Integration of an animated talking face model in a portable device for multimodal speech synthesis

Master of Science in Speech Technology

Teodor Gjermani

gjermani@kth.se

Supervisor: Jonas Beskow

Examinator: Björn Granström

Abstract

Talking heads applications is not a new thing for the scientific society. In the contrary; they are used in a wide range of applications reaching web applications and the game industry as well as applications designed for special groups like hearing impaired people. The purpose of the current study, carried out at the Department of Speech, Music and Hearing, CSC, KTH, is to investigate how the MPEG based virtual talking head platform developed in the department could be integrated into a portable device to facilitate the human computer interaction for target groups, such as hearing impaired or elder people. A research evaluation on the quality of the multimodal speech synthesis produced by the model is also conducted.

Sammanfattning

Applikationer med virtuella talande huvuden är inte något nytt påfund. Tvärtom, de förekommer i allt från webbapplikationer och datorspel till applikationer för speciella grupper såsom hörselskadade. Avsikten med detta arbete, utfört på Tal, musik och hörsel vid skolan för datavetenskap och kommunikation, KTH, är att undersöka hur den plattform för MPEG-baserad animering av virtuella talande huvuden, som är utvecklad vid avdelningen, kan integreras i en portabel enhet för att underlätta människa-datorinteraktion exempelvis för hörselskadade eller äldre människor. En utvärdering av kvaliteten i den multimodala talsyntes som produceras av modellen har också genomförts.

Acknowledgments

I would like to give special thanks to a number of people, without their help and support this work could never be successfully completed. First, my supervisor, Prof. Jonas Beskow, who gave me a lot of advices and precious help during the whole project, coming up with good ideas and valuable suggestions on how to improve the work or to deal with many difficulties encountered during the project. Also his feedback and corrections were important for having the best result on this report. I would also like to thank my examiner Prof. Björn Granström for his advices and thoughts but also my opponent David Lucas Escribano for his precise observations.

Furthermore I would like to thank people from the department of Speech, Music and Hearing for their help and kindness, as they always were there for me when ever they were asked. These were Alexander Seward, Joakim Gustafsson, Jens Edlund and all the other people of the department and especially the ones that participated in the usability tests.

Table of Contents

A. List of Figures

B. List of Tables

1. Introduction

Dialogue systems that make use of animated talking heads have the advantage of acoustically and visually simulating natural human speech, by carrying both audio and visual information. This is very important for the interactivity of a dialog system, especially for target groups, such as hearing impaired or elder people, but also non-native speakers of the language. Research has showed that this part of the population give a very big importance to the visual part of the speech in order to understand what has been said [1] [2]. The movement of the lips and the general articulation of the speaker while producing speech can improve the disambiguation that can occur from the understanding of ambiguous audio signals. Canonical mouth shapes that accompany speech utterances have been categorized, and are known as visual phonemes or "visemes".[3] They are being used in the talking heads applications to simulate the movement of the human face when speech is produced.[4]

However the usage of speech dialog applications is now limited on static personal computer systems that feature fixed screens directly connected to stationary servers (tower), which provides all the necessary processing power. This is not very convenient since the user has to be in front of the screen and does not have the ability to move while he is interacting with the system. This also introduces limitations on the types of applications that can be implemented using speech dialogs and avatars. Thus the need of integrating dialog systems that also use talking head functionality into portable devices is now arising providing the users with a more flexible approach. This way the user could carry around this piece of equipment in a wide range of his everyday activities, including office and home environment. Such a possibility would automatically introduce a potential intelligent agent helping the user to identify and get access to the resources he will need next or in the near future. Using this information, the agent may download files, reserve communications bandwidth, post reminders, or automatically send updates to colleagues to help “smooth" the user's daily interactions. To use a metaphor, such an agent would act as a butler/confidant, who always stands behind the user's shoulder; he is always ready to help him and tries to streamline interactions with the rest of the world; all this with the use of natural language. Although this might sound as a future system, there is some serious work being done in all the fields we mentioned at this very time we speak [5]. A picture describing this interaction is given below.

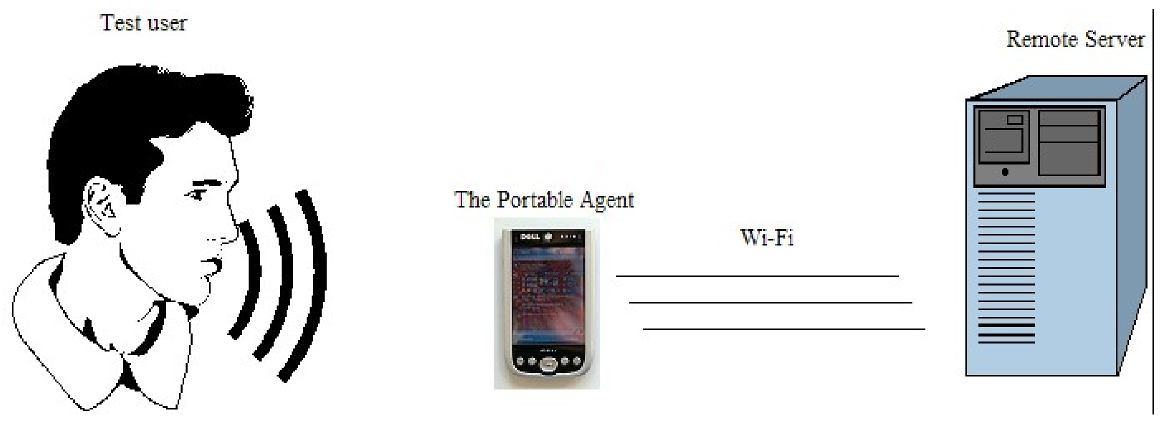

Figure 1. The portable agent

In this master thesis work the multimodal speech synthesis functionality on portable devices is discussed. The aim of this master thesis is to propose the introduction of a portable speech based system, with talking heads functionality. The main purpose is to overcome the limitations of integrating the MPEG-4 based virtual talking head model, which has been already implemented for the desktop computer to a version that will also work on a portable device; namely in our case a Dell Axim v51 PDA running on Windows Pocket PC mobile 5.0 operating system.

In this effort there are several issues to be treated. The first issue considers developing the OpenGL version for the talking head running on the portable device. The second issue is to connect the portable device to the server that is calculating the facial deformation parameters for rendering the talking head component and running the Automatic Speech Recognition (ASR) engine and the dialog system. And last but not least, the coming output audio stream should be synchronized with the rendered talking head.

1.1 Research Question

Talking heads applications are not something new. All previous applications thorough were though restricted in desktop computers and static consoles. In this study we are taking the talking heads from the desktop to the portable light device and try to examine how this influences interaction with the dialogue system. Furthermore we go and make further comparisons between the static and the portable version of the talking head component. Considering that this is a rather pioneer effort that is not implemented yet on any other project of the department of speech music and hearing at KTH, several questions arise and need to be answered

The main question is:

- How does the integration of the talking head platform in to a portable device facilitates the interaction of users with a dialog system?

In the course of work we found some new questions that would need an appropriate answer.

- Can this portable talking head component be used for other applications as well?
- In what occasions this portable version could be more preferable than the standard desktop version?

1.2 Method

Earlier research on talking head systems has been going on for quite a long time. At the department for Speech, Music and Hearing at KTH the an animation platform born under the EU-project pf-star is being used up until now for systems based on desktop computers

[7]. This animation platform is based on Java programming language, OpenGL, and the MPEG-4 facial animation standard launched by the MPEG-4 group [8]. As it is explained later on we made use of a previous model, also developed in the department that was using C++ as a programming language, as a basis for the development of the portable version of the talking head animation engine. Both of them make use of sophisticated animation- and rendering techniques and can produce high-quality facial animations [9]. The real breakthrough though, is the possibility to create a facial animation from any arbitrary, static, three-dimensional face model. This means that, provided the animation platform is provided with the information necessary, anyone of the existing, three-dimensional face models can be used for facial animation and speech synthesis with this platform.[10]

1.3 Goal of the research

The aim of the project was to develop a talking face application for multimodal speech synthesis on a portable device. For this purpose the existing C++ developed engine model of the talking head application was used as a guide. The challenge was to make the model as light as possible, removing the functionalities that were not needed by the portable version. Moreover, all the necessary transformations and modifications had to be made, in combination with the developing of the appropriate algorithms for the model to be able to be integrated to the palmtop computer – portable device.

The demands were to develop a perfectly running version of the talking head engine for usage in portable devices, without the model loosing its ability to perform all its initial functionalities on the desktop computer. In a few words, the final application would be an engine that will work as both desktop facial animation engine and portable device facial animation engine.

Another demand was to make the new engine able to connect and interact with the server computer through wireless connectivity. This will enable the draw of facial animation frames by making use of the facial parameters calculated remotely in the server computer. As it is explained later in the report, the portable device does not have the necessary processing power and resources to perform the appropriate algorithms for creating the facial deformation parameters for creating visemes frames corresponding to the speech that is produced. But also it is not able to perform any intelligent ASR(Automatic Speech Recognition) algorithms nor running the complicated dialog systems that we want to perform. The portable device only works as a renderer of the visemes and as a sound device that together will produce the multimodal speech synthesis engine on the device. In other words, the concept of the project is to make us of a server that will do all the necessary calculations and dialog management and will interact with the portable device through wireless communication.

Last demand was to perform of usability tests of the new system in order to find the answer to the research question that we put at the beginning of this research.

1.4 Limitations

The limitations that we had to take into account in this master's thesis project played an important role on the nature of the final project. It was important to put the right restrictions for deciding what feature to implement and what would be the proper strategy to be followed in the usability tests.

Limitations are of course found on the nature of embedded systems. Their restricted capability concerning features like memory, CPU power and resources posed severe problems on the development and performance of the final facial animation engine. As it is shown later in the report it was rather impossible with the existing portable device (Dell Axim X51v) to reach the same performance, concerning the frame redraws, as the desktop computer engine. Moreover, all the not needed initial functionalities, which were used in the desktop engine, were removed from the portable version, and even some transformations were made that would make the model much lighter from the desktop version. These transformations allowed us to reach the highest performance in the region of 17 to 18 frames per second instead of the 25 frames per second which is possible to draw on the desktop engine.

Although in the beginning there was a thought of implementing a wholly functional talking head agent and maybe combine some agenda functionality the time constraints restricted the project to only complete the most important features of the project which were the following:

1. The integration of the animation engine to the portable device
2. Establish connectivity with the server that computes the animation facial parameters (FAPs)
3. Synchronize the audio with the visual drawing of the animated face for multimodal speech synthesis
4. Conduct usability tests on subjects using the talking head on the portable device.

One further limitation was the restricted number of subjects that were used for the usability tests. Their background was also a limitation, since they were not belonging to the target group, for which the application was designed. Probably when the model will be able to fulfill all the functionalities of a concrete agent the usability test could be more extended and with subjects coming from the target groups for which the project is build for as mentioned earlier.

1.5 Scopes of use

This project creates a real opportunity for all the developed applications in the department that make use of animated talking faces. Dialog systems that already exist in the department, like August, could make use of the portable talking head platform for establishing an even better communication with the user in terms of Human Computer

Interaction (HCI). The portability of the model opens a big and brand new door of challenges and opportunities in the field of multimodal speech synthesis and human computer interaction.

Another potential use of the engine could be as a part of the Synface [1] project as a terminal device for the multimodal speech synthesis part.

But the ultimate use of the model would be the creation of the intelligent agent that we mentioned in the introduction. An important step towards this direction is the MonAmi (Mainstreaming for Ambient Intelligence) project [5], in which this application is expected to play an important part.

MonAmi is an EU funded program for mainstreaming accessibility in consumer goods and services, including public services, through applied research and development, using advanced technologies to ensure equal access, independent living and participation for all in the Information Society. It must be stated that this project pays special attention on demonstrating that accessible, useful services for elderly persons and persons with disabilities living at home can be delivered in mainstream systems and platforms.

As we will describe later in the report the dialogue system of this innovative speech communication interface is the so called Reminder. It is a system developed in the department for Speech Music and Hearing KTH as a help for people suffering from dementia causing problems in their ability to memorize every day tasks. It is a simple calendar application that uses a speech recognizer, which is set to Swedish, enabling an interaction with the use of Swedish language.

1.6 Outline of the report

This report is divided into 5 Chapters.

The first chapter includes a brief introduction to the project. It also includes the research questions, the method, the aim and purpose of the study.

In the second chapter we give a thorough presentation of previous work on facial animation and main principles followed in the facial animation process. We also go through the main limitations that are encountered when developing for embedded devices, their futures causing these limitations and we present the way of working to overcome them. Next we elaborate on the OpenGL and OpenGL ES APIs for accelerated 3D Graphics, and we provide a brief presentation of the differences between the two platforms, mainly giving the features that were used in this project.

The third chapter contains a step by step description of the work. The method and different strategies followed throughout the development stages is presented in detail. The problems that were encountered are presented in detail together with propositions for future work with this kind of applications.

The fourth chapter describes the tests that were conducted with the use of the portable hybrid version of the facial animation engine and the results that were drawn by this research. The tests were important to be able to give strong and clear answers to the research questions we set out to discuss at the beginning of this work. Unfortunately due to the lack of time the tests performed do not give the whole picture of the full power and the real disadvantages of this application, but it does however give a very important insight of how this kind of applications could be used in future projects that we mention in the next section.

Finally the last chapter includes conclusions and propositions for future work. This includes conclusions concerning the tests and how these could helpful for that better integration of this prototype of a portable talking head in the talking head applications already running on the desktop computers.

In Appendix A we make a brief description of the most important differences between the OpenGL and OpenGL ES rendering API's, while in Appendix B we enclose the questionnaire we used in the usability tests.

2. Background

In this chapter, we give a short presentation of the several technologies which were used during this project. This includes a brief introduction of the different fundamental animation techniques concluding to the MPEG-4 standard. The MPEG-4 standard is used as basis not only for the development of the talking head models already existing in the department of TMH, but is also the one we are using in this project work. A study on the limitations and challenges that lie when developing for embedded systems is also presented. Then, follows an introduction to the framework used for the 3D graphics animations and a short comparison to the features of the platforms designed for desktop computers (OpenGL) and the one designed for the mobile windows applications (OpenGL ES). Finally we present the Khronos Native Platform Graphics Interface (EGL) that was used in the project as an interface between the rendering API (Application Program Interface) OpenGL ES and the underlying native platform window system of the portable device.

The face animation engine to be implemented on the portable device consists of several different parts, each of them responsible for different tasks within the application. On the server there are three components. The first component, is the text to speech engine, which is responsible for creating a waveform containing the utterance described by the input text. The second part is the engine that calculates the facial parameters for creating the visemes, by using either the phonemes created by the text to speech engine or the markup texts without using the TTS engine. Finally on the desktop computer runs also the server side of the client server communication with the portable device. The server and the client are connected through internet connection using the TCP/IP protocol. On the client side, which is the portable device, we are using (Dell Axim X51v), there are also three parts running. First there is the client side of the communication with the server. This part is responsible for establishing communication through sockets and reading the packets coming from the server. These packets contain the waveform sound produced but the TTS engine followed by the facial parameters for drawing the animated face. The second component in the client side is the audio device that reproduces the sound. Lat but not least, is the facial animation engine, which is responsible for drawing the face on the screen of the portable device. This is clearly shown in the picture below.

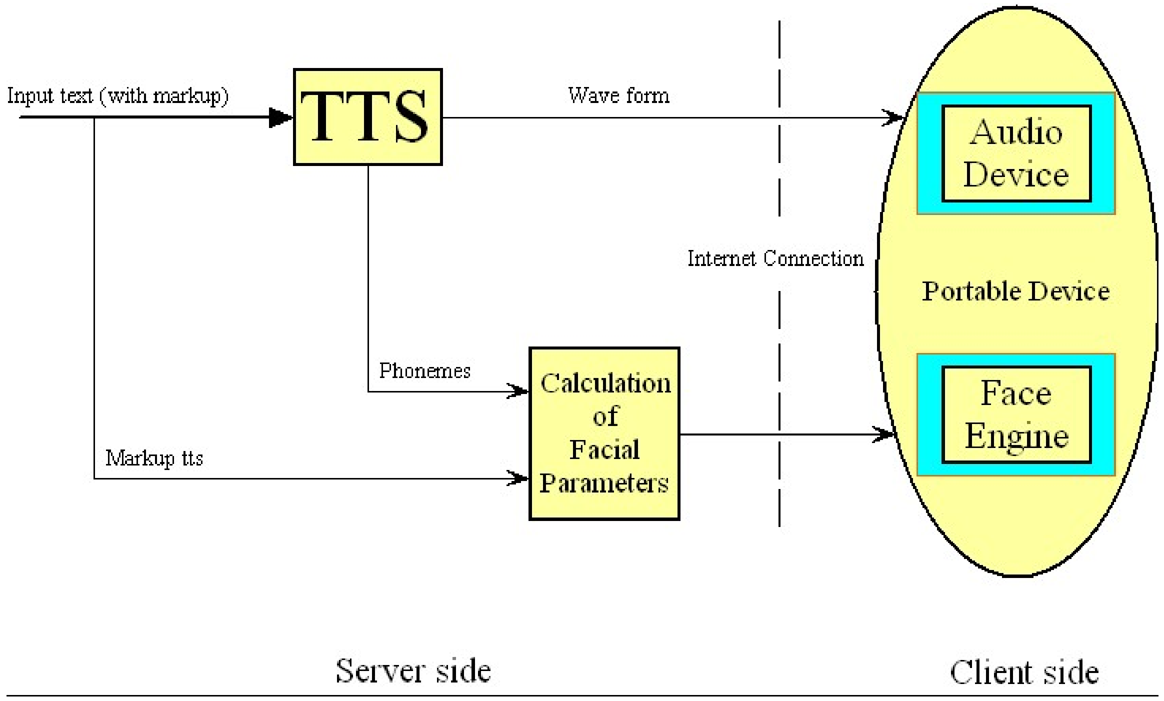

Figure 2. The Talking Head component

The scope of this project was to build all the parts concerning the portable device. These were, to establish connectivity with the serve and, using the incoming data, to produce a talking head application on the portable device. This also included building the face for the PDA, creating the audio feedback utility and synchronizing audio and animation outputs. But let us look closely to the theoretical parts, which we need to know before describing the method used in this project.

2.1 Facial animation

Research on ways to represent human behavior and especially human faces has been going on for the past few decades. Efforts on creating computer animated human faces date back to the early 1970’s. Mostly driven by entertainment end game industry but also medical science and telecommunication companies, the field of computer based facial animation has gone far from the first model presented by Parke in 1972. Nowadays the research is focusing on entire complex and multi-application motion pictures based on computer animation [6].

Basically there are five major techniques used for simulating movements and deformations in facial animation; interpolation, performance animation, pseudo muscle- and muscle-based animation and finally, direct parameterization [1]. These techniques may be used separately, but a system may also use combinations of the techniques.

2.1.1 Parameterization

It is very important to keep the face animation process in a high level so that the user of the animating system should only worry about modeling facial expression and not about translating and deforming every single vertex involved in the drawing process. The techniques mentioned earlier are used for a low-level animation. For a high-level animation a suitable control parameters set is required. Obviously these parameters have to be defined in such ways in order to provide a possibility for high-level face animation. Pandzic describes a somewhat edited version of a taxonomy of a FA-system. This is shown in Figure 1. [7]

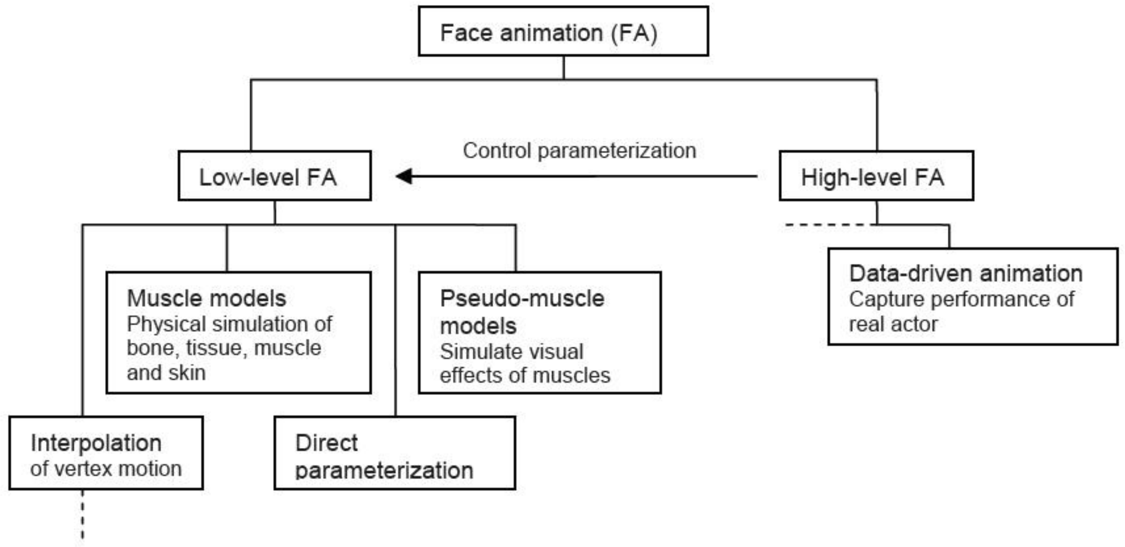
Face animation (FA)
Control parameterization
Low-level FA
High-level FA
Muscle models
Physical simulation of bone, tissue, muscle and skin
Pseudo-muscle models
Simulate visual effects of muscles
Data-driven animation
Capture performance of real actor
Interpolation of vertex motion
Direct parameterization

Figure 3. Pandzic FA taxonomy of Facial Animation(FA) the animator is only concerned about the high level animation.

An extended list of the requirements for an ideal control parameterization, initially described by Parke, is presented by Pandzic and Forchhemier [7].

- *Any face with any expression should be possible to express.*

- *The parameterization has to be easy to use. This demand conflicts with the first demand since a wide range of possible expressions requires a more complex parameterization. Keeping the number of parameters down and making them intuitive is the best way of making the parameterization easy to use.*

- *Subtle expressions have to be expressible. When looking at a face we perceive very small movements, a slightly raised brow, a twitch of the corner of the mouth, and interpret them as signs of different emotions. These are a vital part of the communication process.*

- *One parameter should not affect another parameter. The parameters need to be orthogonal.*

- *The result of a parameter change, or a combination of parameters, has to be predictable.*

- Portability. When applying the same parameters to different face models, the visual result should be the same.

- To allow for data-driven animation, the parameters need to be defined in such ways that they are measurable with a high degree of accuracy. They also have to be able to be derived from a natural face.

- For streaming and real-time applications, a bandwidth-efficient stream of parameters is desirable.

As Jonas Beskow states in his dissertation on talking heads [1], the demands on the parameterization can be quite different and all of these demands may not need to be met, depending on the nature of the application. We have to mention that the second and also the final requirement are very vital for the system that we are designing. The second requirement states that the parameterization must be easy to use and the parameters must be small-numbered. This makes the calculations for the translations and deformations easier and not memory and CPU power consuming. This is extremely important, if we consider the limited capabilities of the portable device, which we are using in this project. The final requirement refers to the stream of parameters and to the need for that stream to be bandwidth-efficient. We would replace the term “desirable” with the more representative, in our case, “demanded”. The stream of parameters must be as “light” as possible so that the portable device is not delaying the draw of the face waiting for a large amount of data delivered by the server.

2.1.3 The MPEG-4 based facial animation

As in other areas of the research community in general, including the field of computer animation of human faces, for a long time there was no standardization on the way of working. That means that there existed no control parameterization standard. As it happens in these occasions, the different FA-systems developed by different research groups used their own ways of parameterization and without a specific standard, which would allow the unhampered exchange of data and results between the various systems.

The MPEG-4 Face and Body Animation standard was the fist facial control parameterization standard and it was developed by the Moving Picture Expert Group (MPEG). The most important part which this new standard introduced was the standard set of low and high level animation parameters (AP) that are defined by the standard. These APs are scalable and can be normalized in any arbitrary face making the standard highly portable over different faces and platforms [8].

2.1.4 FAPs FDPs and FAPUs

Facial Animation Parameters FAPs are specified by the MPEG-4 standard to describe the deformations that take place in different parts of the face during animation. These are 68 parameters divided in 10 subgroups as shown in Table 1 and correspond to different areas of the face [9].

Group		Nr. of FAPs
1.	Visemes and expressions	2
2.	Jaw, chin, inner lower lip, corner lips, mid lip	16
3.	Eyeballs, pupils, eyelids	12
4.	Eyebrow	8

5.	Cheeks	4
6.	Tongue	5
7.	Head rotation	3
8.	Outer-lip positions	10
9.	Nose	4
10.	Ears	4

Table 1. FAP groups of the MPEG-4 standard.

The first group, Visemes and expressions, contains the two high-level FAPs of the standard, the first defining the shape of the face for 14 different static visemes and the second to define the facial deformation for six different emotions; joy, sadness, anger, fear, disgust and surprise. But as stated by Jonas Beskow [1] the true power of the MPEG-4 standard lies in the rest of the FAPs, that work on a low-level. These low-level parameters define each part of the face will be deformed during animation. Osterman [9] has a very nice description on the information that is stored in each FAP and how that is used in the processes of animation. Finally it must be stressed out that for creating good looking talking heads, certain areas of the face acquire more detailed approach then others. That is why in such areas, like the eyes or the mouth, the FAPs are more densely situated, in order to have a higher resolution and better representation of the human face during animation.

Facial Deformation Parameters, FDPs, in coordination with the FAPUs, enable the standard to apply the FAPs to any arbitrary face model permitting it to perform same animations on different models. By assigning Feature Points (FPs) that point out certain key features of the face as spatial references for the FPUs, the standard can perform deformation of generic models stored in the MPEG-4 decoder [10]. The 82 parameters specified by the standard are shown in the picture below as described by Ostermann [9]. Finally the FDP-set can also contain, but not necessarily, texture coordinates, a Face Scene Graph and Face Animation Tables (FATs). Escher [10] gives a thorough description of these properties of the FDP-set. As we will describe in later chapters our model does not use the textures facility and the drawing of the talking head is done purely with drawing of primitive shapes and application of suitable lighting.

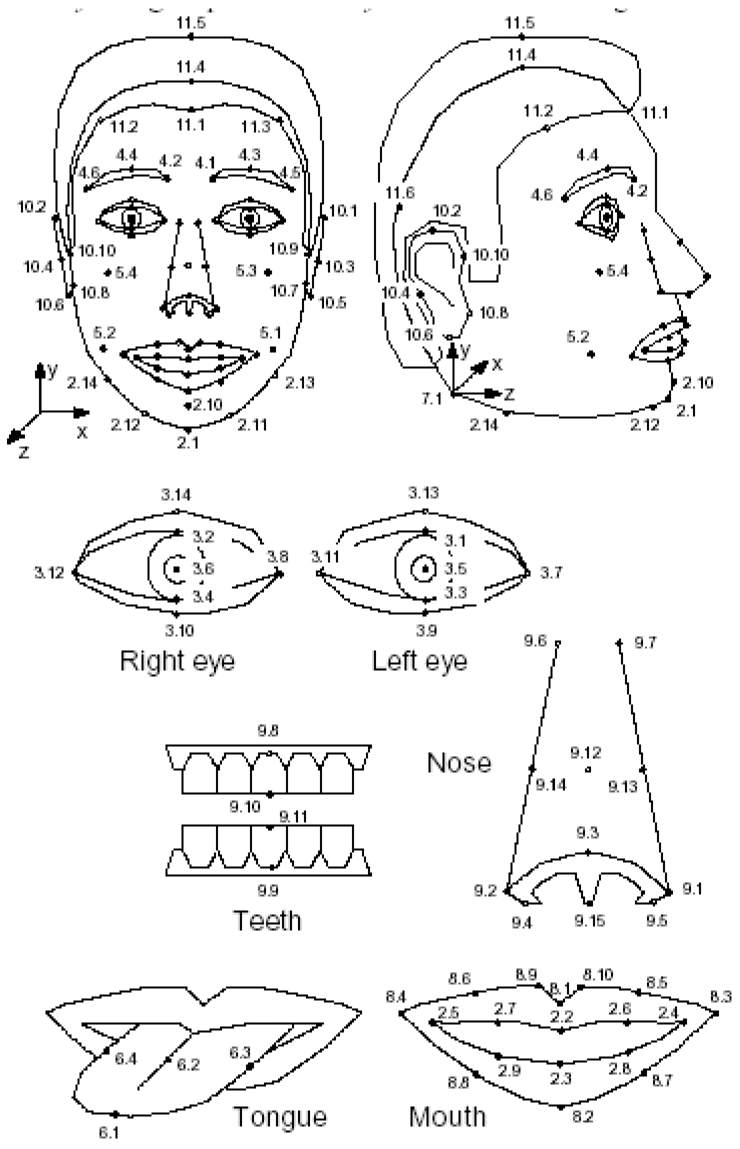

Figure 4. The distribution of feature points across the face.

Face Animation Parameter Units (FAPUs) define five different distances between certain points of the face, when the face is in its neutral state. The FAPUs makes the FAPs scalable between arbitrary face models and their definitions are given below as described by the table below [9]:

FAPU		Definition
1.	IRISD0	Iris diameter, distance between upper and lower eyelid.
2.	ES0	Eye separation, distance between the pupils.
3.	ENS0	Eye-nose separation, length of the nose.
4.	MNS0	Mouth-nose separation, distance between upper lip and lower end of the nose.
5.	MW0	Mouth width.

Table 2. The different Face Animation Parameter Units of the MPEG-4 standard.

2.1.5 The MPEG-4 based facial animation system at TMH-KTH

At the department of Speech Music and Hearing at KTH there are to engines available for facial animation. The most recently used was developed under the EU-project pf-star [11]. The animation module of this version is written in the object-oriented programming language JAVA. The other version of the animation engine is developed in the also object oriented programming language C++. [1] Both of them use the OpenGL API for rendering and displaying the different head models.

We chose to use the C++ implementation of the engine as the basis for developing our system since C++ it is widely used for drawing applications in embedded devices and it would be much easier to work with and find feedback for the different issues and problems that would arise during the implementation process. It is also noticeable that the C++ version would be lighter in terms of recourses needed for running the application. This is, as it is explained in later chapters, very important for embedded applications since they do not have the luxury of high processing power nor large memory and data saving capabilities. Java can present some challenges for embedded-systems developers. First, speed - Java applications are inherently slower than applications compiled into native machine code and embedded processors are generally less powerful than those found on desktops. There is no direct memory access and no interrupt handling, two features usually required for developing low-level software to control hardware. Furthermore, Java built applications can't easily run in real time, to name only a few of the issues.

2.2 Developing for embedded systems

Unlike desktop systems, embedded systems use different user interface technologies; have significantly smaller memories and screen sizes; use a wide variety of embedded processors and have tight constraints on power consumption, user response time, and physical space. As it is easily understood, programming for embedded systems is a whole different and much more complex procedure than dealing with normal Personal Computer (PC) programming. Because of all the limitations described earlier, the programmer must make a lot of effort to try to work with slow processors and low memory, while at the same time battling a need for efficiency comparable with PC applications. Below is a list of issues specific to the embedded field [20].

2.2.1 Tools

Embedded development makes up a small fraction of total programming. There is also a large number of embedded architectures, unlike the PC world where 1 instruction set is used, and the Unix world where there are only 3 or 4 major ones. This means that the tools are more expensive. It also means that they are equipped with lower features in terms of quality and quantity. One more characteristic of these tools is that they are also much more frequently affected by bugs. It is more than sure that on a major embedded project, at some point there will almost always be a compiler bug of some sort.

Debugging tools are another issue. Since general programs are not supported on the embedded processor, it is rather difficult to run any debugging software on it. This makes fixing and setting up of programs extremely difficult. Special hardware such as JTAG (Joint Test Action Group) ports, which is an interface for in-circuit debugging and testing, can overcome this issue in part. However, if we stop on a breakpoint when our system is controlling real world hardware (such as the frame redrawing module in our occasion), non realistic feedback is given and some times even permanent equipment damage can occur. As a result, people doing embedded programming quickly become masters at using serial IO channels and error message style debugging.

2.2.2 Resources

To save costs of the final products, embedded systems are frequently supplied with very cheap processors with limited capabilities. This is not only done because of the cost however. It is obvious that it is not possible to have very strong processors in a limited space and with highly limited battery potentials. This fact means that programs intended for embedded systems need to be written as efficiently as possible. When dealing with large data sets, issues like memory cache misses, will almost never matter in PC programming. This is something that should be taken into serious consideration, though, when programming for embedded systems. It is extremely important to use reasonably efficient algorithms to start, and optimize only when necessary. Of course, normal profilers are doomed not to work well, due to the same reasons mentioned about debuggers. In addition, intuition and an understanding of the developed software and hardware architecture is necessary for effective optimization.

Memory is also an issue. There are serious cost savings issues, in the production of embedded systems. Usually they have the least memory possible for the systems to cope with the requested tasks. Thus, algorithms designed for embedded systems must be memory efficient. Unlike in PC programs, it is a common practice to frequently sacrifice processor time for memory, rather than the reverse. Furthermore memory leaks are forbidden. Embedded applications generally use deterministic memory techniques and avoid the default "new" and "malloc" functions, so that leaks can be found and eliminated more easily.

Other resources programmers expect may not even exist. For example, most embedded processors do not have hardware FPUs (Floating-Point Processing Unit). These resources either need to be emulated in software, or avoided altogether. We will explain later what this means in terms of efficiency and how this was a very important factor for our applications. Since the OpenGL libraries for embedded systems that was used (OGLES) did not support any floating point calculations, all the modules of the OpenGL library that where using floating point were integrated into new OpenGL ES modules that made use of only fixed point calculations.

2.2.3 Real Time Issues

Embedded systems are frequently used to control hardware such as robotic equipments and machines. Therefore they must be efficient in terms of time response to be accurate in real time applications. Failure in response time could cause inaccuracy in measurements, or even damage hardware such as motors. This is made even more difficult by the lack of resources available. Almost all embedded systems need to be able to prioritize some tasks over others, and to be able to put off/skip low priority tasks such as UI in favor of high priority tasks like hardware control.

2.3.4 Fixed-point arithmetic

Some embedded microprocessors may have an external unit for performing Floating Point Arithmetic (FPU), but most low-end embedded systems have no FPU. Most C compilers will provide software floating point support, but this is significantly slower than hardware FPUs. As a result, many embedded projects enforce a no floating point rule on their programmers. This is in strong contrast to PCs, where the FPU has been integrated into all the major microprocessors, and programmers take fast floating point number calculations for granted. Many Digital Signal Processors (DSPs) also do not have an FPU and require fixed-point arithmetic to obtain acceptable performance.

Fixed point arithmetic is used in many 3D graphics engines, like on Sony's original PlayStation, Sega's Saturn, Nintendo's Game Boy Advance (only 2D) and Nintendo DS (2D and 3D). They all use fixed-point arithmetic for the same reasons we already presented: to gain throughput on architectures without an FPU (Floating Point Unit).

A common technique used to avoid the need for floating point numbers is to change the magnitude of data stored in the variables so that fixed point mathematics can be utilized. For example, when adding inches and only to be accurate to the hundredth of an inch is needed, the data can be stored as hundredths rather than inches. This allows the use of normal fixed point arithmetic. It is important to note that this technique works so long as the magnitude of data to be added in advance, and know the accuracy to which you need to store your data.

In our application we are using the transformation from floating point to fixed point with the following simple algorithm.

```
#define __f2x(f)  ((int)(f*65536))
```

which is in simple mathematics is:

$$x = f*65536$$

, where f is the floating point value and x the corresponding fixed point value.

2.3 Dell Axim x51v PDA (Personal Digital Assistant)

The Dell Axim X51v PDA is an extremely solid, reasonably priced and full-featured device and one of the best contenders in the PDA market at this moment. The device is shown in the picture below [21]:

Figure 5. Dell Axim X51v PDA

This device runs on Microsoft’s operating system for mobile devices, That is Windows Mobile. More specifically on Windows Mobile 5.0, which offers many improvements, comparing to previous operating systems for PDAs; the most important being persistent memory. All data and applications are stored in flash memory which will survive a complete battery rundown. WM 5 devices still have RAM, which is volatile and faster than flash ROM. But RAM is used in the same way as a normal desktop PC uses it: running programs are loaded into RAM and operating system files are cached there to improve response times. Programs or files are not longer installed into RAM, only into flash ROM. Since the device need not power RAM at all times, battery life is improved by approximately 10%.

Windows Mobile 5's improved user interface makes some tasks a bit quicker and more intuitive and Internet Explorer, Word Mobile and Excel Mobile are more capable.

The X51v is in the top tier of current Pocket PCs thanks to its 624MHz "Bulverde" Intel XScale PXA270 processor. As it is expected, the unit is fast, though there might be a slightest delay opening menus and windows since the device is running at VGA resolution and uses ROM rather than RAM for application and file storage. It is up to the task of even the most demanding applications including action games and video playback. WM5 is a bit slower than Windows Mobile 2003 because applications and data reside in ROM which is slower than RAM.

Windows Mobile 5.0 caches operating system files heavily to improve performance, and uses a significant portion of available RAM. At boot, 49.47 megs of RAM are shown as available, and another 19 megs of that available memory are used by the OS. That means there will be about 30 megs out of 64 available to run programs, which is certainly adequate even for power users.

The unit has 256 megs of flash ROM, 195 of which are available as internal storage. The operating system and related programs are permanently installed in the remaining 59 megs of ROM. Dell always uses Intel StrataFlash memory for flash ROM which is faster than NAND.

The X51v has integrated WiFi 802.11b wireless networking and Bluetooth 1.2. Wi-Fi has excellent range and is proven to be extremely reliable when connecting to 802.11b access points using WEP encryption. Although the hardware supporting the wireless connectivity seems to be quite robust, we encountered some serious problems with the ActiveSync software utility, used by the PDA to connect and synchronize with the desktop computer. Many times this program was automatically popping up, blocking, in the same time, the active wireless connection. This was only solved by making a fake server connection “fooling” the operating system of the PDA that we were already connected to a server computer.

Finally the main reason for choosing this piece of equipment is its powerful 3D graphics chip. The X51v uses the Intel 2700G graphics processor with 16 Megabytes of RAM. That's an impressive amount of video memory for a PDA, and beats out all other models on the market. The 2700G comes in two versions with 384k and 704k of on-die video memory, but the chip can address additional external memory and that's how Dell got 16 Mb into this model. The graphics processor also offers 3D acceleration, while all other PDAs except the now discontinued gaming-oriented Palm OS Tapwave Zodiac have only 2D acceleration if any at all. The 2700G was designed by Intel to work with the PXA270 family of processors, and it has a fast pipeline between the CPU and graphics processor.

The graphics accelerator performs excellently for GDI (Graphics Device Interface) and Open GL based software rather than GAPI (Game API) which is currently most commonly used by Pocket PC applications, especially games. [12]

2.4 Open GL Open GL ES and EGL

2.4.1 Open GL

Open Graphics Library (OpenGL) is a software interface to graphics hardware. The interface consists of a set of several hundred procedures and functions that allow a programmer to specify the objects and operations involved in producing high-quality graphical images, specifically color images of three-dimensional objects. Most of OpenGL requires that the graphics hardware contain a framebuffer. Many OpenGL calls pertain to drawing objects such as points, lines, polygons, and bitmaps, but the way that some of this drawing occurs (such as when antialiasing 1 or texturing is enabled) relies on the existence of a framebuffer. Further, some of OpenGL is specifically concerned with framebuffer manipulation. [13]

2.4.2 OpenGL ES

- The Standard for Embedded Accelerated 3D Graphics

Open Graphics Library for Embedded Systems *(OpenGL ES) is a royalty-free, cross-platform API for full-function 2D and 3D graphics on embedded systems - including consoles, phones, appliances and vehicles. It consists of well-defined subsets of desktop OpenGL, creating a flexible and powerful low-level interface between software and graphics acceleration. OpenGL ES includes profiles for floating-point and fixed-point systems and the EGL™ specification for portably binding to native windowing systems. OpenGL ES 1.X is for fixed function hardware and offers acceleration, image quality and performance. OpenGL ES 2.X enables full programmable 3D graphics. OpenGL SC is tuned for the safety critical market." (Khronos Group)* [14].

For fixed function hardware: OpenGL ES 1.X is defined relative to the different OpenGL specifications, From OpenGL 1.3 to OpenGL 1.5. OpenGL 1.0. is defined relative to the OpenGL 1.3 specification and emphasizes enabling software rendering and basic hardware acceleration. On the other hand the OpenGL ES 1.1 is relative to OpenGL 1.5 and emphasizes hardware acceleration of the API, but is fully backwards compatible with 1.0. It provides enhanced functionality, improved image quality and optimizations to increase performance while reducing memory bandwidth usage to save power. The OpenGL ES 1.1 Extension Pack is a collection of optional extensions added to OpenGL ES 1.1 that reduced variability and bring significant improvements in image quality and performance.

There are two OpenGL ES profiles, the Common and the Common-Lite profile. In the Common profile, the wide range of support for differing data types (8-bit, 16-bit, 32-bit and 64-bit; integer and floating-point) is reduced wherever possible to eliminate non-essential command variants and to reduce the complexity of the processing pipeline. Double-precision floating-point parameters and data types are eliminated completely, while other command and data type variations are considered on a command-by-command basis and eliminated when appropriate. In the Common-Lite variation of the Common profile, the floating-point data type is also eliminated in favor of the fixed-point data type.

The Common-Lite profile is a fixed-point profile. The precision and dynamic range requirements are minimal to allow a broad range of implementations, while strong enough to allow portable application behavior for applications written strictly to the minimum behavior. The accuracy requirements allow pipeline implementations to internally use either fixed-point or floating-point arithmetic. The Common profile is a superset of the Common-Lite profile and requires floating-point-like dynamic range to avoid unexpected behavior in applications using floating-point input.

The fixed point computations for the Common-Lite profile has to fulfill the following requirements. Fixed-point computations must be accurate to within ±2^−15. The maximum representable magnitude for a fixed-point number used to represent positional or normal coordinates must be at least 215; the maximum representable magnitude for colors or texture coordinates must be at least 210. The maximum representable magnitude for all other fixed-point values must be at least 215.The arithmetic definition is:

- $x \cdot 0 = 0 \cdot x = 0$.
- $1 \cdot x = x \cdot 1 = x$.
- $x + 0 = 0 + x = x$.
- 0^0 = 1.

Fixed-point computations may lead to overflows or underflows. The results of such computations are undefined, but must not lead to GL interruption or termination [15].

2.4.3 EGL

EGL is an interface between rendering APIs such as OpenGL ES or OpenVG (The standard for Vector Graphics acceleration - referred to collectively as client APIs) and an underlying native platform window system. It refers to concepts discussed in the OpenGL ES and OpenVG specifications, and is firmly related to these concepts. EGL uses OpenGL ES conventions for naming entry points and macros. EGL provides mechanisms for creating rendering surfaces onto which client APIs can draw, creating graphics contexts for client APIs, and synchronizing drawing by client APIs as well as native platform rendering APIs. EGL does not explicitly support remote or indirect rendering, unlike the similar GLX API.

As it is stated in the EGL 1.3 specification, EGL is intended to be implementable on multiple operating systems (such as Symbian, embedded Linux, Unix, and Windows-Windows Mobile 5.0) and native window systems (such as X and Microsoft Windows). It is a very powerful tool for creating drawing contexts and that is why it is chosen for the development of this application.

3. Method

Below the process of the work is described. As mentioned, the goal of this project was the integration of an animated talking face to a portable device for multimodal speech synthesis. The procedure includes three different steps. At first the facial animation engine has to be deployed in the portable device. Next connection with the server, computing the facial deformation parameters in relation with the utterance to be played, has to be established. Finally synchronization of the audio and visual streams has to be achieved. But first of all we will shortly present the tools that have been used as well as the required technologies. (C++, OPENGL ES). Then we describe the actual process of the work of integrating the talking face to the portable device, porting OpenGL to OPENGL ES, connecting with the dialog system, and synchronization of video and audio streams.

3.1 Tools

The tools we had for working on this project were:

- The facial animation engine developed in the Department for desktop computers. The platform is developed with C++ as a programming language.(Gub.lib)

- The sven.gub facial model file.

- A portable device for deploying the portable engine,(Dell Axim X51v).

- Microsoft Visual Studio 2005 as a developing tool

- Microsoft ActiveSync for connecting and synchronizing the portable device to the desktop computer were the development was taking place.

- OpenGL ES for Embedded Accelerated 3D Graphics rendering.

- The Software Development Kit (SDK) used was: SDK_OGLES-1.0 for Windows Mobile 5.0 Pocket PC ARMV4I CommonLite profile. (SDK_OGLES-1.0_WINDOWSMOBILE5_POCKETPC_ARMV4I_COMMONLITE_2.00.20.1514.zip)

- C++ as a programming language

- The rendering callback routine implemented by the Imagination Technologies Ltd [17] for the OpenGL ES SDK User Guide [18]. This also uses EGL as a native platform graphics interface.

- Lots of imagination.

3.2 OpenGL Study

As I was not familiar with OpenGL, a very important part when I started with this project was to extensively study the OpenGL Programming Guide [19] that was proposed to me by Jonas Beskow as a very good introduction to OpenGL. The study was mostly focused on the chapters that were concerning the project in hand and those were, State Management and Drawing Geometric Objects, Viewing, Color, Lighting, The Framebuffer and Error Handling. After thorough study of the book and experimenting with the OpenGL, GLUT and GLU utilities I started to study the code that was used in the development of the desktop facial animation engine.[1]

3.3 Changing the existing Desktop Facial animation Engine

The existing desktop facial animation engine was used in several cases, from language teaching to multimodal interaction applications and facial expression tests. It is more than evident that it was designed to support several functionalities that were not at all needed by the portable version. All these functionalities made the model quite heavy for the limited processing power of a normal palmtop computer. All these functionalities require a lot of processing power, memory and several hardware and space resources which are not available in the usual portable devices.

This was the first thing we had to think about. What were the functionalities that were not going to be used on the portable device and how they would be excluded from the portable version of the engine without, on the other hand, having any effect on the initial engine. Consequently, after we decided, always in collaboration with Jonas Beskow, who was responsible for the development of the desktop engine, which functionalities were to be excluded, we started to separate the code that was corresponding to the support of such functionalities from the code that would be used for the development of the portable device talking face engine. Code that was used for texturing, picking [18] or layered design modules were removed since they were not needed and they could be excluded without having any negative consequences to the talking head application. This required extensive study of the programming code of the desktop engine project and several meetings with Jonas Beskow.

The next step was to study the differences of the OpenGL library and the corresponding OpenGL ES designed for embedded systems. The OpenGL framework was used for the development of the desktop facial animation engine. Therefore we decided to use the OpenGL ES standard in our effort to integrate the talking face component to the portable device. In Appendix A we provide an extended presentation of the differences between the two APIs.

The next step was the process of making the required changes to the existing code, and especially to the parts of the code concerning OpenGL routines. Changes were made on several parts of the code of the desktop engine and in different extend depending on the functionality and on what amount the OpenGL routines where compatible with the corresponding OpenGL ES routines. Below we present a small part of the code that was changed:

```
void cXXXSurface::drawPolygonsTQVA() {
  GLERRORCHECK(g,"before cXXXSurface::drawPolygonsTQVA",this-
>get_name().c_str());
#ifdef OGL
  glVertexPointer(3,GL_DOUBLE,sizeof(cVertexNormal),vertices-
>first());
  glNormalPointer(GL_DOUBLE, sizeof(cVertexNormal),&((vertices-
>first())[0].normal));
  if (vnode_ptr() != NULL && vnode_ptr()->haveTexCoords_flag &&
texture_mode) {
    glTexCoordPointer(3,GL_DOUBLE, sizeof(cVertexNormal),
                  &((vertices->first())[0].tex));
    glEnableClientState(GL_TEXTURE_COORD_ARRAY);
  }
#else
  /*
  * Check if the vertices were changed.
  * If so recompute the Fixed vertices
  */
      if (vertices->vertices_changed) {
      int j=0;
      int i=0;
      vtx_d.reserve((vertices->length())*3);
      nrm_d.reserve((vertices->length())*3);
      for (i=0;i<vertices->length();i++) {
            vtx_d[j]=((int)f2vt((*vertices)[i].x));
            vtx_d[j+1]=((int)f2vt((*vertices)[i].y));
            vtx_d[j+2]=((int)f2vt((*vertices)[i].z));
            nrm_d[j]=((int)f2vt((*vertices)[i].normal.x));
            nrm_d[j+1]=((int)f2vt((*vertices)[i].normal.y));
            nrm_d[j+2]=((int)f2vt((*vertices)[i].normal.z));
            j=j+3;
      }
      glVertexPointer(3,VERTTYPEENUM,0,vtx_d.first());
      glNormalPointer(VERTTYPEENUM, 0,nrm_d.first());
      vertices->vertices_changed = false;
      }
#endif
   cbug(name << ":rpTQVA,TQupdated_flag = " << TQupdated_flag
<<endl);
   if (!TQupdated_flag)
        updateTQIndices();

#ifdef PROFILING
```

```
   if (g->noDraw_flag) return;
#endif
   if (tri_indices.length()) {
#ifdef OGL
    glDrawElements(GL_TRIANGLES,tri_indices.length(),
               GL_UNSIGNED_INT,tri_indices.first());
#else
      glDrawElements(GL_TRIANGLES,tri_indices.length(),
               GL_UNSIGNED_SHORT,tri_indices.first());
#endif
  }
#ifdef OGL
  if (quad_indices.length())
    glDrawElements(GL_QUADS,quad_indices.length(),
               GL_UNSIGNED_INT,quad_indices.first());
#endif //OGL
}
```

Table 3. Example of differences between OpenGL and OpenGL ES implementation.

In the presented code the part that concerns the OpenGL implementation is under the "`#ifdef OGL`" pre-compiler settings, while the other code concerns the OpenGL ES version.

3.4 Debugging on the PDA

After making all the initial changes the engine was compiled successfully and the project was ready to be deployed to the portable device in order to start the debugging process. It has to be pointed out that through all this process we had no actual result of the work whatsoever and nobody could guarantee that the model would be able to work in the portable device. Taking into consideration the limited processing power and resources of the PDA combined with the differences of the OpenGL and OpenGL ES APIs there was no guarantee that the project would be able to run successfully.

The debugging process was the most time consuming part of the project since the model, which was about to be implemented was very complicated and consisted of extended source code. Numerous problems concerning different challenges of the embedded systems design and developing process were encountered on a regular basis during the debugging procedure. Memory overflows and underflows while doing arithmetical computations using fixed point values were very common and needed to be taken with special care, especially during the implementation of the gluLookAt function from the GLU ES utility. Very frustrating were the problems that were encountered when loading the facial parameters from the model file. They were very common and close related to the platform that was running on the portable device. Here we make a small presentation of the steps the project went through before the complete working program was ready:

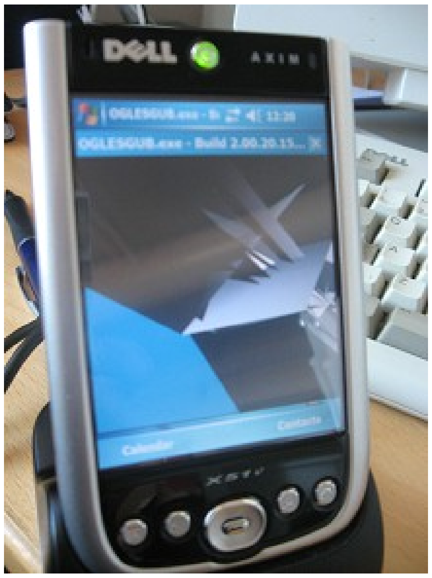

Figure 6. The first drawing result when we tried to run the compiled application.

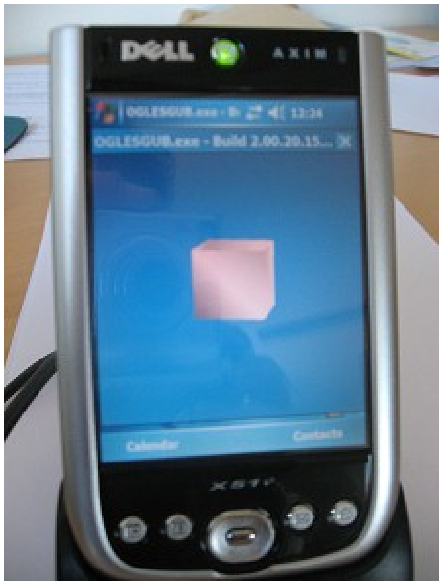

Figure 7. Back to the basics. Drawing of a simple primitive using the settings of the talking head model but using static vertices and polygons.

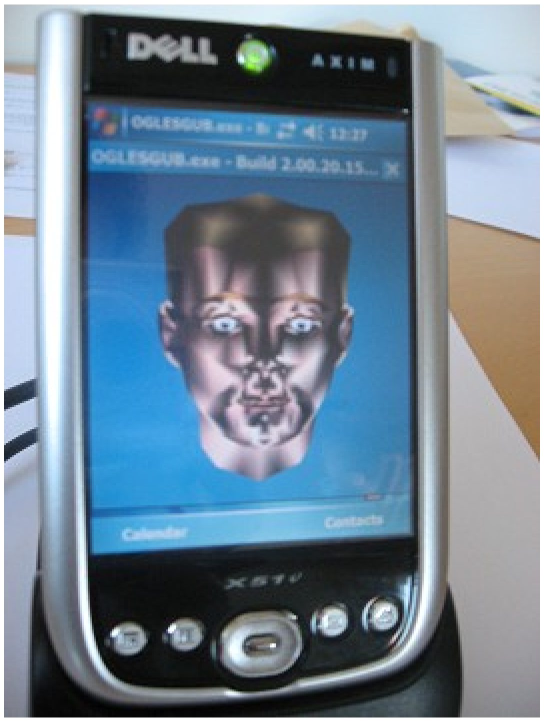

Figure 8. First sight of mister Sven!

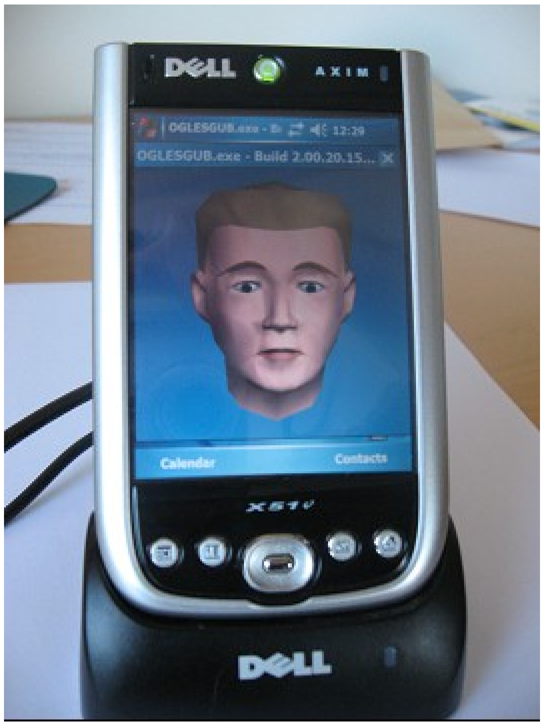

Figure 9. There is Sven!

3.5 Running the model

After the changes were done the model was ready to run on the portable device. When we were sure that the model was running efficiently on the portable device, tests with facial parameters loaded from a local folder in the same directory as the executable file were conducted. The utterance “Hallå världen” was chosen for the tests as the most common message for any new build applications. We prepared a file that was containing the facial parameters for this sentence and provided it in addition with the wave sound file to the model so it could produce a first version of the desired facial animation engine. This is shown in the figure below.

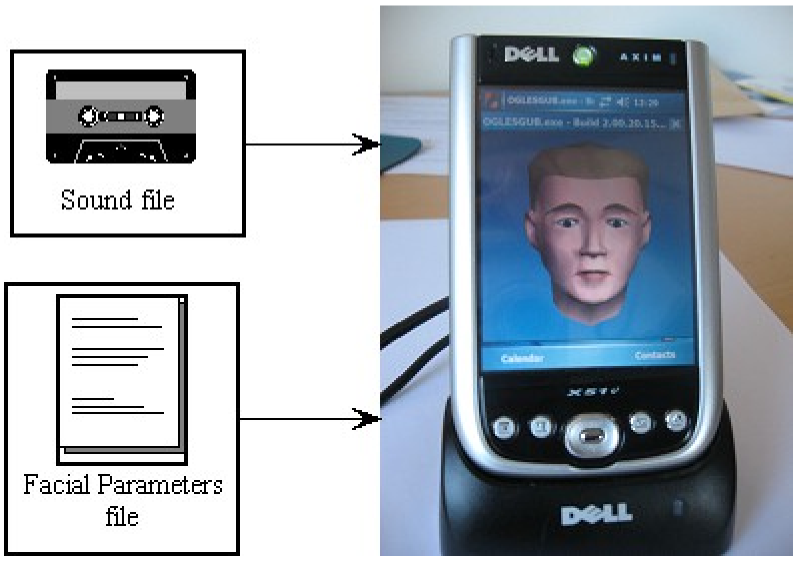

Figure 10. Run the model locally.

Running the model locally we had the chance to measure the performance of the model. This is a very important issue of this project and that is why we are going to elaborate more on this subject. On the desktop computer it is very easy to get feedback from the running application, which is being developed, just by executing it in the command line console. Then it is possible to send print events to the console to check different aspects of the application. This is not the case for portable devices where as it was explained, the resources in terms of hardware and software are extremely limited. The lack of the command line utility forces us to check the application performance by writing to a file. The frame rate of the render model, as written in the output file, is 17 frames/second. It would reasonable to assume that without the file opening/writing process we expect the performance to rise up by one or two frames thus reaching a performance of 18 to 19 frames/second.

3.6 The remote server communication and final result

The next step was to establish communication with the server computer in order to receive both the audio stream and the facial parameter to be used for the rendering of the animated talking face. The connection was established using sockets over server-client architecture and through wireless internet connection using the TCP/IP Connection protocol.

The data are sent through the socket in packets of a maximum length of 512 bytes that is sent by the server. Each packet consists of a field containing the version of the protocol we are using and a field containing the code specifying the nature of the data contained in the packet. The code is an integer that has values 1-4.

- For Code = 1 the packet contains fixed data for the facial animation. The data consist of an integer specifying the length of the data containing the fixed numbers. The packet is as follows.

Header	Code	Packet Data Size (int)	Fixed Data
<6 bytes>	<4 bytes>	<4 bytes>	< Packet Data Size >

- For Code = 2 the packet contains binary data for the wave sound to be played. The data consist of an integer specifying the length of the binary data for the wave sound to be played. The packet is as follows.

Header	Code	Packet Data Size (int)	Wave Sound Data
<6 bytes>	<4 bytes>	<4 bytes>	< Packet Data Size >

- For Code = 3 the packet contains only 8 more bytes that correspond to the integer values for the length of the fixed and sound data. This packet is always delivered first and is signalizing the beginning of transmission. The packet is as follows.

Header	Code	Fixed Data Size (int)	Wave Sound Data Siize (int)
<6 bytes>	<4 bytes>	<4 bytes>	< 4 bytese >

- For Code = 4 the transmission has ended, the data are loaded to memory and we are ready to play the sound and start the facial animation process. This packet contains no further data.

Header	Code
<6 bytes>	<4 bytes>

```
The fixed data are arranged in the file in a sequence of integers. 17
sequential integers are responsible for one frame:

# fixed-data:
# N x 17 x [int]

The wave data are binary data forming a file following the wav-format

# wave-data:
# wav-formatted file
```

We used the server-client communication using TCP/IP sockets. This is a very common and well known architecture for establishing communication between two or more remote computers. For this reason we will not extend on describing the nature of the architecture but rather only provide a picture describing the basic idea of the architecture.

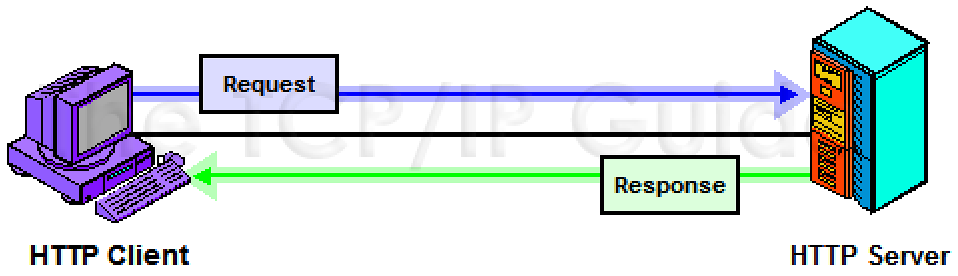

Figure 11. Client Server Communication architecture

Eventually, the socket is read and the chunk containing the facial parameters was saved in the memory while the chunk containing the sound to be played is saved in a wav format file in the same directory as the executable file.

The final step of the development process was to synchronize the audio and visual output for the user. The facial parameters are saved in memory and each sequence of 17 parameters are responsible for the drawing of one single frame. The problem of synchronization is to play the correct frame at the correct moment always in relation to the sound that is played. Taking into account that the most preferable frame rate for the human eye is 25 frames per second, in an ideal situation we would have to draw one frame every 1/25 seconds. So the problem is how to control this rate. For example if we are drawing different frames faster than this rate the human eye will not be able to see all of them since they will be changing too fast. On the other hand, if we are drawing all the frames in a slower rate the result will be the animation to be slow and the sound to be ahead of the face animation. The trick is to play the same frame if the redraw is faster than the required rate, or to skip the drawing of frames if the rate is slower. This is done through a very simple procedure where we save the time stamp just after the command for playing the sound file and then every time we are about to make a new draw on the screen we do a simple computation to calculate the frame to be played. The algorithm chosen is as follows.

```
PlaySound();
StartTimeMSec = (msec)GetCurrentTime();
while(redraw)
{
   Frame = (int)((msecs)(GetCurrentTime())-StartTimeMSec)*.025);
   Redraw(Frame);
}
```

Table 4. Audio-Visual synchronization.

The flow diagram of the application is shown in the picture below:

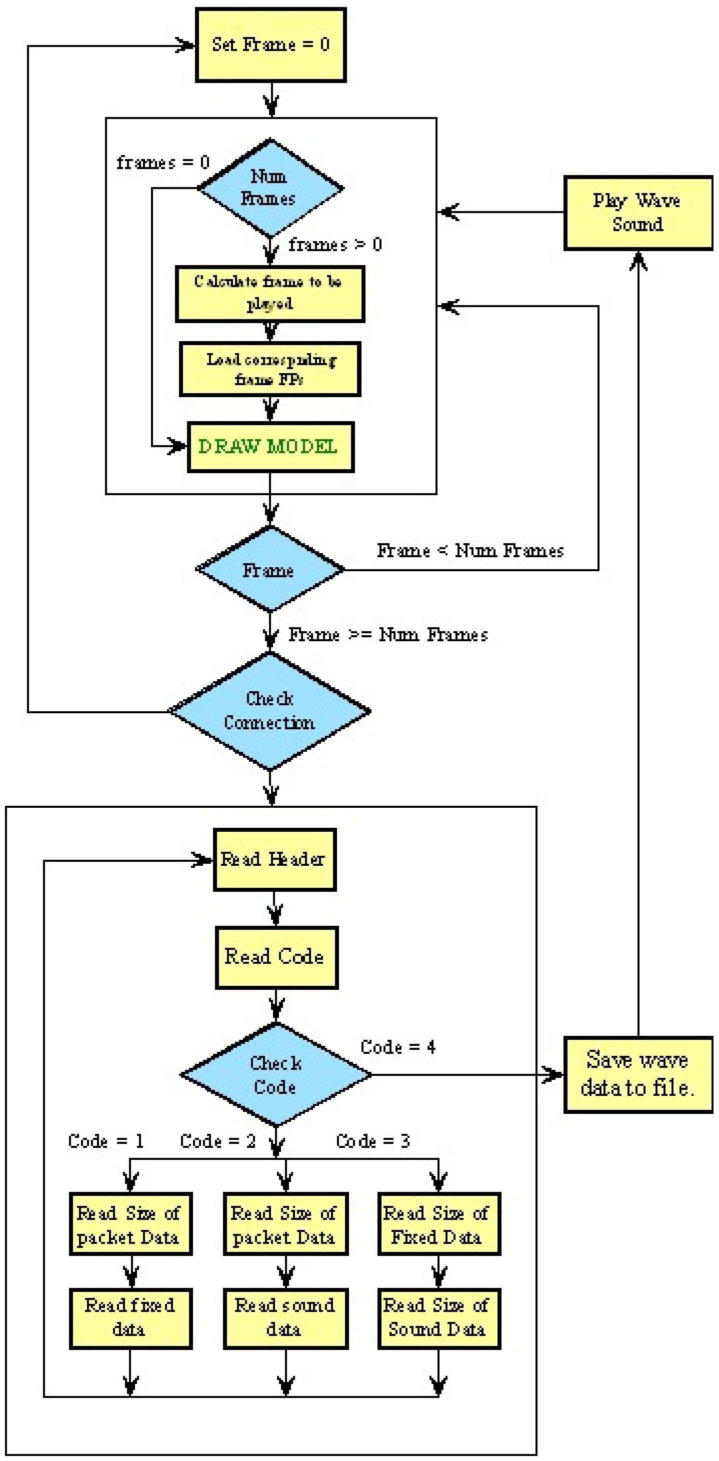

Figure 12. Application Flow.

In our application the rate of the redraw of the frames was lower than 25 frames/second. The redraw process reached a rate of 18 to 19 frames/second. We will explain in the next chapters, about the possible effects this imperfection could have to the perception of speech by users.

4. Evaluation/Results

The result of this project is a functional face animation engine for a portable device, which makes the interaction with animated talking heads merge to the physical environment by introducing portability. To evaluate the developed tool a number of tests have been conducted. The subjects of these tests differ in gender, age and background. This diversity helped to reveal strengths but also imperfections in the tool. In this chapter we describe how these tests were processed and how they revealed important information about the face animation engine that we developed. By analyzing the results, some limitations and flaws can be identified and also insight about future work can be gained.

The first tests were concerning the robustness and performance of the model. The application was tested over the network with a big amount of data and the results were very promising. After minor changes in the settings both in the client PDA side and on the server computer side the application was robust and ready to be used for the tests. The tests were useful for determining the size of the packet we use in the communication as well as finding a more reasonable rate of sending the data from the server.

The next step was to conduct usability tests using real subjects to measure the performance of the application in terms of user friendly interaction.

4.1 Test description.

The tests were conducted as follows. The new multimodal speech synthesis engine was connected to a dialogue system running on the server. This dialogue system is the so called Reminder. It is developed in the department for Speech Music and Hearing KTH as help for people suffering from dementia causing problems in their ability to memorize every day tasks. It is performing a simple calendar application, which is reading tasks that are registered in a calendar. In our case we are using Google calendar. It returns the tasks on demand and depending on the dates. The speech recognizer is set to Swedish, enabling an interaction with the use of Swedish language. This dialogue system is used in the part with which the department is participating in the MonAmi project.

The user interacts with the server through the use of a wireless microphone and the server after proceeding the user’s request, sends the necessary data, facial parameter and wave file, through internet to the PDA so that the answer can be played on the portable device.

The facilitator, in this case, me, first explains the procedure of the test, the tasks that the subject is asked to perform and of course describes to him what the subject of the tests at hand is. Next he observes the test and makes all the necessary notes.

In the tests, the facilitator was playing a very important part. He was responsible for explaining to the users the test procedure. This included a brief description of the application that is being tested as well as a presentation of its capabilities and incapabilities. He was also describing what exactly is being tested so that the user knows where he is focusing his attention. His task during the tests was to observe the procedure and make notes about the problems the users were encountering and check if the requirements, that were set, were fulfilled.

The test procedure is shown in the picture below.

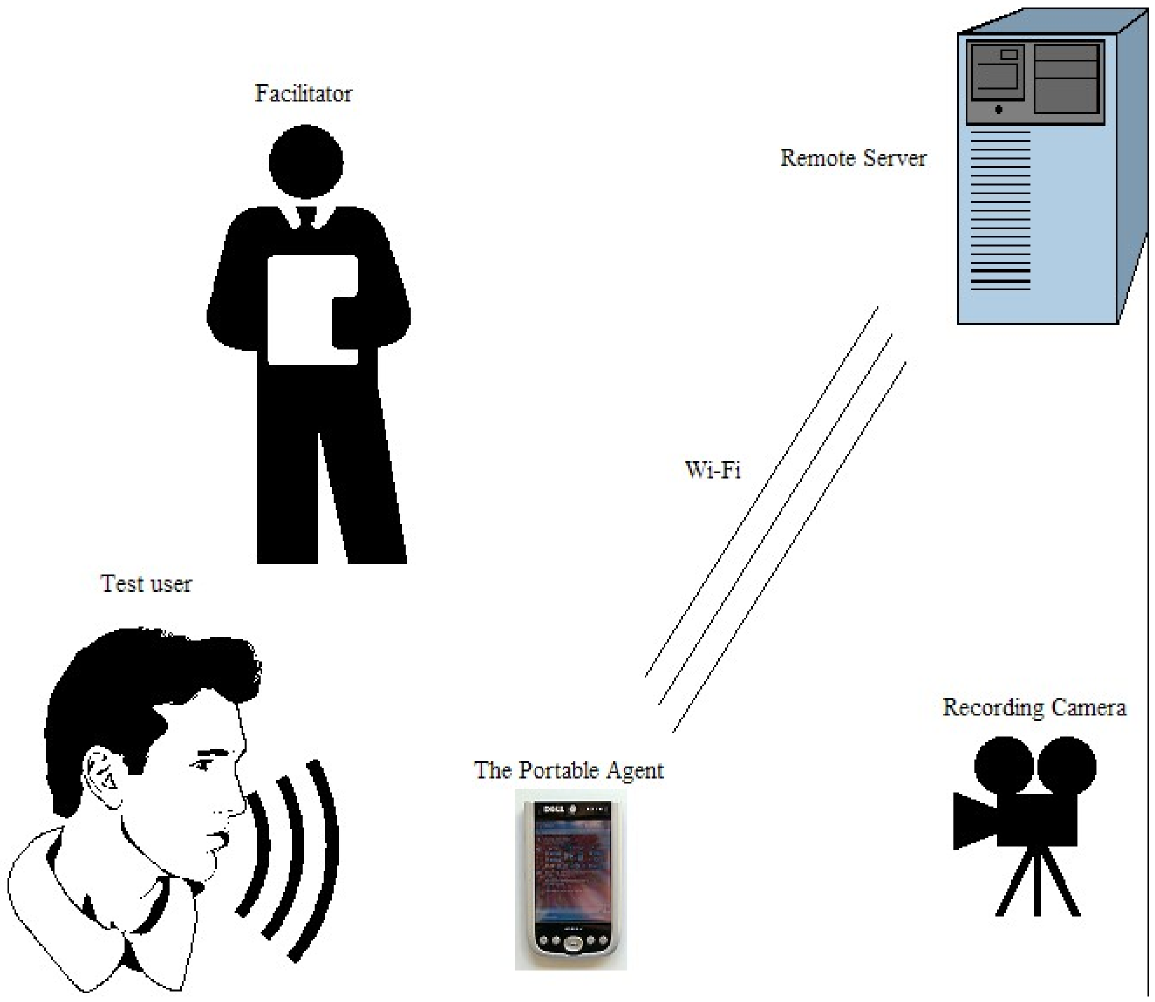

Figure 13. The Test Procedure.

The use of wireless microphone gives our system the complete ubiquitous character that we were aiming for in the beginning of this project. Although we started up with the idea of sending the audio input of the user through internet in a similar way as it is done right now from the server side to the PDA, the solution of the wireless microphone was a safe way to make the application completely independent from the server and totally ubiquitous, quite faster and easier than dealing with a whole new communication issue between the PDA and the server. Eventually the user can move around in the room or even in a long distance from the desktop server computer, thus making the dream of portability of the talking heads application and multimodal interaction a great new feature in speech driven applications and the ubiquitous computing field.

4.2 The Speech Dialog material and the Test requirements

The test procedure consists of a set of predefined tasks that the user is asked to perform using the portable agent. The tasks are

- Ask the calendar for the forthcoming seminars.
    - "När har jag seminarium?"
- Ask for the writer of the message.
    - "Vem skrev det?"
- Lunch or meeting dates.
    - "När äter jag lunch?"
- Forthcoming flights.
    - "När tar jag flygget?"
- Also ask the day's activities.
    - "Vad har jag idag?"

The user is free to move around the testing room and to interact with the agent doing the tasks that he is asked to do. The evaluator, in this occasion, me, discreetly observes the procedure, noting any difficulties the user is encountering and checking the list of requirements that the system is need to fulfill. The list of requirements is as follows:

- Tasks completed easily.
- The user is moving around and still is able to interact

4.3 The subjects

We deliberately tested the portable agent in subjects that were varying in age and gender in order to have the best possible insight on the effects the agent will have in different target groups. We used 20 subjects, 15 men and 5 women with the age varying from 23 to 62. All subjects were students or employees in the department of TMH, KTH

4.4 The follow up questionnaire

To find out the subjects' general feelings on the interaction with the portable talking face agent a follow up questionnaire was designed (Appendix B). The questions of this questionnaire were: "How natural did you feel the interaction with the portable agent was?", "Did you feel more free to interact with an agent that is not fixed on a desktop computer?", "Would you use the talking head agent as an everyday tool for questioning your calendar?". We use a rating system from 1 to 5 to give the chance to the subjects to have as many choices as possible for expressing their opinions. The values from 1 to 5 were concerning answers different for every question but they were all using the same scaling logic. At the end of the questionnaire there was also a small empty place so that the subject could also give some additional comments, suggestion or even thoughts about fields of application for the proposed agent.

4.5 Results

The results from the tests gave us important information about how the demo application was treated by the users and also gave us valuable insight for future work and tests. It was interesting the fact that from the 20 subjects that tested the application, 10 of them were people working in the department and were used to communicate with agents running in desktop computers supported by powerful dialog systems, which were very social and with much more intelligence then the simple calendar dialog that we tried to test the portable device with. Obviously, this was not a good idea, since the experienced test users were having rather high expectations for this first demo of the portable agent. The poor performance of the dialog system, in combination with the slightly delayed answers from the system, caused a small frustration and the interaction was far from natural. With inexperienced users however, who were strictly following the dialog that was proposed, the results were rather more promising, since they were in general very satisfied with the performance and the featured functionalities.

This created an initial confusion to the people that were already used to interacting with similar agents running on desktop computers. One other thing that we noticed which was interesting was that when testing in noisy environment people tend to use the portable device as a phone. Thus when they were having difficulties hearing what was said they would skip the talking head and would put the device near to the ear to listen what the agnet is saying. This was only solved when we clarified that we are testing the face and that they were supposed to use it in order to understand what is said even in noisy environments. Probably this was a problem related with the fact that the subjects tested were not the ones that the system is mainly designed for, which are the hearing impaired or elder people with difficulties in hearing. On the other hand nobody complained about the way the agent was speaking. The face looked very natural to them and the speech was perfectly synchronized with the sound so that the result was not disturbing the perception of speech at all. Although we did not asked this, it was very important that no one had to notice something undesirable about the actual result of the animated talking face drawing.

As described in the test procedure, while the subjects where interacting with the portable agent we were observing the process and making notes about the procedure and about two important clues of the test.

4.5.1 Tasks completed easily

The users were asked to interact with the agent by directing some simple questions, concerning entries in the Google calendar. The agent was not characterized by great intelligence and his social behavior was rather absent. By social behavior we mean that he could not understand sentences like “hallå!” or other utterances irrelevant to the calendar entries. And by intelligence we mean that the questions to be made to the agent were rather restricted to simple entry queries with keywords like “lunch”, “mötte” or “flygg”. This caused a big confusion, especially with expert users of talking head agents as they were expecting him to be social and in the same time to be able to answer to more complex questions than the ones suggested by the facilitators. This lead to wrong and totally irrelevant answers from the agent and this was extremely annoying to the subjects. Consequently we noticed that the behavior towards the agent was changing and the subjects were not patient at all at his new born version of the portable agent.

On the other hand, when the subjects where not expecting so much from the dialog system that was running behind the agent, the tasks where completed rather easily and the subjects where obviously satisfied. The interaction was smooth and quite normal and the talking agent was attracting the subjects' attention. It was really interesting to see that the subjects were paying attention to what the agent was saying and their perception of the produced speech did not change with the talking face application, meaning that the movements of the agent were natural and entirely synchronized with the audio signal.

4.5.2 The user is moving around and still is able to interact

One important part of the testing procedure was that we had to make sure that the subject was free to move around the testing room and still be able to interact with the agent. Although we expected that this would be a very easy thing to test since the device was entirely independent and portable this did not prove to be the case in all occasions. In some occasions the users did not feel comfortable to move around in the room. This was because they were not used to, and they could not really treat the portable device running an application like this, as they would treat their mobile phone for example. In general however, all the subjects, who chose to move around interacting with the agent, felt really free in the process and this is also recorded in the results of the answers we got from the questionnaires that followed the tests.

4.5.3 The questionnaire results

The answers that we got from the questionnaires were very important to gain valuable insight about the way the users felt interacting with the agent. In this process the users made also comments about their answers, which helped in having a more clear idea about the test procedure and about where they were basing their answers. Next we go through all three questions and examine the results we had from the users' answers. Some clarifications are also given concerning the answers that were given from the users.

The first question concerning the interaction with the portable agent gave as the following results.

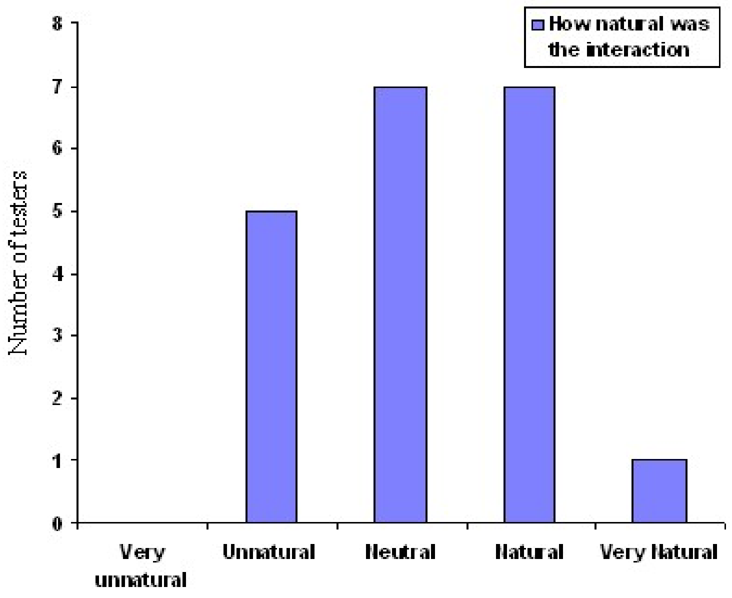

Figure 14. Naturalness of interaction

As we can see in the results, the test users' opinions were mostly on the natural side although we had 5 subjects that thought it was not natural. Most of the people that felt that the interaction was not natural were having problems with the dialogs that the agent was running. As we explained earlier, the dialog system was neither social nor intelligent enough to be able to have a natural dialog, as the expert users would expect him to have. This was causing them to have a negative opinion about the naturalness of the interaction.

The majority of the test users though, a total of 15 subjects, had a rather positive opinion about how natural the interaction with the portable agent was. This was mainly because they did not encounter any serious problems with the dialog system and the dialog was flowing in a smooth way allowing them to focus on the talking head agent.

The second question about the freedom in the movements in comparison with the desktop computer gave us the following results.

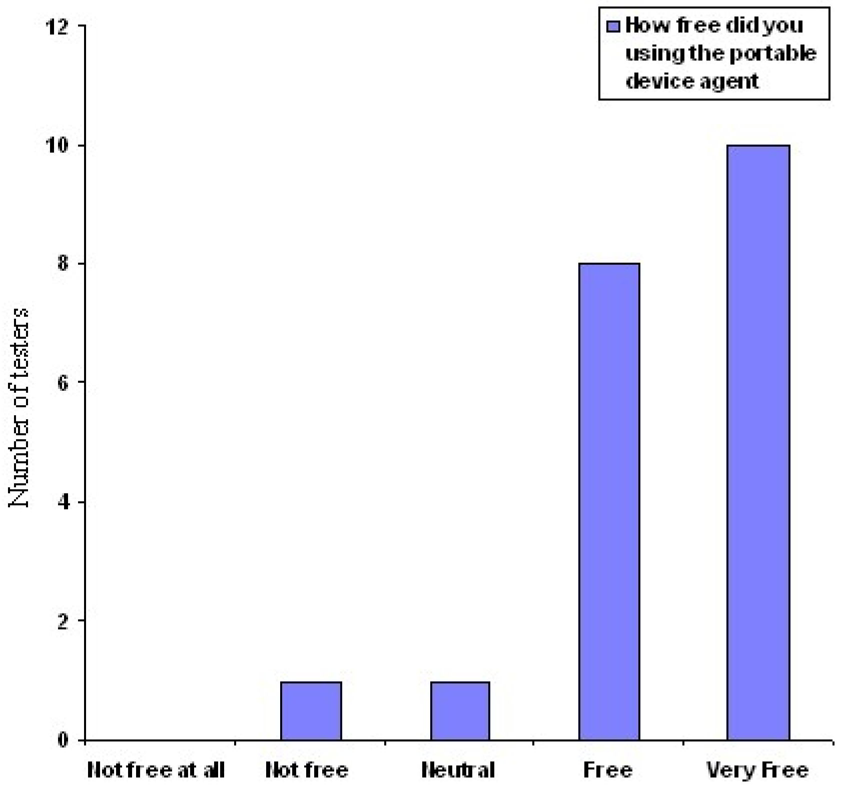

Figure 15. Freedom in interaction

It was more than clear that the opinion of the test users, about how free they felt using this agent instead of using the desktop computer counterpart, were very positive. A total of 18 subjects thought that the application was giving them enough freedom to move around and still be able to interact with the agent. The few subjects that were not that excited by the idea explained that, they did not feel the need to use a portable device like that for their calendar application. These results are very important in our research and they were giving a very clear answer to the initial research question that we set up to examine.

Finally the answers concerning the last question about the use of this talking head agent as an everyday tool for questioning the calendar are shown below.

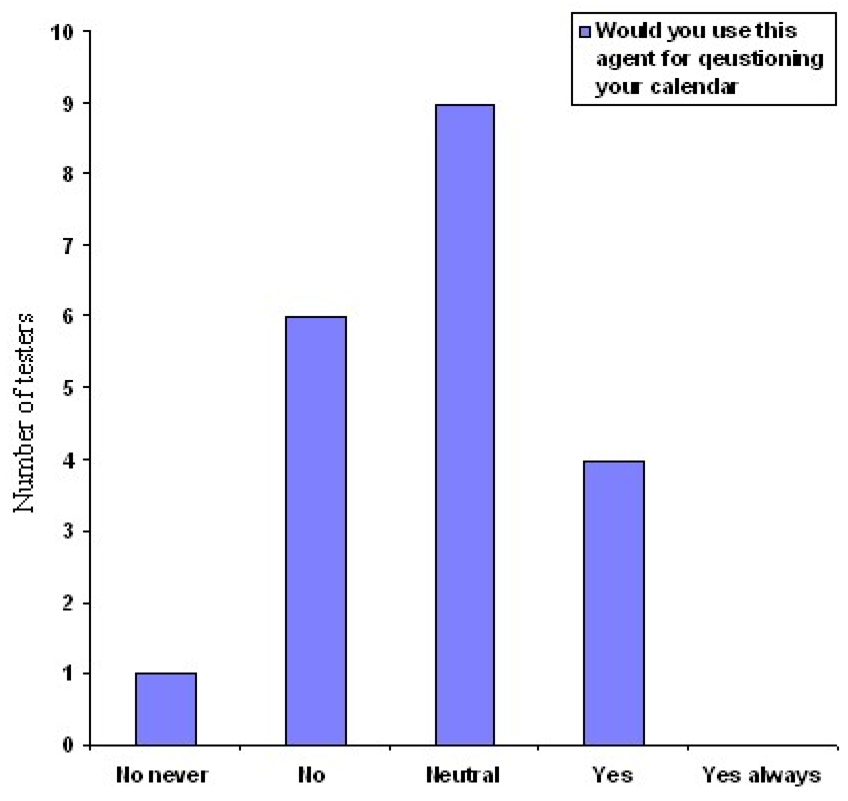

Figure 16. Use of the agent for the calendar application

The answers to these questions were at least ambiguous. The test users were rather confused when they were asked if they would use the talking head agent for questioning their calendar. This was rather expected since the users were not belonging to the actual target group whom this application was initially aiming for. The multimodal speech syntheis talking head application is a designed to be a help for hearing impaired persons or elder people with difficulties in hearing. The test users, who tried the agent, were not in need of such a functionality, since they can go along quite sufficiently by either just listening to what the agent says or by manually searching on the calendar. This fact, in combination with the fact that they were not used to an application like this running on the portable device, such as the PDA or a mobile phone device, explains why the majority of the test users where rather negative or neutral in using the agent for questioning the calendar. Many users noted that they would not mind if the agent were there but they would prefer that the calendar would also show up on the screen. What was essential in the end was that we found out that people where not really negative on the talking head feature accompanying the calendar information, but they really did not feel the need for such functionality. This of course is not the case for the target group we are aiming for, as we explained in earlier chapters.

5. Conclusions and Future Work

The goals of this project were:

1. The integration of the animation engine to the portable device
2. Establish connectivity with the server that computes the animation facial parameters (FAPs)
3. Synchronize the audio with the visual drawing of the animated face for multimodal speech synthesis
4. Conduct usability tests on subjects using the talking head on the portable device.
5. Be able to answer the research question of this study.

By making the draw of the animated talking face on the portable device the first requirement was fulfilled. The frame rate, which we managed to reach in the drawing was about 18 frames per second. This performance was enough to give us very good results when we tested the engine with real users. The result was natural enough and none of the test users complained about the quality of the animated talking face.

Connectivity with the server was also implemented using the server client architecture of a wireless network and using TCP/IP transfer protocol. The connection was fast and stable enough to allow the tests to run and the user to be able to interact with a dialog system running on the server.

The third requirement was to synchronize the audio signal with the visual drawing of the animated face, so that the multimodal speech synthesis could be achieved. This requirement was also fulfilled since the sound and visual output where perfectly synchronized and the test users' perception of the speech coming from the agent was not negatively affected.

The usability tests were also conducted and the results were used for answering the research questions of this study.

We proved with the use of this portable device that the interaction of users with the dialog system was facilitated and improved in two directions. The first direction has to do with the fact that the user is not restricted in the screen of the desktop computer to be able to interact with a dialog system, which is brought to him through the animated agent. The results showed that people were very satisfied with the fact that they could move around and still be able to interact with the agent performing tasks related to the dialog system the agent was connected to. The second part concerns the ability of the portable agent to deliver quality information to people who have hearing difficulties because of their age or some impairment. The agent was capable enough to deliver the visual information of speech which is very important for people belonging to these target groups, for them to understand the utterances that are being said. Although we did not test the agent with such subject, the fact that the agent was not affecting our test users' perception of the produced speech allow us to believe that the agent would be a definite help for people of this target group and could deliver the correct and so much needed visual information.

Concerning the next question, if the application could be used in other applications as well, we firmly believe, after the study we conducted, that this portable engine is the first step of a number of applications in the area of multimodal speech synthesis and ubiquitous computing. It is the tool that was needed to bringing the speech technology know how to the ubiquitous computing reality. So the reality of the intelligent context aware agent that would work as a personal butler as we described in the introduction of this study is definitely a challenge for which next efforts should aim for.

The next and final question was about the preference of the portable agent against the desktop agent. We do think that the portable agent could substitute the desktop agent in all the activities of the normal user. The portable agent enables the user to interact with the agent every where in the world as long as wireless connectivity is available. This is a big advantage of the portable device against the desktop version. On the other hand the desktop version still has the advantage of the big screen display, which makes it more easy for users that have difficulties in seeing properly.

The fact that this engine runs on a windows mobile operating system for portable devices makes it a real contender for mobile phone applications, since companies, like Sony-Ericsson, are launching into the market new mobile phone devices that will support this operating system. It would be nice if the agent could be tried in these phone devices and tested with users again since the feeling of the mobile phone is much more familiar to the users than that one of a PDA.

Future work is something that we should also focus on. We must explain that the talking head application, which we just designed and developed within the scope of this master thesis project is just a first demo. We strongly believe that portable agent can become a lot better in all aspects; in terms of frame-draw performance, connectivity and robustness issues, and also in terms of supported functionalities. There are a lot of suggestions that could be made in every one of these aspects.

The improvement of this agent and the connection to different kind of applications is an interesting and open challenge for future studies, and the real strengths of this first animated talking head agent are yet to be revealed.

References

[1] Beskow, J. (2003), *Talking Heads – Models and Applications for Multimodal Speech Synthesis*. Diss. Department of Speech, Music and Hearing, KTH Stockholm 2003.

[2] Agelfors E, Beskow J, Dahlquist M, Granström B, Lundeberg M, Spens KE, Öhman T (1998). Synthetic faces as a lipreading support. In Proceedings of the International Conference on Spoken Language Processing, Sydney, Australia, 7: 3047-50.

[3] Fisher CG (1968). Confusions among visually perceived consonants. Journal of Speech and Hearing Research, 11, 796-804

[4] Beskow, J. (1997), Animation of Talking Agents. In: Proceedings of AVSP'97, ESCA Workshop on Audio-Visual Speech Processing, pp. 149-152. Rhodes, Greece, September 1997.

[5] Monami Project (Mainstreaming on Ambient Intelligence), http://www.monami.info/, February 2008.

[6] Parke, F.I., Waters, K. (1996), Computer Facial Animation. Wellesley, Massachusetts: A K Peters.

[7] Pandzic, I., Forchheimer, R. (2002b), The Origins of the MPEG-4 Facial Animation Standard. In: Pandzic, I., Forchheimer, R.(Eds.), MPEG-4 Facial Animation – The Standard, Implementation and Applications, Chichester, West Sussex, England: John Wiley & Sons. pp. 3-13.

[8] Nordenberg, M (2005). Creating Robust MPEG-4 Face Models. Unpublished, Department of Speech, Music and Hearing, KTH, Stockholm, Sweden.

[9] Ostermann, J. (2002). Face Animation in MPEG-4. In: Pandzic, I., Forchheimer, R.(Eds.), MPEG-4 Facial Animation – The Standard, Implementation and Applications, Chichester, West Sussex, England: John Wiley & Sons. pp. 17-56.

[10] Escher, M., Pandzic, I., Magnenat Thalmann, N. (1998), Facial Deformations for MPEG-4. In: Computer Animation, pp. 56, 1998.

[11] Beskow, J., & Nordenberg, M. (2005). Data-driven Synthesis of Expressive Visual Speech using an MPEG-4 Talking Head. In Proceedings of Interspeech 2005. Lisbon.

[12] Dell Axim X51v Owner's Manual, http://support.dell.com/ February 2008.

[13] The OpenGL Graphics System: A Specification (Version 1.5),Mark Segal, Kurt Akeley, Editor (version 1.1): Chris Frazier, Editor (versions 1.2, 1.2.1, 1.3, 1.4, 1.5): Jon Leech.

[14] http://www.khronos.org/, February 2008

[15] OpenGL R ES, Common/Common-Lite Profile Specification,Version 1.1.10 (Difference Specification) (Annotated),Editor (version 1.0): David Blythe, Editor (version 1.1): Aaftab Munshi, April 4, 2007

[16] Khronos Native Platform Graphics Interface, (EGL Version 1.3), (December 4, 2006), Jon Leech.

[17] http://www.imgtec.com/, February 2008.

[18] OpenGL ES SDK User Guide, Imagination Technologies Ltd., Feb 2007.

References

[19] OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.1 (2rd Edition), Mason Woo, Jackie Neider, Tom Davis, Dave Shreiner, OpenGL Architecture Review Board, December 1998.

[20] Designing Embedded Hardware, Second Edition, John Catsoulis, Second Edition May 2005

[21] www.dell.com Products/Handhelds &Tablets, 2008

Appendix A - Important Differences Specifications between OpenGL and OpenGL ES

In this paragraph we will try to outline the OpenGL ES Common-Lite profiles in comparison to the OpenGL framework. we will try to provide a concise description of the differences between a full OpenGL renderer and the Common-Lite renderer, always in relation to the OpenGL specification.

Concerning syntax: The OpenGL command and type naming conventions are retained identically. The new types fixed and clampx are added with the corresponding command suffix, ’x’. Commands using the suffixes for the types **byte, ubyte, short, and ushort are not supported,** except for glColor4ub. The type **double and all double-precision commands are eliminated**. The result is that the Common-Lite profile uses only the suffixes ’i’ and ’x’.

Concerning basic GL Operation: The basic command operation remains identical to OpenGL 1.5. The major differences from the OpenGL 1.5 pipeline are that commands **cannot be placed in a display list**; there is no polynomial function evaluation stage; and blocks of fragments cannot be sent directly to the individual fragment operations.

Concerning GL Errors: The full OpenGL error detection behavior is retained.

Concerning Begin/End Paradigm: The Common-Lite profile draws geometric objects exclusively using vertex arrays. Associated colors, normals, and texture coordinates are specified using vertex arrays. The associated auxiliary values for color, normal, and texture coordinate can also be set using a small subset of the associated attribute specification commands (Color4, Normal3, and MultiTexCoord4). Since the commands Begin and End are not supported, no internal state indicating the begin/end state is maintained. The QUADS, QUAD STRIP, and POLYGON primitives are not supported.Color index rendering is not supported. Edge flags are not supported.

Appendix A

Concerning Buffer Objects: It is sometimes desirable to store frequently used client data, such as vertex array data in high-performance server memory GL buffer objects provide a mechanism that clients can use to allocate, initialize and render from memory. Buffer objects can be used to store vertex array and element index data.

Concerning Rectangles: The commands for directly specifying rectangles are not supported.

Concerning Coordinate Transforamations: The full transformation pipeline is supported with the following exceptions: no support for specification of double-precision matrices and transformation parameters; no support for the transpose form of the LoadMatrix and MultMatrix commands; no support for COLOR matrix; and no support for texture coordinate generation. The double-precision only commands DepthRange, Frustum, and Ortho are replaced with single-precision or fixed-point variants from the OES single precision and OES fixed point extensions. The minimum depth of the MODELVIEW matrix stack is changed from 32 to 16.

Concerning Colors and Coloring: Lighting model is supported with the following exceptions: no support for the color index lighting, secondary color, different front and back materials, local viewer, or color material mode other than AMBIENT _AND _DIFFUSE. Directional, positional, and spot lights are all supported. An implementation must support a minimum of 8 lights. The Material command cannot independently change the front and back face properties, so the result is that materials always have the same front and back properties. Two-sided lighting is supported, though the front and back material properties used in the lighting computation will also be equal. The ColorMaterial command is not supported, so the color material mode cannot be changed from the default AMBIENT _AND _DIFFUSE mode, though COLOR_MATERIAL can be enabled in this mode. Neither local viewing computations nor separate specular color computation can be enabled using the LightModel command, therefore only the OpenGL default infinite viewer and single color computational models are supported. Smooth and flat shading are fully supported for all primitives.

Concerning Points and Lines: Aliased and antialiased points and lines are fully supported. Line stippling is not supported.

Concerning Polygons: Polygonal geometry support is reduced to triangle strips, triangle fans and independent triangles. All rasterization modes are supported except for point and line PolygonMode and antialiased polygons using polygon smooth. Depth offset is supported in FILL mode only.

Concerning Bitmaps, Pixel Rectangles and Drawing, Reading, and Copying Pixels: No support for bitmap images and directly drawing pixel rectangles is included. Limited PixelStore support is retained to allow different pack alignments for ReadPixels and unpack alignments for TexImage2D. DrawPixels, PixelTransfer modes and PixelZoom are not supported. The Imaging subset is not supported. Only ReadPixels is supported with the following exceptions: the depth and stencil buffers cannot be read from and the number of format and type combinations for ReadPixels is severely restricted.

Concerning Texturing: we did not make use of Textures, so we will not expand to this functionality. OpenGL ES 1.1 requires a minimum of two texture units to be supported. 1D textures, 3D textures, and cube maps are not supported.

Concerning Per-Fragment Operations: Although we do not make use of these features, all OpenGL per-fragment operations are supported, except for color index related operations and certain advanced blending features (BlendColor, BlendEquation, and blend squaring). Depth and stencil operations are supported, but a selected config is not required to include a depth or stencil buffer.

Appendix A

Concerning Whole Framebuffer Operations: All whole framebuffer operations are supported except for color index related operations, drawing to different color buffers, and accumulation buffer.

Concerning Evaluators, Selections, Feedback, Display Lists, Flush and Finish, Hints: Selections, Feedback and Display Lists functionalities are not supported unlike Flush and Finish that is supported. Hints are retained except for the hints relating to the unsupported polygon smoothing and compression of textures (including retrieving compressed textures) features.

Concerning Client and server attribute stacks: They are not supported by the Common-Lite profile; consequently, the commands PushAttrib, PopAttrib, PushClientAttrib, and PopClientAttrib are not supported. Gets are supported by the profile to allow an application to save and restore dynamic state. In the specification of the OpenGL Common-Lite profile there is an extended table of the supported state variables. []

Appendix B - The follow-up questionnaire with the testers' opinions.

Personal information.

Sex: Male Female
Age:

Questions around the test that which was carried out:

1. How natural did you feel the interaction with the portable agent was?

Very unnatural. Very natural

1 2 3 4 5

2. Did you feel more free to interact with an agent that is not fixed on a desktop computer?

Not at all. A lot more free.

1 2 3 4 5

3. Would you use the talking head agent as an everyday tool for questioning your calendar?

Never. Always.

1 2 3 4 5

Any comments you would like to add:

___

___

___