene may be described as a set of individual objects which are composed into a so called *scene graph* (Figure 5).

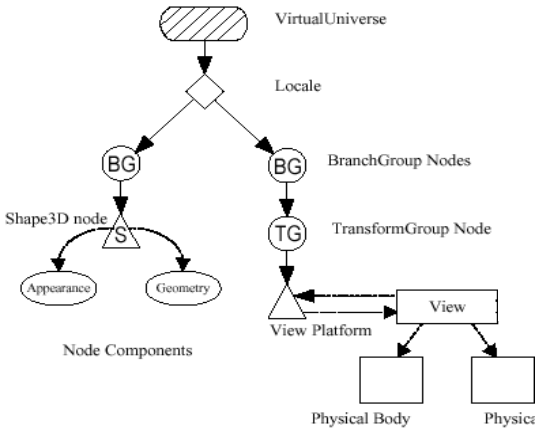


Figure 5. Structure of a generic Java3D *scene graph*, which is composed of both group and leaf nodes.

Graphic objects are either simple geometric primitives (cubes, triangles), or user-defined structures which are in turn defined as a subgraph of simpler objects. Transform nodes may be inserted in the scene graph to correctly locate shapes in 3D space. Each shape may be animated separately by an object of class `Behavior`.

Following this philosophy, we designed the Java face module as a sub-tree containing several objects, derived from Java3D's base classes `TransformGroup`, `Shape3D`, and `Behavior`.(Figure 6).

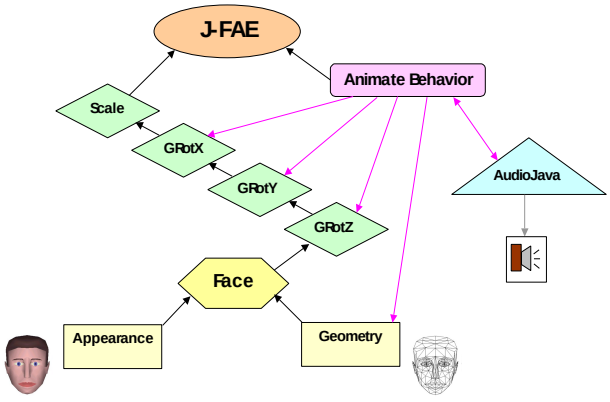


Figure 6. Structure of the JFAE sub-graph.

The `JavaFacialAnimationEngine` class represents the external interface of the module. It extends the `BranchGroup` class, and thus can be

inserted in any Java3D scene as easily as a single triangle. It has methods for selecting face models and streams to play, and to start and stop animation.

Three `TransformGroup` nodes handle the rotation of the head over the X,Y, and Z axes. They are directly controlled by the GROTX, GROTY and GROTZ parameters extracted from the FAP stream. A scaling `TransformGroup` allows the correct sizing of the virtual face, as the VRML coordinates may be expressed in different measurement units.

The `Face` class extends the `Shape3D` generic base class, and holds the description of the geometry of the face, as specified in a VRML file. Geometric information is stored in 4 data structures:

- `VertexData`, which contains the coordinates of all vertices, and, for each vertex, a list of the polygon to which it belongs to;
- `PolygonData`, which describes the triangles which define the face, and links each triangle with the information about its vertices contained in VertexData;
- `GroupData`, which divides the triangles into 24 groups corresponding to the various anatomic parts of the head (lips, eyes, hair, ad so on);
- `FeaturePointsData`, which tells which vertices of the model correspond to the key facial features defined by the standard (the corners of the mouth, the point of the nose, and so on).

The `Face` class also defines the appareance of the head, e.g. the set of parameters which controls its rendering: reflectiveness, shading model, and back faces culling.

A `FapStream` class parses the animation stream coming from a remote URL, and, at each frame, makes the animation parameters available to the animation engine via the `getFAPFrame()` method.

The `AnimateBehavior` class controls the animation of a `Face` object.
During the construction phase, it creates an `AudioJava` object (described later), and registers itself with the Java3D rendering engine, by specifying that it must be activated each time Java3D renders a frame. In fact, Java3D conceptually follows this simple cycle continuously:

1. Read inputs from keyboard, joystick, mouse, or other 3D input devices
2. Activate all registered behaviors
3. Render the current frame
4. Go to step 1

During step 2, Java3D calls the method `processStimulus()` of each `Behavior`. In this method `AnimateBehavior` performs synchronization with audio and computes the current position of the face.

Each time it is called, this method computes the number of the FAP frames to display from the number of audio samples already played by `AudioJava` (this is possible as both the audio sampling rate and the FAP frame rate are known).

To obtain realistic audio/video synchronization at a 25 fps video frame rate, the clock reference used must vary at least every 40ms. With Sun's JDK 1.3, it was not possible to extract synchronization information from the system clock through calls to `System.getCurrentMillis()`(which returns the numbero of milliseconds elapsed since midnight of 1st Jan. 1970) as the return values are update only every 60 ms.If it has already be displayed (this is possible with fast machines), the method returns immediately.

If the animation is late with respect to audio, a number of FAP frames are skipped.

`AnimateBehavior` performs then an interpolation phase, during which the position of the mouth due to the presence of a viseme parameter is mixed with the position defined by the low-level parameters. The same process is repeated for the high-level parameter expression. As an example, if the expression is joy, the face must smile, and if low-level parameters indicate to close eyes, the result is a smiling face with closed eyes.
Two other interpolations allow a considerable bandwidth reduction by exploiting symmetries of the head:

- Animation parameters relative to the right side of the face may be recreated from those relative to the left side (and vice versa)
- Animation parameters relative to the outer and inner part of the lips are also very strictly correlated

When all FAP values for the current frame have been correctly computed, the vertex coordinates of the head model are initialized with the values corresponding to the "neutral face" (eyes opened, mouth closed, face looking to the user, all muscles relaxed). Then the position corresponding to the current FAP frame is obtained by applying several base movements to the vertices surrounding each feature point (see Table 1 for examples).

Method	**Effect**
MoveWTRANSX	Weighted translation over the X axis
RotateYROTX	Rotation around the X axis with an angle computed from the projection of the feature point on the Y axis.
RotateGROTX	Rotation the entire model around the X axis

Table 1. Some examples of the base coordinates transformations used in JFAE.

These so-called animation rules are dynamically computed by the `SemanticData` class for each model. `SemanticData` loads from a model-dependent semantic file the relation between the displacement of feature points and the displacement of surrounding vertices. It computes the FAP units from the distances between some key feature points, like the distance between the eyes, the distance between the upper corner of the mouth and the nose, and so on. In fact the values of animation parameters defined by MPEG 4 are referred to these units, so that the same FAP stream can animate correctly different faces.
The reproduction of the associated audio stream is delegated to the `AudioJava` class. It uses the `javax.sound.sampled` library which is included in the JDK since version 1.3. The library supports the transparent reproduction of various file formats. While at present only *mu-law* compression is available, several third-party developers are working on other audio coding formats, including MP3 [7].

Advaced browser integration may be obtained through LiveConnect [11], a framework developed by Netscape to allow interaction between Javascript code in an HTML page and Java applets contained in the page.

The requirements for running the Java Facial Animation Engine as an applet are:

- Netscape 6 for seamless browser integration (because Netscape 6 includes the JRE 1.3 as its default Java Virtual Machine)
- Java 3D (any version)

However, it is possible to use older browsers if JDK or JRE, release 1.3, are installed on the client machine. Even JDK 1.2 is suitable if the Java sound library, available from Sun as part of Java Media Framework version 2.0, is installed separately.

5. RESULTS AND PERFORMANCE EVALUATION

The use of an high level graphics library helped us reducing applet size and thus download times. JFAE is contained in a 34 kB JAR file. A compressed model, with the related semantic file, adds another 30 to 100kB, according to the complexity of the model.

The animation parameter stream has very low bandwidth requirements: 9kbit/s for the ASCII format and 1.2 kbit/s for the binary format. Our prototype presently uses 8khz, 8 bit per sample, mu-law audio files, which require 64kbit/s. When integration of a Java MP3 decoder, like the freely available JavaLayer will be complete, the overall required bandwidth for audio and 25 fps animation will remain under 10kbit/s. In comparison, MPEG1 CIF video requires 50 kbits for a single frame.

JFAE showed good animation results with respect to rendering of facial expression (Figure 7).

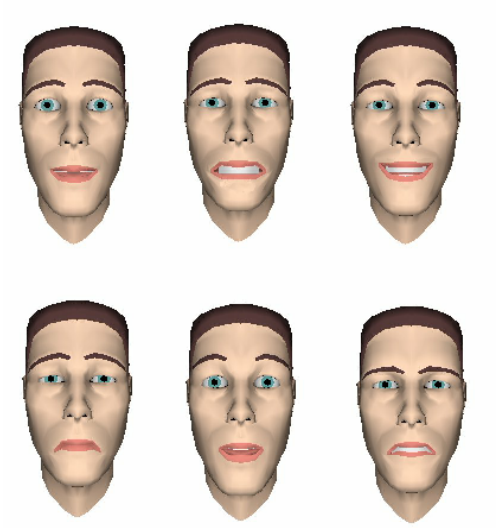


Figure 7. Model Oscar performs the six base expressions defined by MPEG4: *fear*, *disgust*, *happiness*, *sadness*, *surprise*, *anger*.

Using a Java3D VRML object loader from the Web3D consortium [18], we easily created complex scene including several objects and virtual faces.

Since performances of Java code are usually lower than those of native code, we conducted several experimental evaluations in order to quantify this gap in the case of 3D animation software.
Anyway, it must be noted that current Java 3D implementations are in turn based on OpenGL native code. In this hybrid architecture, while native code speeds up graphic rendering, an additional overhead is introduced by the data translation required to do method calls between native code and Java.

For all tests the software configuration was the following:

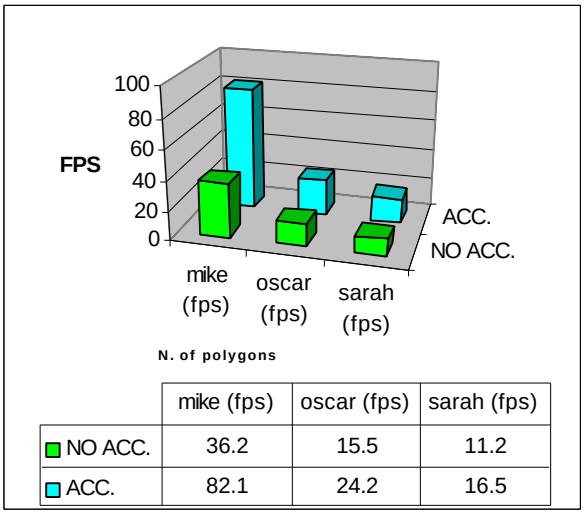
- Windows NT Workstation 4.0
- Sun JDK 1.3rc 2
- Java3D 1.2beta1 for OpenGL

The hardware configurations for the three test machines were:
P III 600: Intel Pentium III 600 MHz, 128 Mb of RAM, Asus V-6800 video card with a GeForce 256 graphic processor and 32 Mb of video RAM
P II 400: Intel Pentium II 400 MHz, 128 Mb of RAM, ASUS V3200 TNT video card with 16 Mb of video RAM
K6 II 350: AMD K6 II 350 MHz, 128 Mb of RAM, Matrox G400 DualHead video card with 32 MB of video RAM.

A first test was addressed at evaluating computational load in function of model complexity. For this test we

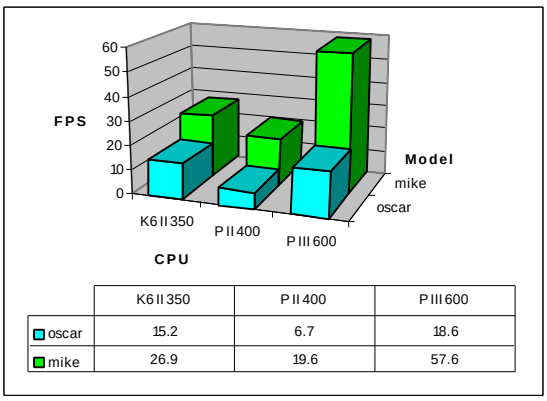
used the Pentium III 600 workstation, with a 200x200 visualization window, which correspond to the typical size of an image/video in a Web page. As Figure 8 shows, the activation of hardware acceleration in the video card greatly improved the obtained frame rate.

This test also confirmed experimental evidence collected by the MPEG 4 *Implementation Studies Group* [9], which showed that rendering time is usually a linear function of the number of polygons and of the number of pixels displayed.



	mike (fps)	oscar (fps)	sarah (fps)
NO ACC.	36.2	15.5	11.2
ACC.	82.1	24.2	16.5

Figure 8. Performance versus model complexity.



	K6II 350	PII 400	PIII 600
oscar	15.2	6.7	18.6
mike	26.9	19.6	57.6

Figure 9. Performance with different hardware configurations.

In a second test we compared the frame rate obtained on the 3 different hardware configurations with the two models Oscar and Mike. While the simpler model, Mike (abour 800 polygons) was animated at good frame rates even on the lower end configurations, we verifyied that more complex models require at least a Pentium III class machine (Figure 9). In some cases, the K6-III PC outperformed the more expensive Pentium II, thanks to the advanced hardware acceleration of 3D graphic rendering implemented by the Matrox video card.

A third test was conducted on the Pentium II PC to directly compare the rendering times of the Java and native versions of our animation engine(Figure 10).

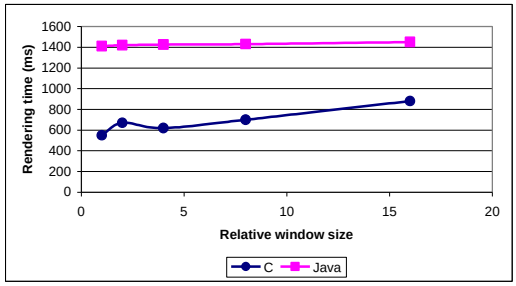


Figure 10. Comparison between Java and native code versions fo FAE.

The results showed that rendering times were at least 2-3 times higher in the Java version, while the gap was considerably reduced when displaying the animations in larger windows or full-screen.

A fourth test evaluated the performance degradation caused by the simultaneous animation of several models (which is not possible in the C version). While the frame rate was cut by an half each time a new model was added to the scene, JFAE successfully animated 3 copies of model Mike at 20 fps on a 900x600 pixel window on the Pentium III machine (Figure 11).

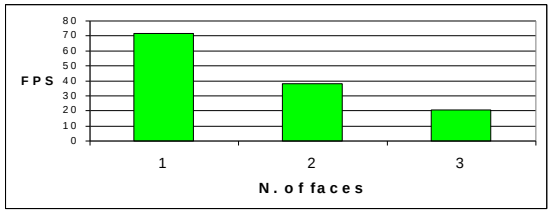


Figure 11. Performance degradation when animating multiple models simultaneously.

In conclusion, even if the Java version showed somewhat slower frame rates with respect to the native version of FAE, it proved to be usable on low end PCs if simpler face models are used. Overall, JFAE showed less degradation of performances when rendering the animations full-screen.

6. APPLICATIONS

Between the different possible applications of virtual characters we would like to cite four web-oriented applications.

A **virtual salesman**, hosted in catalog pages of e-commerce sites, could read product descriptions, and give advice to users. As JFAE can be easily inserted into more complex Java3D virtual scenes, it is possible to use it as the clerck of a 3D virtual shop [10].

Many sites now offer free, web-based email to millions of users. Their user-interface could be greatly enhanced by the presence of an animated **mail reader**, if JFAE is integrated with a text-to-speech engine.

An extension of traditional telephone call-centers, the **Web call-center** may use a virtual face as a common front-end to the user, while animating it either from a human or an automatic operator on the server side.

For the fourth application, the **virtual news service**, we have developed a simple prototype where our two models Oscar and Mike, act as virtual anchormen and comment some images regarding last-minute news (Figure 12).

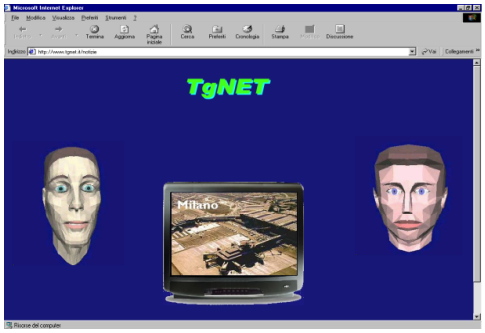


Figure 12. A virtual news service on the Web**.**

Presently, our laboratory is working with the italian deaf people association (*FIADDA*), to investigate how much the possibility to read the mouth movements of the virtual character helps hearing impaired people to better understand speech. As part of this research, we are conducting a user survey with both deaf people and their relatives to evaluate user reactions to this technology.

7. FUTURE WORK

In order to improve the flexibility of the Java Facial Animation Engine, we plan to implement in Java a calibration algorithm, so to be able to derive several different faces from a single VRML model [4]. Also the integration of a pure Java Mpeg 2 layer 3 audio decoder would improve audio quality over slow networks.

The use of the phoneme to FAP converter under development in our laboratory would let any SAPI compliant text-to-speech engine dynamically generate animation sequences from text. It will be possible to use our virtual face as a front-end to "intelligent" conversation engines, like Eliza[5] or CLIPS [3], which presently uses a textual interface.

On the long term, our research will be focused on the extension of animation to the entire body, as specified by version 2 of the MPEG 4 standard.

8. REFERENCES

[1] La Cantoche, http://www.cantoche.com

[2] I. Chen, *"A speech based facial animation decoder"*,http://www.cmlab.csie.ntu.edu.tw/~ichen/VRTalk_Demo.htm

[3] CLIPS project, http://www.ghgcorp.com/clips

[4] M. Costa, L. Ambrosini, F. Lavagetto, R. Pockaj, *"3D Head Model Calibration based on MPEG-4 Parameters"*, The 6th IEEE International Workshop on Intelligent Signal Processing and Communication Systems, Melbourne, Australia, 1998.

[5] Joseph Weizenbaum ,*"ELIZA--a Computer Program for the Study of Natural Language Communication Between Man and Machine,"* Communications of the Association for Computing Machinery 9 36-45, 1966.

[6] G. Fries, A. Paradiso, F. Nack, and K.Schuhmacher *"A Tool for Designing MPEG-4 compliant Expressions and Animations on VRML Cartoon-Faces"*, in Proceedings of AVSP Conference, August 7-9 1999.

[7] *"A pure Java Mpeg 2 Layer 3 decoder"*, http://javazoom.hypermart.net/javalayer/javalayer.html.

[8] F. Lavagetto, and R. Pockaj, *"The Facial Animation Engine: towards a high-level interface for the design of MPEG-4 compliant animated faces"*, IEEE Trans. on Circuits and Systems for Video Technology, Vol. 9, n.2, March 1999.

[9] G. Lafruit, L. Nachtergaele, A. Scherpenberg, T. Huybrechts, and J. Bormans, *"Computational Graceful Degradation Analysis in SNHC"*, ISO/IEC JTC1/SC29/WG11/MPEG98/M3567, July 99.

[10]A. Marriott, L. Ambrosini, and F. Lavagetto, *"Virtual Salesperson"*, In Proceedings of Australian. Workshop on AI in Electronic Commerce, Sydney, Australia, December 1999.

[11]*"Netscape Navigator and LiveConnect"*, http://home.netscape.com/navigator/v3.0/livecon nect.html.

[12]F. Parke, *"Parametrized Models for Facial Animation"*, IEEE Computer Graphics Applications, 2(9):61-68, November1982.

[13]Rob Koenen, *"MPEG-4 Multimedia for our time"*, IEEE Spectrum, february1999.

[14]RedTed, http://www.redted.mcmail.com.

[15]H. Sowizral, K. Rushforth, and M. Deering, *"The Java 3D API specification"*, Adison-Wesley, 1998.

[16]*Facial Animation sites list,* http://mambo.ucsc.edu/psl/fan.html.

[17]K. Waters, and F. Parke, *"Computer Facial Animation"*, A. K. Peters Ltd, 1996.

[18]Web3D Consortium, http://www.web3d.org.

[19]W Interactive, http://www.winteractive.fr.