# LiveMail: Personalized Avatars for Mobile Entertainment

Miran Mosmondor
*Ericsson Nikola Tesla, Krapinska 45, p.p. 93, HR-10 002 Zagreb*
*miran.mosmondor@ericsson.com*

Tomislav Kosutic
*KATE-KOM, Drvinje 109, HR 10 000 Zagreb*
*tomislav.kosutic@kate-kom.com*

Igor S. Pandzic
*Faculty of electrical engineering and computing, Zagreb University, Unska 3, HR-10 000 Zagreb*
*igor.pandzic@fer.hr*

## Abstract

LiveMail is a prototype system that allows mobile subscribers to communicate using personalized 3D face models created from images taken by their phone cameras. The user takes a snapshot of someone's face - a friend, famous person, themselves, even a pet - using the mobile phone's camera. After a quick manipulation on the phone, a 3D model of that face is created and can be animated simply by typing in some text. Speech and appropriate animation of the face are created automatically by speech synthesis. Animations can be sent to others as real 3D animated messages or as short videos in MMS. They can be used as fun messages, greeting cards etc. The system is based on a client/server communication model. The clients are mobile devices or web clients so messages can be created, sent and received on the mobile phone or on a web page. The client has a user interface that allows the user to input a facial image and place a simple mask on it to mark the main features. The client then sends this data to a server that builds a personalized face model. The client also provides an interface that lets the user request the creation of animated messages using speech synthesis. It is planned to have several versions of the client: Symbian, midlet-based, web-based, wap-based, etc. The server is responsible for sending messages adjusted to the capabilities of the receiving platform. The server, Symbian client, midlet-based client and the web client have been implemented as prototypes. We present the system architecture and the experience gained building LiveMail.

## 1. Introduction

Mobility and the Internet are the two most dynamic forces in communications technology today. In parallel with the fast worldwide growth of mobile subscriptions, the fixed Internet and its service offerings have grown at a rate far exceeding all expectations. The number of people connected to the Internet is continuing to increase and GPRS and WCDMA mobile networks are enabling connectivity virtually everywhere and at any time with any device. With advances in computer and networking technologies comes the challenge of offering new multimedia applications and end user services in heterogeneous environments for both developers and service providers.

The goal of the project was to explore the potential of existing face animation technology [11] for innovative and attractive services for the mobile market, exploiting in particular the advantages of technologies like MMS and GPRS. The new service will allow customers to take pictures of people using the mobile phone camera and obtain a personalized 3D face model of the person in the picture through a simple manipulation on the mobile phone. In this paper we present architecture of LiveMail system. We describe how unique personalized virtual characters are created with our face adaptation procedure. Also, we describe clients that are implemented on different platforms, most interestingly on mobile platform, since 3D graphics on mobile platforms is still in its early stages. Various different network and face animation techniques were connected into one complex system and we presented the main performance issues of such system. Also, the system uses a MPEG-4 FBA standard that could ultimately enable video communication at extremely low

bandwidths, and work presented in this paper could bring us one step further in that direction.

The paper is organized as follows. In the next section we give a brief introduction on used standards and technologies. Next, we present overall system architecture continuing with more details on server and client implementation in the following sections. Finally, system performance evaluation is given in the section 6.

## 2. Background

### 2.1. 3D modelling

Creating animated human faces using computer graphics techniques has been a popular research topic the last few decades [1], and such synthetic faces, or virtual humans, have recently reached a broader public through movies, computer games, and the world wide web. Current and future uses include a range of applications, such as human-computer interfaces, avatars, video communication, and virtual guides, salesmen, actors, and newsreaders [12].

There are various techniques that produce personalized 3D face models. One of them is to use 3D modeling tool such as 3D Studio Max or Maya. However, manual construction of 3D models using such tools is often expensive, time-consuming and it sometimes doesn't result with desirable model. Other way is to use specialized 3D scanners. In this way face models can be produced with very high quality using them in a mobile environment is not practical. Also, there have been methods that included use of two cameras placed at certain angle and algorithms for picture processing to create 3D model [8]. Some other methods, like in [9], use three perspective images taken from a different angles to adjust deformable contours on a generic head model.

Our approach in creating animatable personalized face models is based on face model adaptation of existing generic face model, similar to [14]. However, in order to achieve simplicity on a camera-equipped mobile device, our adaptation method uses a single picture as an input.

### 2.2. Face animation

Created personalized face model can be animated using speech synthesis [10] or audio analysis (lip synchronization)[13]. Our face animation system is based on the MPEG-4 standard on Face and Body Animation (FBA) [5][2]. This standard specifies a set of Facial Animation Parameters (FAPs) used to control the animation of a face model. The FAPs are based on the study of minimal facial actions and are closely related to muscle actions. They represent a complete set of basic facial actions, and therefore allow the representation of most natural facial expressions. The lips are particularly well defined and it is possible to precisely define the inner and outer lip contour. Exaggerated values permit to define actions that are normally not possible for humans, but could be desirable for cartoon-like characters.

All the parameters involving translational movement are expressed in terms of the Facial Animation Parameter Units (FAPU) (Figure 1. ). These units are defined in order to allow interpretation of the FAPs on any facial model in a consistent way, producing reasonable results in terms of expression and speech pronunciation. They correspond to fractions of distances between some key facial features (e.g. eye distance). The fractional units used are chosen to allow enough precision.

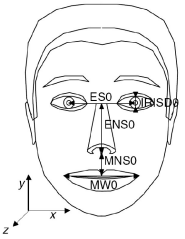


Figure 1. A face model in its neutral state and defined FAP units (FAPU) (ISO/IEC IS 14496-2 Visual, 1999)

The FAP set contains two high level FAPs for selecting facial expressions and visemes, and 66 low level FAPs. The low level FAPs are expressed as movement of feature points in the face, and MPEG-4 defines 84 such points (Figure 2. ). The feature points not affected by FAPs are used to control the static shape of the face. The viseme parameter allows rendering visemes on the face without having to express them in terms of other parameters or to enhance the result of other parameters, insuring the correct rendering of visemes. A viseme is a visual correlate to a phoneme. It specifies shape of a mouth and tongue for minimal sound unit of speech. In MPEG-4 there are 14 static visemes that are clearly distinguished and included in the standard set. For example, phonemes /p/, /b/ and /m/ represent one type of viseme. Important thing for their visualization is that the shape of the mouth of a speaking human is not only influenced by the current phoneme, but also by the previous and the next phoneme. In MPEG-4, transitions from one viseme to the next are

defined by blending only two visemes with a weighting factor. The expression parameter allows definition of high-level facial expressions.

The FAPs can be efficiently compressed and included in a Face and Body Animation (FBA) bitstream for low bitrate storage or transmission. An FBA bitstream can be decoded and interpreted by any MPEG-4 compliant face animation system [3][4], and a synthetic, animated face be visualized.

### 2.3. 3D graphics on mobile devices

Important aspect of our face animation system is 3D graphics on mobile phones. The last few years have seen dramatic improvements in how much computation and communication power can be packed into such a small device. Despite the big improvements, the mobile terminals are still clearly less capable than desktop computers in many ways. They run at a lower speed, the displays are smaller in size and have a lower resolution, there is less memory for running the programs and for storing them, and the battery only last for a short time.

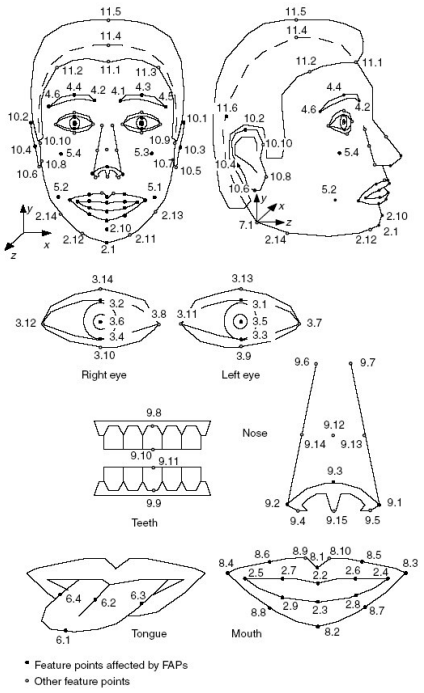


Figure 2. Face feature points (ISO/IEC IS 14496-2 Visual, 1999)

Rendering 3D graphics on handheld devices is still a very complex task, because of the vast computational power required to achieve a usable performance. With the introduction of color displays and more powerful processors, mobile phones are becoming capable of rendering 3D graphics at interactive frame rates. First attempts to implement 3D graphics accelerators on mobile phones have already been made. Mitsubishi Electric Corp. announced their first 3D graphics LSI core for mobile phones called Z3D in March 2003. Also other manufacturers like Fuetrek, Sanshin Electric, Imagination Technologies and ATI published their 3D hardware solution for mobile devices a few months after.

Beside hardware solutions, other important thing for 3D graphics on mobile devices is availability of open-standard, well-performing programming interfaces (APIs) that are supported by handset manufacturers, operators and developers alike. These are OpenGL ES (OpenGL for Embedded Systems) and Java Mobile 3D graphics (also known as JSR 184) that have emerged in last several months.

## 3. System architecture

The basic functions of the LiveMail service are simple creation of personalized face model, transmission and display of such virtual character on various platforms. The creation of such personalized face model consists of recognition of characteristic face lines on the taken picture and adjustment of generic face model to those face lines. The most important face lines are size and position of face, eyes, nose and mouth. Speech animation of the character is created from the input text. More details on the each module implementation are given in next chapters while this chapter describes system architecture.

Personalization of virtual characters and creation of animated messages with speech and lips synchronization is a time-consuming process that requires more computational power then the phone provides. Our system architecture is defined by this limitation. Capabilities of mobile devices have improved in last few years, but they are still clearly not capable of such demanding computing. Thus, our system is not implemented on one platform, rather is divided in several independent modules. The system is based on a client/server communication model. The basic task of the server is to perform computationally expensive processes like personalization of virtual character and creation of animated messages with speech and lips synchronization. The client side is responsible for displaying the animated message of the personalized virtual character and handling user requests for the creation of a new personalized virtual character and its animated message through proper user interface (Figure 3. ). In this way LiveMail clients are implemented on

various platforms: Symbian-based mobile devices, mobile devices with Java support (Java 2 Micro Edition), WAP and Web interfaces.

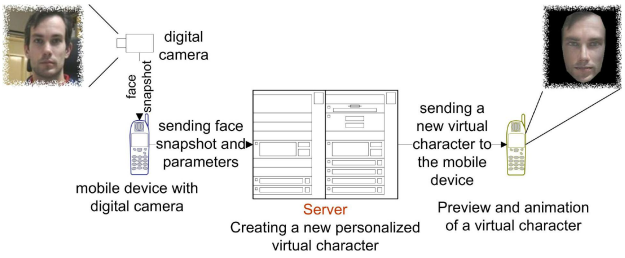


Figure 3. Simplified use-case scenario

The client application consists of a user interface through which users can create new personalized virtual characters and send animated messages, preview created messages and view received animated messages. When creating the personalized character the interface allows the user to input a facial image and place a simple mask on it to mark the main features. After selecting main face features the client sends this data to the server that builds a personalized face model (Figure 4. ).

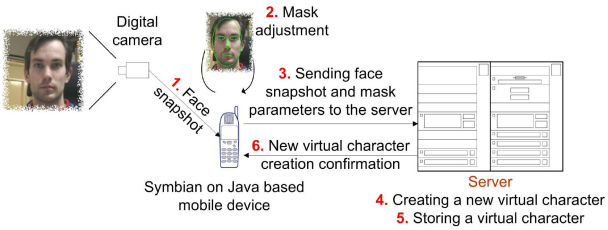


Figure 4. Personalized 3D face model creation

When creating animated messages, the user selects a virtual character, inputs text and addresses the receiver. The client application then sends a request for creation of the animated message to the server, which then synthesizes the speech and creates matching facial animation using the text-to-speech framework. The

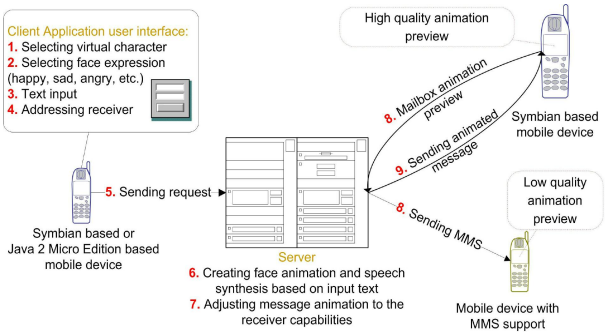


Figure 5. Creating animation message through client user interface

animated message is then adjusted to the capabilities of the receiving platform. For mobile devices that cannot display even medium quality 3D animation, animated message it is converted to a short video or animated GIF and sent by MMS (Figure 5. ).

## 4. The server

The server is a combination of a light HTTP server and an application server (Figure 6. ). The HTTP server provides clients with user interface and receives requests, while the application server processes client requests: creates new personalized 3D face models and animation messages. The user interface is dynamically generated using XSL transformations from an XML database each time client makes a request. The database holds information about user accounts, their 3D face models and contents of animated messages. The LiveMail server is a multithreaded application and the light HTTP server can simultaneously receive many client requests and pass them to the application server for processing. The application server consists of many modules assigned for specific tasks like: 3D face model personalization, animation message creation and more. During the process of 3D face model personalization and animated message creation there are resources that cannot be run in multithread environment. Therefore they need to be shared among modules. Microsoft's Text to speech engine, which is used for speech synthesis and as a base of 3D model animation, is a sample of a shared resource. To use such a resource all application server modules need to be synchronized.

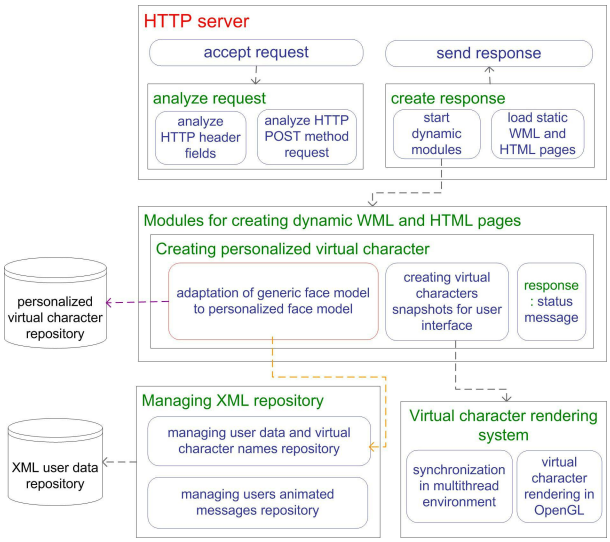


Figure 6. Simplified server architecture

The server's primary task is virtual character adaptation. The client's request for new virtual character creation holds entry parameters for adaptation process: the picture of person whose 3D model is created and the characteristic facial points in that picture (creating face mask). The server also has a generic 3D face model that is used in adaptation. Based on these inputs, the server deforms and textures the generic model in such a way that it becomes similar to the face in the picture, and thus produces the new 3D model ready for animation. The model is stored on the server for later use in VRML format.

The virtual character adaptation algorithm starts by mapping characteristic facial points from a facial picture to characteristic points in corresponding generic 3D face model. This initial mapping is followed by three main stages of adaptation:

The first stage is normalization. The highest and the lowest point of generic 3D face model are modulated according to the characteristic facial points from the facial picture. We translate the generic 3D face model to highest point from facial picture (point 11.4 on Figure 2. ). The size of the translated model does not match the size of the facial picture mask so we have to scale it. We calculate vertical ratio of generic model and facial picture mask and move every point of the generic model in corresponding ratio. We distinguish vertical and horizontal scaling. In horizontal scaling we look at the horizontal distance between every point to highest point from facial picture, relative to face axis symmetry. Vertical scaling is easier, as there is no axis symmetry.

The second stage is processing. We distinguish texture processing and model processing. With N existing points we create a net of triangles that cover all normalized space. It's important to notice that beyond the face the net must be uniform. Based on known points and known triangles the interpolator algorithm is able to determine the coordinates of any new point in that space using interpolation in barrycentric coordinates. The interpolator used here is described in [15]. We forward characteristic points received from client's request to an interpolator. The interpolator (based on triangles net) will determine location of other points that we need to create new model.

The third stage of the algorithm is renormalization. It is practically the same algorithm as in first segment, except we roll back model from normalized space back to space where it was before starting of algorithm.

## 5. The client

The animation itself is created on the server that provides users with transparent access, meaning various client types could be used. After personalized face model with proper animation is created, it is send back to the client, where it can be previewed. Multi-platform delivery, and the capability to implement support for virtually any platform is one of the contributions of this system. Our strategy is to use a bare-minimum face animation player core. This core can be easily ported to any platform that supports 3D graphics.

The face animation player is essentially an MPEG-4 FBA decoder. When the MPEG-4 Face 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 Face Animation Tables (FAT) approach [2]. FATs define how a model is spatially deformed as a function of the amplitude of the FAPs. Interpolation was the earliest approach to facial animation and it has been used extensively. We prefer it to procedural approaches and the more complex muscle-based models because it is very simple to implement, and therefore easy to port to various platforms; it is modest in CPU time consumption; and the usage of key positions (morph targets) is close to the methodology used by computer animators and could be easily adopted.

The way the player works is the following. Each FAP (both low- and high-level) is defined as a key position of the face, or morph target. Each morph target is described by the relative position 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 position 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. 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.

Due to its simplicity and low requirements, the face animation player is easy to implement on a variety of

platforms using various programming languages. Additionally, for the clients that are not powerful enough to render 3D animations, the animations can be pre-rendered on the server and sent to the clients as MMS messages containing short videos or animated GIF images. In next chapters, we describe the following implementations of the client: the Symbian client, Java applet-based web client, and a generic mobile phone client built around J2ME, MMS and WAP. The first two implementations are full 3D clients, while the last one only supports pre-rendered messages.

## 5.1. Symbian client

Our mobile client is implemented on Symbian platform as a standalone C++ application. After taking a photo with camera, the user needs to adjust the mask with key face part outlined (Figure 7. ). The mask is used to define 26 feature points on the face that are then, together with picture sent to the server for face adaptation, as described previously.


Figure 7. Symbian user interface and mask adjustment

After creation of personalized face model, it is sent back to the user where it can be previewed (Figure 8. ). The face animation player on Symbian platform for mobile devices is based on DieselEngine. The DieselEngine is collection of C++ libraries that helps building applications with 3D content on various devices. DieselEngine has low-level API (Application Program Interface) that is similar to Microsoft DirectX and high level modules had to be implemented. The most important is a VRML parser that is used to convert 3D animatable face model from VRML format to Diesel3D scene format (DSC). Other modules enable interaction with face model like navigation, picking and centering.

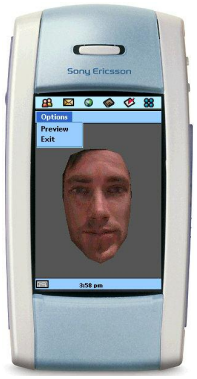


Figure 8. 3D face model preview on Symbian platform

## 5.2. Java applet-based web client

A parallel web service is also offered as an interface to the system that allows users to access all the functionality: creating new personalized virtual characters and composing messages that can be sent as e-mail. The system keeps the database of all the created characters and sent messages for a specific user identified by e-mail and password.

As another way of creating new personalized virtual characters, the Java applet is used with the same functional interface described earlier. Compared to mobile input modules, the Java Applet provides more precision in defining feature points since the adjustment of the mask is handled in higher resolution of the desktop computer. Also the cost of data transfer, which is significantly cheaper compared to mobile devices, makes it possible to use higher resolution portrait pictures in making better looking characters.

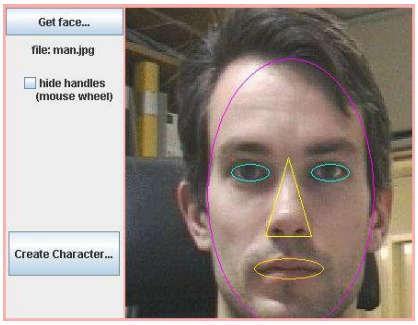


Figure 9. Mask adjustment on Java applet-based web client

New messages can be made with any previously created characters. The generated animation is stored on the server and the URL to the LiveMail is sent to the specified e-mail address. The html page URL is produced dynamically. The player used to view LiveMails is a Java applet based on the Shout3D

rendering engine. Since it is a pure Java implementation and requires no plug-ins, the LiveMail can be viewed on any computer that has Java Virtual Machine installed.

### 5.3. Generic mobile client built around J2ME, MMS and WAP

The face animation player is easy to implement on a variety of platforms using various programming languages. However, for the clients that are not powerful enough to render 3D animations we an offer alternative: the animations can be pre-rendered on the server and sent to the clients as MMS messages containing short videos or animated GIF images.

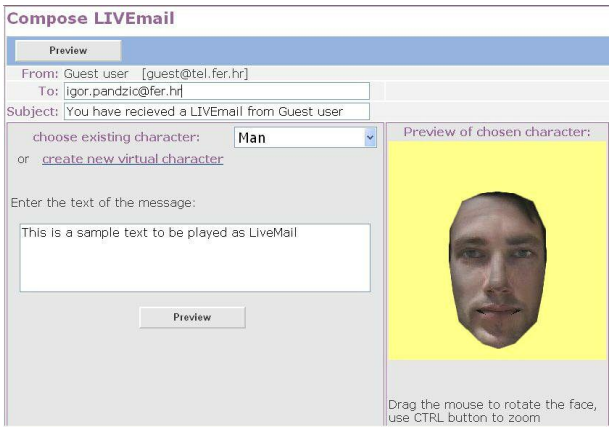


Figure 10. Composing LiveMail message on web client

A LiveMail WAP service is also provided for mobile phones. It provides the functionality of creating and sending LiveMail with previously generated personalized characters.

User interface through which personalized face model is created is also implemented on Java 2 Micro Edition (J2ME) platform. However, this platform alone does not define access to native multimedia services like, for example, camera manipulation and picture access. So, additional J2ME package, called Mobile Media API (MMAPI), was used.

## 6. System performance evaluation

In this section system performance evaluation is presented. First of all, the size of create new personalized virtual character request greatly depends on size of the face image. Other request data (user name, virtual character name, face mask parameters) is always approximately 1K. With average picture of 120x160 pixels request size is 15K. Request to create animated message carries few data and is always smaller than 1K.

### 6.1. Server

The described adaptation algorithm is very time consuming. Our measurements show that 98% of entire process is spent on the adaptation algorithm. The remaining 2% is spent on model storage and preview images for client user interface. With a generic 3D face model constructed of approx. 200 polygons adaptation algorithm takes 7.16 seconds on an AMD Athlon XP 2800+ (Figure 11. ). Each adaptation algorithm started by client's request is run in a separate thread so multiprocessor systems can handle simultaneous client requests much faster.

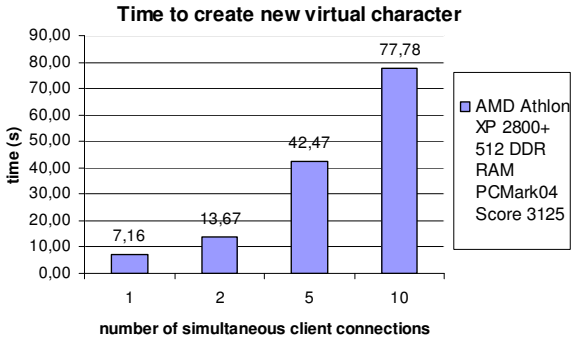


Figure 11. Time to create a new virtual character in respect to number of simultaneous client connections

The second most valuable server's task is creation of animated messages. Its execution time depends on message text length (Figure 12. ). Also, Microsoft Text to Speech engine cannot process simultaneous text to speech conversions so all client requests are handled one at the time (while all others wait).

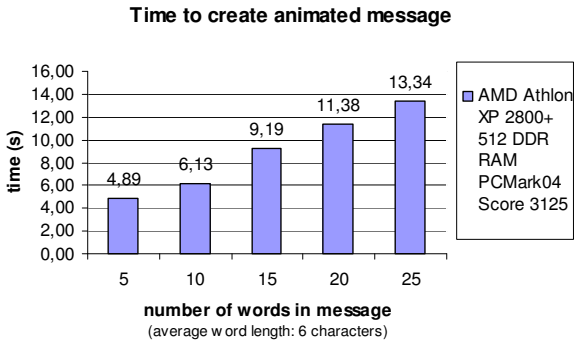


Figure 12. Time to create animated message in respect to number of words in message

### 6.2. Client

On the other side, required bandwidth for displaying animated message on Symbian client is approx 0.3 kbit/s

for face animation, and that is considerably less than sending raw video in any format. The reason for this is that FBA bitstream is built upon model based coding as was described previously. If we take analogy with voice, it uses the same principle like in GSM codec; it sends only parameters that are used on decoding side by a model to reproduce a sound, or in our case a face animation.

In addition to face animation, other bandwidth requirements for displaying animated message on Symbian client are 13 kbit/s for speech and approximately 50K download for an average face model. The Web client requires additional 150K download for the applet.

The Symbian client implementation was tested on Sony Ericsson P800 mobile device with various static face models. Interactive frame rates were achieved with models containing up to several hundreds polygons. The generic face model used in our LiveMail system uses approximately two hundred polygons and its performance is shown in Figure 13. After animation has started (in this case in 21st second), frame rate drops to average of 10 frames per second (FPS), but this is still relatively high above the considered bottom boundary for interactive frame rates on mobile devices.

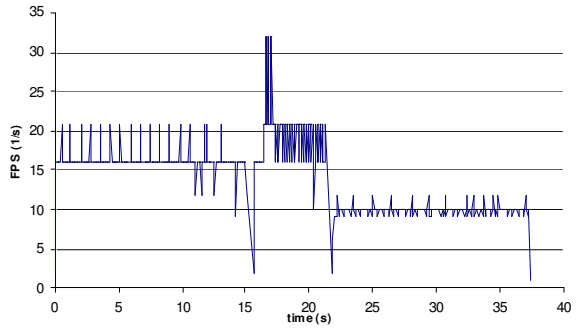


Figure 13. Generic face model performances on SE P800 mobile device

Web client showed performance of 24-60 fps with textured and non-textured face models of up to 3700 polygons on a PIII/1000. This performance is satisfactory for today's mobile PC user connecting to the Internet with, for example, GPRS. More details on this implementation and performance can be found in [11].

## 7. Conclusions

In this paper we have introduced a system that can be used as a pure entertainment application. Users deliver personalized, attractive content using simple manipulations on their phones. They create their own, fully personalized content and send it to other people. By engaging in a creative process - taking a picture, producing a 3D face from it, composing the message - the users have more fun, and the ways they use the application are only limited by their imagination.

LiveMail is expected to appeal to younger customer base and to promote services like GPRS and MMS. It is expected to directly boost revenues from these services by increasing their usage. Due to highly visual and innovative nature of the application, there is a considerable marketing potential. The 3D faces can be delivered throughout various marketing channels, including the web and TV, and used for branding purposes.

Besides entertainment aspects, there has been a great deal of research work. Barriers were crossed with the face animation player on mobile platform and with 3D face model personalization on the server. We have connected various different network technologies and face animation techniques in one complex system and presented the experiences gained building such a system. Also, the system uses a MPEG-4 FBA standard that could ultimately enable video communication at extremely low bandwidths. Although emphasis was on the entertainment, work presented in this paper could bring us one step closer to that goal.

## 8. Acknowledgments

We would like to acknowledge Visage Technologies AB, Linköping, Sweden, for providing the underlying face animation technology used for the system described in this paper.

## References

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

[2] Igor S. Pandzic, Robert Forschheimer (editors), *MPEG-4 Facial Animation - The standard, implementations and applications*, John Wiley & Sons, 2002, ISBN 0-470-84465-5.

[3] M. Escher, I. S. Pandzic, N. Magnenat-Thalmann, *Facial Deformations for MPEG-4*, Proc. Computer Animation 98, Philadelphia, USA, pp. 138-145, IEEE Computer Society Press, 1998.

[4] F. Lavagetto, 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, No. 2, March 1999.

[5] ISO/IEC 14496 - MPEG-4 International Standard, Moving Picture Experts Group, http://www.chiariglione.org/mpeg/standards/mpeg-4/mpeg-4.htm

[6] 3D Arts, DieselEngine SDK, http://www.3darts.fi/mobile/de.htm

[7] T. Fuchs, J. Haber, H.-P. Seidel, *MIMIC - A Language for Specifying Facial Animations*, Proceedings of WSCG 2004, 2-6 Feb 2004, pp. 71-78.

[8] Z. Liu, Z. Zhang, C. Jacobs, M. Cohen, *Rapid Modeling of Animated Faces From Video*, In Proceedings of The Third International Conference on Visual Computing (Visual 2000), pp 58-67, September 2000, Mexico City

[9] H. Gupta, A. Roy-Chowdhury, R. Chellappa, *Contour based 3D Face Modeling From A Monocular Video*, British Machine Vision Conference, 2004.

[10] Pelachaud, C., Badler, N., and Steedman, M., *Generating Facial Expressions for Speech*, Cognitive, Science, 20(1), pp.1–46, 1996.

[11] Igor S. Pandzic, Jörgen Ahlberg, Mariusz Wzorek, Piotr Rudol, Miran Mosmondor, *Faces Everywhere: Towards Ubiquitous Production and Delivery of Face Animation*, Proceedings of the 2nd International Conference on Mobile and Ubiquitous Multimedia, Norrkoping, Sweden, 2003

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

[13] P. Hong, Z. Wen, T. S. Huang, *Real-time speech driven Face Animation*, in I. S. Pandzic, R. Forchheimer, Editors, *MPEG-4 Facial Animation - The Standard, Implementation and Applications*, John Wiley & Sons Ltd, 2002.

[14] W.Lee, N.Magnenat-Thalmann, *Fast Head Modeling for Animation*, Journal Image and Vision Computing, Volume 18, Number 4, pp.355-364, Elsevier, March, 2000

[15] Igor S. Pandzic, *Facial Motion Cloning*, Graphical Models journal.