Published in the proceedings of Computer Animation 2002.

Conversational Virtual Character for the Web

KARLO SMID[1], IGOR S. PANDZIC[2]

[1]Ericsson Nikola Tesla ETK
Krapinska 45, p.p. 93,
HR-10 002 Zagreb
karlo.smid@etk.ericsson.se

[2]Department of Electrical Engineering
Linköping University, SE-581 83 Linköping
igor@isy.liu.se

Abstract

Talking virtual characters are graphical simulations of real or imaginary persons capable of human-like behaviour, most importantly talking and gesturing. Coupled with artificial intelligence (AI) techniques, the virtual characters are expected to represent the ultimate abstraction of a human-computer interface, the one where the computer looks, talks and acts like a human. Such an interface would include audio/video analysis and synthesis techniques combined with AI, dialogue management and a vast knowledge base in order to be able to respond quasi-intelligently to the users - by speech, gesture and even mood. While this goal lies further on in the future, we present an architecture that reaches towards it, at the same time aiming for a possibility of practical applications in nearer future. Our architecture is aimed specifically at the Web. It involves a talking virtual character capable of involvement in a fairly meaningful conversation with the user who types in the input.

1. Introduction

Talking virtual characters are graphical simulations of real or imaginary persons capable of human-like behaviour, most importantly talking and gesturing. Coupled with artificial intelligence (AI) techniques, the virtual characters are expected to represent the ultimate abstraction of a human-computer interface, the one where the computer looks, talks and acts like a human.

In such a system, the AI technique must enable a talking virtual character to lead a natural language conversation with a user. Natural language conversation stands for the conversation that we (human beings) use when we talk to each other.

Many researchers have been involved in AI researches into natural language conversation [Weizenbaum66, Alice, Garner97, Hutchens98]. They have proposed different techniques and produced several natural language conversation systems. Every year they present their work by competing for the Loebner Prize [Loebner]. The Loebner Prize is the first formal instantiation of a Turing test [Turing50].

The AI system of our Virtual Character is A.L.I.C.E. [Alice]. It is not based on complicated AI mechanisms like neural network, knowledge representation, search, fuzzy logic [Garner97], genetic algorithms or parsing [Hutchens98]. The theory that is behind its AI is called "Case-Based Reasoning" or CBR. The first implementation of CBR was program ELIZA [Weizenbaum66]. A.L.I.C.E. won the Loebner Prize in 2000 and 2001.

The ultimate conversational human-computer interface would include audio/video analysis and synthesis techniques coupled with AI, dialogue management and a vast knowledge base in order to be able to respond quasi-intelligently to the user – by speech, gesture and even mood [Interface, Magnenat-Thalmann00]. While this goal lies further on in the future, we present an architecture that reaches towards it, at the same time aiming for a possibility of practical applications in nearer future. Our architecture is aimed specifically at the Web. It involves a talking virtual character capable of involvement in a fairly meaningful conversation with the user who types in the input.

Our system architecture represents a speech on demand system. An overview of the characteristics and design options for such systems is given in [Beard01]. According to [Beard01] our system architecture is a server bias implementation. We have chosen this approach because we wanted a lightweight client. The negative side of this approach is that we have a need for bandwidth and compression of generated bit stream files. Advantages are that all processing is done on the server side and the system upgrades the need to be done only on server side.

After presenting our architecture in the next section, we give more details about the AI system we used, and in particular about our Facial Animation Player, the component that delivers the animated character to the Web and the Facial Motion Cloning method for producing animatable face models automatically. What is proposed in the end is future work on the given architecture.

2. System Architecture

The design goal was to develop a Web-based Virtual Character capable of leading a natural language conversation with a user. The artificial intelligence of the Virtual Character is based on A.L.I.C.E. [Alice]. The animated Virtual Character is an extension of our previous work [Pandzic01a]. The main task was to develop optimal technical solutions for integrating those software tools into a single application.

Expected result was a publicly accessible Web page requiring no plug-ins and working at least in the current versions of Netscape and Internet Explorer. The page has to contain an animated Virtual Character and a text box. When the user types English natural text in the text box, Virtual Character replies by talking.

Solution of the design goal is the system architecture given in Figure 2. The system is a typical client-server architecture. Client and server sides have several modules. Client side consists of Facial Animation Player (FAPly) and Communication Handler (CH). On the server side there are A.L.I.C.E. natural language conversation server and Web server with Common Gateway Interface (CGI) programs Fbagen and Garbage.

Facial Animation Player (FAPly) is explained in section 4. It presents the animated Virtual Character on the Web page and animates it.

Communication Handler (CH) is at the heart of the given system architecture. It was programmed by using Java applets. It connects FAPly with A.L.I.C.E. The interface between CH and A.L.I.C.E. is HTTP protocol, more precisely its GET method. In that way A.L.I.C.E. keeps all its functionality (internal variables). The negative impact is that A.L.I.C.E. session is strictly coupled with client's IP address. That means that it is impossible to have multisession with A.L.I.C.E. from one computer. CH gets "logical" answers from A.L.I.C.E. and using Web server's CGI interface and Fbagen program dynamically produces talking and lip sync bit stream files for the FAPly. After FAPly plays those bit stream files, CH deletes them by using Web server's CGI interface and Garbage program.

A.L.I.C.E. server is a natural language conversation server. It gives "intelligence" to FAPly.

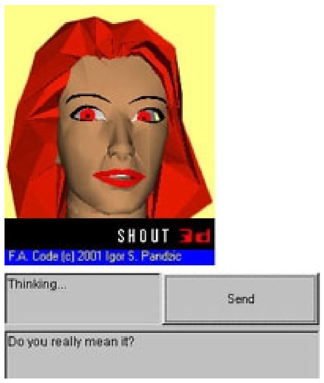
The Web Server holds FAPly and CH and provides a client with them on his request (Web browser). It is also used as a temporary repository for speech and lip sync bit stream files.

The **CGI Fbagen** program is used for dynamic generation of speech and lip sync bit stream files. Input to this program is an "intelligent" answer that CH gets from the A.L.I.C.E. server. It uses Text to speech TTS interface from Microsoft's SAPI engine for generating speech bit stream file, and it also uses the lip sync encoder that was presented in our previous work [Pandzic01a] for generating the lip sync file. The lip sync file is encoded as an MPEG-4 FBA [ISO14496] bit stream, using high-level viseme parameters for lip movement encoding and viseme blend with linear interpolation for coarticulation.

The **CGI Garbage** program deletes the speech and lips sync bit stream files played by FAPly.

This system architecture was programmed by using Java applets and Web server's CGI interface. CGI programming was done using C++. Microsoft's SAPI engine must be installed on the host that runs Web and A.L.I.C.E. servers.

Users type their input in the text box, and by pressing the ENTER key or SEND button confirm their input . The answer textbox is almost immediately filled with the A.L.I.C.E. answer (Figure 1). After some time, Demy will talk.



Thinking...

Do you really mean it?

Figure 1: Demy in action.

The steps that CH takes during one user input are illustrated in Figure 2.

During its initialisation, CH sets the session ID that is used as the name for the speech and lip-sync bit stream files. Session ID is client's connection time. This step is done only once per session. When the user types the input in the text box and presses SEND button or ENTER key, CH asks the A.L.I.C.E. server through this input, and A.L.I.C.E. answers (step 1. and 2.). Then CH sends this answer to the CGI Fbagen program. Fbagen notifies CH when generation of speech and lip-sync bit stream files is done (steps 3. and 4.). Now files are ready and CH triggers FAPly to play those files (step 5.). FAPly plays files (step 6.). The result is that Demy speaks. The cycle is finished when CH deletes speech and lips sync bit stream files using the Garbage program (step 7.).

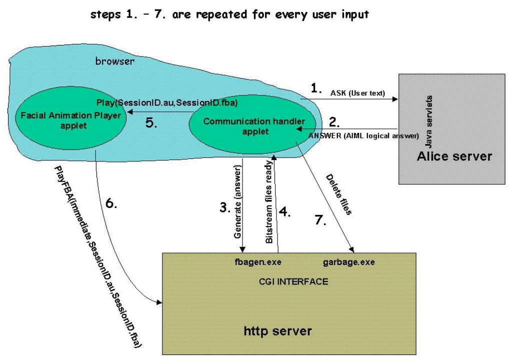

steps 1. - 7. are repeated for every user input
browser
Facial Animation Player applet
Communication handler applet
Play(SessionID.au,SessionID.fba)
5.
1.
ASK (User text)
2.
ANSWER (AIML logical answer)
Java servlets
Alice server
PlayFBA(Immediate,SessionID.au,SessionID.fba)
6.
3.
Generate (answer)
Bitstream files ready
4.
7.
Delete files
fbagen.exe
garbage.exe
CGI INTERFACE
http server

Figure 2: System architecture of the "Intelligent" Virtual Talking Character Demy

3. The Artificial Intelligence system

The artificial intelligence system in our architecture is based on the A.L.I.C.E. system [Alice]. A.L.I.C.E. (Artificial Linguistic Internet Computer Entity) is a natural language artificial intelligence chat robot. Dr. Richard S. Wallace is the author of A.L.I.C.E.

A.L.I.C.E. migrated to the platform-independent Java language in 1998. It was made open source under the GNU general public license, and more than 300 developers from around the world have contributed to the A.L.I.C.E. project.

The theory that describes the A.L.I.C.E. algorithm is called "Case-Based Reasoning" or CBR [Wallace00]. The CBR "cases" are the categories in AIML. The algorithm finds best-matching pattern for each input. Thus A.L.I.C.E. learns and thinks.

Artificial Intelligence Markup Language AIML [AIML] is based on Extensible Markup Language XML. Its purpose is to give chat robots (bots) intelligence and knowledge. It is a minimalist markup language.

4. The Facial Animation Player

The first choice we made when we were designing the player was to make it MPEG-4 FBA compatible [ISO14496, Escher98, Tekalp00]. This guarantees a variety of Facial Animation content sources, as many developers already support the standard and this trend is increasing. The choice of MPEG-4 also ensures very low bit rate needs. The MPEG-4 FBA decoding process itself is based on integer arithmetic, its implementation is very compact and it is very modest in CPU usage.

When the MPEG-4 Facial Animation Parameters (FAPs) are decoded, the player needs to apply them to a face model. Our choice for the facial animation method is interpolation from key positions, essentially the same as the morph target approach widely used in computer animation and the MPEG-4 Facial Animation Table (FAT) approach [ISO14496, Tekalp00]. Interpolation was probably the earliest approach to facial animation and it has been used extensively [Parke74, Parke82, Arai96]. We prefer this one to procedural approaches like [Magnenat-Thalmann88, Chadvick89, Kalra92, Escher98], and especially to the more complex muscle based models like [Platt81, Waters87, Terzopoulos90] for the following reasons:

- It is very simple to implement, and therefore easy to port to various platforms.
- It is modest in CPU time consumption
- The usage of key positions (morph targets) is close to the methodology used by computer animators and should be easily adopted by this community

The way it works is the following. Each FAP (both low- and high-level) is defined as a key position of the face, or *morph target*. To be consistent with the computer animation terminology, we will use the term "morph target" throughout this article. Each morph target is described by the relative movement of each vertex with respect to its position in the neutral face, as well as the relative rotation and translation of each transform node in the scene graph of the face. The morph target is defined for a particular value of the FAP. The movements of vertices and transforms for other values of the FAP are then interpolated from the neutral face and the morph target. This can easily be extended to include several morph targets for each FAP and use a piecewise linear interpolation function, like the FAT approach defines. However, current implementations show simple linear interpolation to be sufficient in all situations encountered so far. The vertex and transform movements of the low-level FAPs are added together to produce final facial animation frames. In case of high-level FAPs, the movements are blended by averaging, rather than added together.

4.1. Implementation

Due to its simplicity and low requirements, the Facial Animation Player is expected to be easy to implement on a variety of platforms using various programming languages. The implementation we describe here is written as a Java applet and based on the Shout3D rendering engine [Shout3D].

We have successfully tested this implementation of the MPEG-4 Facial Animation Player with several face models as illustrated in Figure 3. We can achieve interactive frame rates with models of up to 3000 polygons. The player has correctly interpreted the test FBA bit streams (Marco, Wow, Emotions) as well as the bit streams produced by the text-to-speech system. As the demonstration web page [Pandzic01a] shows, the applet is fully controllable from the web page by JavaScript, making all interactions possible. The implementation has been tested and works robustly in the two major Web browsers.

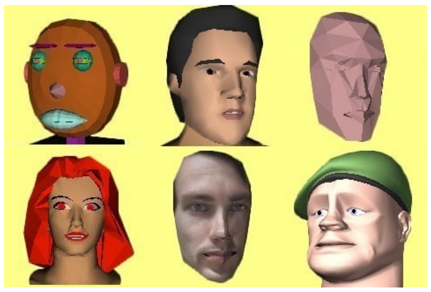


Figure 3. Examples of face models experimentally animated using the player: dummy, a model built using 3D modelling software; Miraface, a model donated by MIRALab, University of Geneva, to ISO as MPEG-4 reference software; Candide, source Linköping University; Demy, designed by Sasa Galic; Jörgen, source Linköping University; Commander Lake, source 3DS Max.

4.2. Producing Animatable Face Models

In this section we describe our approach to the production of face models that can be directly animated by the Facial Animation Player described in the previous section.

We believe that the most important requirement for achieving high visual quality is the openness of the system for visual artists. It should be convenient for them to design face models with the tools they are used to. While numerous algorithmic facial animation systems have been developed, the best-looking animations in current productions are done manually by artists or by facial tracking equipment and performing talent. This manual creation is painstakingly time-consuming, but some aspects can be automated.

The concept of morph targets as key building blocks of facial animation is already widely used in the animation community. However, morph targets are commonly used only for high level expressions (visemes, emotional expressions). In our approach we follow the MPEG-4 FAT concept and use morph targets not only for the high level expressions, but also for low-level MPEG-4 FAPs. Once their morph targets are defined, the face is capable of full animation by limitless combinations of low-level FAPs.

Obviously, creating morph targets not only for high level expressions, but also for low-level FAPs is a tedious task. We therefore propose a method to copy the complete range of morph targets, both low- and high-level, from one face to another. This means that an artist could produce one very detailed face with all morph targets, then use it to quickly produce the full set of morph targets for a new face. The automatically produced morph targets can still be edited to achieve a final detail. It is conceivable that libraries of facial models with morph targets suitable for copying to new face models will be available commercially. The method we propose for copying the morph targets is called Facial Motion Cloning. Our method is similar in goal to the Expression Cloning [Noh01]. However, our method additionally preserves the MPEG-4 compatibility of a cloned facial motion and it treats transforms for eyes, teeth and tongue. It is also substantially different in implementation.

Facial Motion Cloning can be schematically represented by Figure 4. Inputs to the method are the source and target face. The source face is available in neutral position (*source face*) as well as in a position containing some motion we want to copy (*animated source face*). The target face exists only as neutral (*target face*). The goal is to obtain the target face with the motion copied from the source face – the *animated target face*.

To reach this goal we first obtain *facial motion* as the difference of 3D vertex positions between the animated source face and the neutral source face. The facial motion is then added to the vertex positions of the target face, resulting in the animated target face.

In order for this to work, the facial motion must be normalised, which ensures that the scale of the motion is correct. In the *normalised facial space*, we compute facial motion by subtracting vertex positions of the animated and the neutral face. To map the facial motion correctly from one face to another, the faces need to be aligned with respect to the facial features. This is done in the *alignment space*. Once the faces have been aligned, we use interpolation to obtain facial motion vectors for vertices of the target face. The obtained facial motion vectors are applied by adding them to vertex positions, which is possible because we are working in the normalised facial space. Finally, the target face is denormalized.

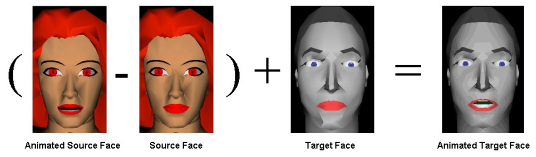


Figure 4: Overview of Facial Motion Cloning

5. Performance results

5.1. Test environment

We have successfully implemented and tested the system and it is currently available for public trial at http://lancelot.isy.liu.se.

Server configuration is a very important issue for the system architecture. Table 1 shows the server configuration used in the current public trial.

Kernel version	Microsoft Windows NT, Uniprocessor Free
Product type	Workstation
Product version	4.0
Service pack	6
Processors	1
Processor speed	450 MHz
Processor type	x86 Family 6 Model 5 Stepping 2, GenuineIntel
Physical memory	256 MB
IDE standard (both on disk and motherboard side	ATA33
Network adapter	Ethernet 100Mb

Table 1: Server configuration

Physical memory is a critical resource in server configuration. 256 MB is minimum amount for the normal prototype functionality. Choosing a right test environment is very important for producing real test results. Internet must be an integral part of it. Figure 5 represents the test environment. Lancelot is a server side in our system architecture and pci022 is a client side. Table 2 represents performance results in the given test environment.

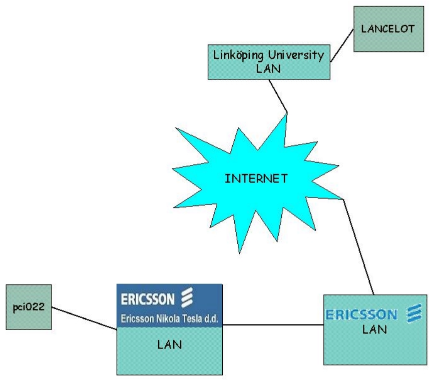


Figure 5: Test Environment

Number of words in answer	Answer size/B	Audio bit stream size/kb	Generate time/s	Time to answer/s	Actual length of the answer/s	Bandwidth kb/s
2	4	8	4	9.1	1	1.57
5	23	12.9	4	6.6	1.4	4.96
22	141	63.7	11	13.6	7.9	24.50
5	25	13.4	4	7.9	1.3	3.44
7	38	19.2	5	7	2.4	9.60
111	749	406	54	73	52	21.37

Table 2: Performance results

5.2. Test execution

For measuring Generate time (GT) performance parameter, a test program based on the CGI Fbagen program was created. The input to the test program were CGI environment variables. Measurements were performed "off line". "Off line" means that CGI environment variables were prepared in Microsoft Windows batch file and a test program was started using that batch file on a local computer. The local computer had to have the same configuration as the Lancelot. One of CGI environment variables was Alice's generated answer. The output from the test program was Generate time (as a duration between the start and end of the test program) and speech bit stream file.

Time to answer was measured during the connection to the public trial web page. The test requirement was to produce the answers used in measuring GT performance parameter.

5.3. Comments on performance results

Time to answer (TTA) is the most important performance parameter. It represents duration between the moment when a user sends its query to the system (by pressing SEND button or ENTER key) and the moment when system responds by talking. TTA performance parameter consists of Generate time (GT) parameter and time in which the generated answer travels through the network from the system to the user. In the Table 2 we can see that most of the TTA parameter is GT performance parameter. So, performance bottleneck is not the Internet, but the process that generates real-time answers.

GT parameter also has two components. The first component is time that Alice system needs to give a text answer. That time is not the problem because Alice is very fast. The problem is the second component. Alice's text answer is the input into CGI Fbagen program. The output from the Fbagen is data which animates Facial Animation Player. That data is much heavier than Alice's text answer and that is the reason why Fbagen is much slower than Alice.

Bandwidth is calculated as $\frac{TTA - GT}{Audiobitstreamsize}$ and we can see that it is not a constant entry. TTA for the short answers (up to 30 words) in an implemented prototype is up to 10 seconds. This time is not acceptable for the practical implementation, but for this version of prototype it is a good result.

The prototype was tested against ten simultaneous users, but in the Ericsson Nikola Tesla LAN. Users didn't notice any problems in particular workload.

Demy was also presented on Faculty of Electrical Engineering and Computing (FER) as a part of a lecture at the department of Telecommunications. Demy was installed on FER LAN and answers were faster than the ones on the Testing Environment. Students didn't mention the slow answers in their comments. The facts that Demy was animated, rather intelligent and that could talk were more important for them.

6. Conclusions and future work

We have implemented a system architecture prototype that reaches towards the goal of a fully interactive, human-like, seemingly intelligent conversational virtual character. Existing software tools have been used. In the given prototype Facial Animation Player is a wrapper around A.L.I.C.E. and represents A.L.I.C.E. answers in more "human" way. Although the implemented prototype is not the final solution of the design goal, it is very close to it.

Future work can go in several directions: user interface, database of AIML files, Virtual Character speech ability, generating believable non-verbal communication and synchronising it with speech and technical improvements in order to bring the system from prototype to production level.

6.1. User interface

Improvements in user interface should be reconsidered. In the current prototype, user interface is a textbox in which a user types his input. In the next prototype speech recognition interface should be implemented as an option.

6.2. Virtual Character's speech ability

Facial Animation Player uses a relatively simple solution for Virtual Character's speech ability. Those are speech bit stream files generated with Microsoft's SAPI engine and played from the server's disk. Compression and streaming of those files is needed for a faster response. Putting emotions into Demy's talk will be considered.

6.3. Believable non-verbal communication

Facial Animation Player can also play Virtual Character's facial gestures (anger, surprise, etc.). The system that is available for public trial plays some gestures (head noise and blinks), but those gestures are not dynamically driven by the system answers. They are just periodically played to make Demy more "human". How to dynamically generate information for those gesture animations driven by AIML should be reconsidered.

6.4. AIML database

Database of AIML files represents Virtual Character's intelligence and knowledge. Some databases for practical purpose should be created (for example database about Demy and its creators with explanation of Demy's system architecture).

6.5. Prototype technical improvements

Speech generation is implemented in the current application, using Microsoft's SAPI engine which is based on COM objects. Those COM objects are used in C++ CGI program. Common Gateway Interface is an "old" technology that is supported by Web servers. But, since we have already been using Microsoft's COM technology, IIS (Microsoft's Internet Information Server) should be reconsidered as a Web server of the system architecture. IIS provides direct connections with COM objects thus avoiding CGI interface. Another recognized problem is that this prototype doesn't work with Netscape Navigator. Netscape Navigator has problems with Java. The prototype uses standard Java and the problem is in Netscape's Java implementation. The prototype works fine with Internet Explorer.

The prototype with given technical improvements and enhanced talking ability can be used in several practical applications: Frequently Asked Questions (FAQ), weather forecasting, news casting, a virtual salesman, a virtual administrator, etc.

The system architecture can be easily adopted for the mobile platforms. The server side looks the same. The only requirement is that the mobile client supports Java and that he has fast connection towards the Internet (General Packet Radio Service GPRS, Universal Mobile Telecommunications System UMTS).

7. Acknowledgements

This research is partly supported by the VISIT program of the Swedish Foundation for Strategic Research (SSF) and by Ericsson Nikola Tesla (ETK).

8. References

[Ahlberg01] "Using the Active Appearance Algorithm for Face and Facial Feature Tracking " J. Ahlberg, 2nd International Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-time Systems (RATFFG-RTS), pp. 68 - 72, Vancouver, Canada, July 2001.
[Alice], Artificial Linguistic Internet Computer Entity, http://www.alicebot.org
[AIML], Artificial Intelligence Markup Language, http://alicebot.org/alice/aiml.html
[Arai96] "Bilinear interpolation for facial expressions and metamorphosis in real-time animation", Kiyoshi Arai, Tsuneya Kurihara, Ken-ichi Anjyo, The Visual Computer, 12:105-116, 1996.
[Beard01], "Usable TTS for Internet Speech on Demand", Simon Beard, John Stallo, Don Reid, OzCHI Conference 2001
[Cohen93] M.M.Cohen and D.W.Massaro, "Modeling Coarticulation in Synthetic Visual Speech." In M.Thalmann & D.Thalmann (Eds.) Computer Animation'93. Tokyo: Springer-Verlag.
[Cossato98] Cosatto E., Graf H.P., "Sample-Based Synthesis of Photo-Realistic Talking Heads", Proc. Computer Animation '98, Philadelphia, USA, pp. 103-110.
[Chadvick89] "Layered construction for deformable animated characters", Computer Graphics, 23(3): 234-243,1989
[Eisert97] P. Eisert, S. Chaudhuri and B. Girod, "Speech Driven Synthesis of Talking Head Sequences," 3D Image Analysis and Synthesis, pp. 51-56, Erlangen, November 1997.
[Escher98] "Facial Deformations for MPEG-4", M. Escher, I.S. Pandzic, N. Magnenat-Thalmann, Computer Animation 98, Philadelphia, USA, pp. 138-145, IEEE Computer Society Press, 1998.
[Forchheimer83] "Low Bit-rate Coding through Animation", Robert Forchheimer and Olov Fahlander, Proceedings Picture Coding Symposium 83
[Forchheimer84] "A Semantic Approach to the Transmission of Face Images ", Robert Forchheimer, Olov Fahlander and Torbjörn Kronander, Proceedings Picture Coding Symposium 84
[Garner97], Luigi Caputo, Robby Garner, Paco Xander Nathan, "FRED, Milton, and Barry: Evolution of Intelligent Agents on the Web", AMSE-ISIS 1997, Reggio Calabria, Italy.
[Hutchens98], Jason Hutchens, Michael D. Alder, "Introducing MegaHAL", proceedings of the Human-Computer Communication Workshop, pp. 271-274, 1998.
[Interface] The InterFace project, IST-1999-10036, www.ist-interface.org
[ISO14496] ISO/IEC 14496 - MPEG-4 International Standard, Moving Picture Experts Group, www.cselt.it/mpeg

[Kalra92] Kalra P., Mangili A., Magnenat-Thalmann N., Thalmann D., Simulation of Facial Muscle Actions based on Rational Free Form Deformation", Proceedings Eurographics 92, pp. 65-69

[Magnenat-Thalmann88] "Abstract muscle actions procedures for human face animation", N. Magnenat-Thalmann, N.E. Primeau, D. Thalmann, Visual Computer, 3(5): 290-297, 1988.

[Magnenat-Thalmann00] Nadia Magnenat-Thalmann, Sumedha Kshirsagar, "Communicating with Autonomous Virtual Humans", Proceedings of the Seventeenth TWENTE Workshop on Language Technolgy , Enschede, Universiteit Twente, October 2000, pp 1-8.

[Noh01] "Expression Cloning", Jun-yong Noh, Ulrich Neumann, Proceedings of SIGGRAPH 2001, Los Angeles, USA

[Pandzic99] "Synthetic Faces: What are they good for?" Igor S. Pandzic, Joern Ostermann, David Millen, The Visual Computer, 1999.

[Pandzic00] "From Photographs to Interactive Virtual Characters on the Web", Igor S. Pandzic, Gael Sannier, Proc. Scanning 2000, Paris, France

[Pandzic01] "Life on the Web", Igor S. Pandzic, Software Focus Journal, 2(2): 52-59, John Wiley & Sons, 2001.

[Pandzic01a] "A Web-Based MPEG-4 Facial Animation System", I.S. Pandzic, Proc. ICAV 3D 2001, demonstration at www.icg.isy.liu.se/~igor/MpegWeb

[Parke74] "A Parametric Model for Human Faces", F.I. parke, PhD Thesis, University of Utah, Salt Lake City, USA, 1974. UTEC-CSc-75-047

[Parke82] "Parametrized models for facial animation", F.I. Parke, IEEE Computer Graphics and Applications, 2(9): 61-68, November 1982.

[Parke96] "Computer Facial Animation", F.I. Parke, K. Waters, A K Peters Ltd. 1996, ISBN 1-56881-014-8

[Pearson95] "Development in Model-Based Video Coding", Proc. of the IEEE, 83(6): 892-906, June 1995.

[Platt81] "Animating Facial Expressions", S.M. Platt, N.I. BadlerComputer Graphics, 15(3): 245-252, 1981.

[Quartz] Quartz Version 6.0, Symbian Technical Paper, Symbian Developer Network, www.symbiandevnet.com/techlib/techcomms/techpapers/papers/v6/over/quartz/index.html

[Shout3D] Shout 3D, Eyematic Interfaces Incorporated, http://www.shout3d.com/

[VRML] VRML, ISO/IEC 14772-1:1999, www.web3d.org/fs_specifications.htm

[Tekalp00] "Face and 2-D Mesh Animation in MPEG-4", Tekalp M.A., Ostermann J., Image Communication Journal, Tutorial Issue on MPEG-4 Standard, Elsevier, 2000.

[Terzopoulos90] "Physically-based facial modeling, analysis and animation", D. Terzopoulos, K. Waters, Journal of Visualization and Computer Animation, 1(4): 73-80, 1990.

[Turing50], A.M.Turing, "Computing Machinery and Intelligence", MIND the Journal of the Mind Association, vol. LIX, no. 236, pp. 433-60, 1950

[Wallace00], Richard.S.Wallace, "Don't Read Me: A. L. I. C. E. and AIML Documentation", http://alicebot.org/articles/wallace/dont.html, 2000.

[Weizenbaum66] Weizenbaum, J., "ELIZA - A computer program for the study of natural language communication between man and machine", Communications of the ACM 9(1): 36-45, 1966.

[Loebner], http://www.loebner.net/Prizef/loebner-prize.html

[Waters87] "A muscle model for animating three-dimensional facial expressions", K. Waters, Computer Graphics (SIGGRAPH'87), 21(4): 17-24, 1987.

[WIN] W Interactive SARL, www.winteractive.fr