Multimedia SMS Reading in Mobile Phone

R. Talafová, G. Rozinaj, J. Čepko, and J. Vrabec

***Abstract*—**This paper is devoted to the speech synthesis and development of a speech synthesizer for a mobile cell phone. The presented results are a part of a more complex project for multimedia reading of short messages (SMS) on the mobile phone. After receiving the SMS a talking head based on a sender's photo will appear on the screen and animate the reading while the speech will be synthesized in parallel. This work further analyzes an implementation of the speech synthesizer – this means loading the database, synthesis, creating the annotation file and creating the output sound signal. The final synthesized speech utterance is played together with the face animation of the talking human face

***Keywords*—**diphones, mobile phone, speech synthesis

I. Introduction

With the rising of the hardware power of mobile devices (including phones) the human computer interaction (HCI) has started to develop quite a lot of new mobile applications in the last decade. The goal of this effort is to integrate multiple input-output modalities into one interface. In this interface the speech modality is processed by automatic speech recognition (ASR) and text-to-speech (TTS) synthesis systems. Depending on the implemented complexity couple of modes of speech interaction has emerged: embedded systems that run locally on the mobile devices [1][2][3], client-server systems that put most of the load on a remote server [4][5], or those that are capable of both these modes [6]. Some examples of speech enabled applications that make use of these modes are: personal speech assistants [1], mobile assistants for the blind [2], working city guides and navigation systems [6], web-based multimodal services [5], applications for documenting traffic accident reports [3], or speech-based inventory and time management services [4].

Although that the progress in telecommunications and in mobile phones during the last years is huge, the speed and the memory of mobile phones will still be less powerful than those of standalone PCs. That is why a couple of works emerged, that were handling the problem of adapting up-to-date TTS systems for mobile devices [7][8][9]. The main goal of these works is to balance the demands of achieved speech quality against CPU and memory load.

Manuscript received January 31, 2007. This work was supported by Slovak Grant Agency under grants VEGA 1/3110/06, AV 123/06, and AV020/07.

G. Rozinaj is with the Slovak University of Technology in Bratislava, Faculty of Electrical Engineering and Information Technology, Department of Telecommunications, Ilkovičova 3, 81219 Bratislava, Slovakia (phone: +421-2-68279414; fax: +421-2-68279601; e-mail: gregor.rozinaj@stuba.sk).

R. Talafová , J. Čepko and J. Vrabec are with the same institution as G. Rozinaj (e-mail: renata.talafova@gmail.com, jozef.cepko@stuba.sk, vrabec@ktl.elf.stuba.sk).

R. Talafová is also with Siemens-PSE, Dúbravská cesta 4, 84537 Bratislava, Slovakia.

The purpose of this paper has been to create an application for reading short messages in cell phones. The idea of such an application is not new and there have been some systems developed already, working both as embedded [10][11] and client-server based [12]. However, the application presented here is a combination of the speech synthesis and the face animation. Speech part is represented by speech synthesizer and animation is represented by 3D model of human face. After the SMS has been received, application starts to synthesize the text and the output acoustic signal is played together with the animation. This paper describes a concrete type of service and its implementation in mobile phone.

II. Face animation

The human face can be modeled and animated by many methods. For choosing the best method we need to define requirements for module of animation for mobile phone. The list of requirements for module of animation is:

- The animation has to by continuous and performance optimized, because of running on mobile phone.
- The animation has to run synchronous with synthesized.
- The quality of the animation should be the best, final animation should be as similar to reality as possible.

There two basic approaches: two-dimensional and three-dimensional and real time animation and forward calculated animation. Current mobile phones are fast enough for real time three-dimensional animation, so we decided use this approach in our work.

The real time animation allows users an interactive intervention to animation and reduces the time necessary for the preparation of the animation. The disadvantage of this method is its quality of an image. The calculation of each image should not be longer than approximately 0.05s, because there should be at least 20 images per second (20 FPS – frames per second).

The advantage of this method is evident. Quality of three-dimensional modeling is better than the quality of two-dimensional approach. This animation models the reality more naturally. In our animation it is possible to turn around the human face and the model is illuminated according to a real geometry.

There are many various techniques for modeling human face in space, for example polygonal modeling, modeling by parametric areas, sub-division modeling.

For the face animation following methods are known: interpolation, parameterization, simulation of muscles, etc.

A. *Models and visemes*

The foundations of the model of the human face and its visemes that we use, was taken from the project FaceGen [14]. Both the model and the visemes are in the object (OBJ) format, which means the files include not only the model, but also the texture which belongs to it.

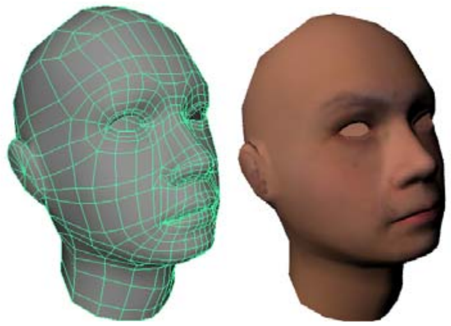


Fig. 1 neutral face model and face texture

OBJ files are stored in the resource part of the application (in the jar file, the file produced by Java in a specific format). Every time the application is launched, each OBJ file is read and its nodes and polygons are stored in the main memory. So we already have the whole face available. The eyes are represented by two separate models. After all objects are loaded, they are matched with each other.

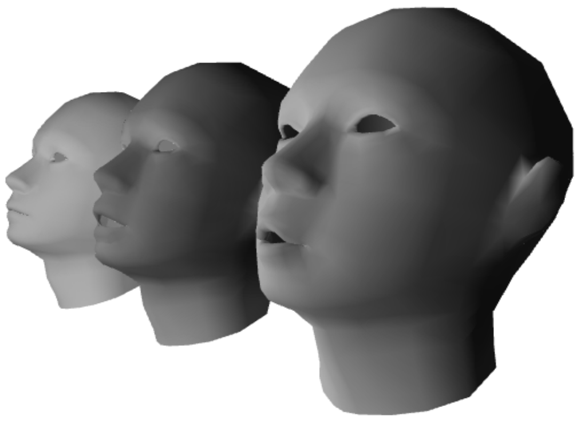


Fig. 2 face visemes

The viseme, we are referring to is a deformed model of the face. This is not just any kind of the deformation; it is the deformation as if the face was saying the given phoneme. The model of the visemes still has the same number of nodes, which have the same numbering scheme and are connected with the same lines as the neutral model. The only change is the position of the nodes. Because of this simplification we are able to perform an easy interpolation of the nodes and the orthogonal.

The animation itself is realized by the before mentioned viseme interpolation. The neutral model is read from the file (together with the other models and visemes). The interpolation is performed between these models.

The animation is also based on a time (this is a real time type of animation). In the beginning the time is set to zero. The time is counted in each step of the animation and it also determines how the model is deformed. The deformation is also based on a factor, which is a number between 0 and 1. Number 1 means a 100% match (or correlation) with the second viseme and number 0 means 100% correlation with the first viseme. The nodes in the animation for the factors between 0 and 1 are determined by the interpolation.

B. *Interpolation of visemes*

We use interpolation in order to get the positions of the nodes and the normals of our model in a certain time. To simplify things, we are looking only for the interpolation between the two visemes, and not for a function that would represent the best approximation sequence.

The interpolation will be done in the form of function C(x):

$$C(x) = f(x) * A + (1 - f(x)) * B \text{ , } 0 \leq x < 1 \qquad (1)$$

where:
C(x) – is the position of the node we want to get by interpolation,
A – is the coordinate of the previous model,
B – is the coordinate of the model we want to get,
f(x) – is the interpolation function, while $D(f) = <0; 1>$ and $H(f) = <0; 1>$,
x – is the parameter which represents normalized time, where.
$x = (t-t_1) / (t_2-t_1)$,
t – is the time,
t_1 – is the time of beginning of the interpolation, where $C(x) = A$,
t_2 – is the time of the end of the interpolation, where $C(x) = B$.

The function *f(x)* has been empirically chosen. It is a special function that meets certain needed criteria, such as the animation should not be linear and has to look smooth.

Following considerations have to be also true: $f(0)=0$ and $f(1)=1$. There are also other requirements, for example the derivation in 0 and 1 has to be equal zero.

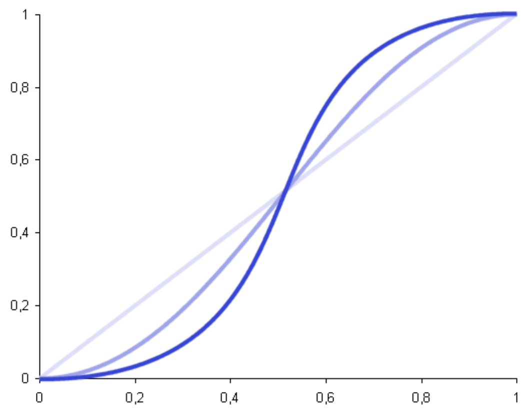


Fig. 3 interpolating function, required non-linearity

These conditions make sure, that the transition between visemes will be smooth and non linear. It will be changing fast from the beginning and gradually slowing, until it reaches the desired viseme. We could also use a linear interpolation function, however this would cause a very uniform face motion and the face movement would not be very natural.

Fig. 3 shows the nonlinearity of the interpolation function *f(x)*. The function which meets these criteria is for example:

$$f(x) = (-\cos(x*\pi) + 1) / 2. \qquad (2)$$

III. Speech synthesis and synthesizer

The idea that a machine could generate speech has been in minds of people for some time already, but the realization of such machines has really been practical only within the last 50 years. The rise of concatenative synthesis began in the 70s, and has largely become practical as large-scale electronic storage has become cheap and robust. Before 1980, research in speech synthesis was limited to the large laboratories that could afford to invest the time and money for hardware. By the mid-80s, more labs and universities started to join in as the cost of the hardware dropped. By the late eighties, purely software synthesizers became feasible; however the speech quality was still decidedly inhuman. And although we are now at the stage were talking computers are with us, there is still a great deal of work to be done.

Speech synthesis means creating human-like speech using a machine, which is known as speech synthesizer.

There are several types of these synthesizers, but each is made to do the same: to reproduce the given text in the clearest and most understandable manner. There are four basic approaches:

- Synthesis using units
- Formant synthesis
- Articulation synthesis
- HMM synthesis

On Fig. 4 we can see a block diagram of a general synthesizer. Of course this diagram is simplified to our needs and some elements (such as a feedback found in some learning synthesizers, etc.) are omitted.

However, virtually every synthesizer consists of following parts:

- Entry text input, analysis
- Preprocessing
- Synthesis
- Post processing
- The synthesized speech

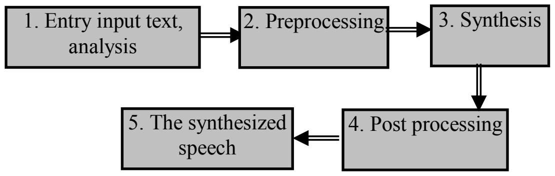


Fig. 4 block diagram of a general synthesizer

In our implementation of the speech synthesizer for mobile phone we have decided to use the diphone synthesis and naturally the diphone database. The major advantage of this solution is in its size. Slovak language can be sufficiently covered by only 1550 diphones and this makes the size of the solution very reasonable (especially compared to other approaches).

IV. Diphone synthesis

A diphone consists of two following phonemes. The boundaries of the diphone are in the middle of these sounds. This means, that a diphone length is not double, as one might suspect, but approximately the same as length of one phoneme. The advantage of using diphones and not phonemes is that they better represent the change between sounds, because their boundaries are in the middle of sounds where the characteristic time curve is stable.

In the theory, the number of diphones is the square of number of phonemes (all combinations of two phonemes is a square). However, the real number is lower, because the particular language does not use, or does not utilize all of them. We can get the real number of diphones by closely studying the language [13].

A. Creating database

Before starting the diphone synthesis, the diphone database has to be created. This database consists of real speech recordings which are broken into small parts – diphones. There are two options how to create and record this database. Either to choose words, which will cover all diphones from a dictionary, or use some other approach. These words need not to have a meaning; the aim is to have the smallest possible set of recordings.

Of course, the better is the recording quality, the better is the speech output. Therefore it is advisable to use a studio quality recording. Usually, the recording can not be done at once, because the narrator would get tired and the recording quality would deteriorate. Therefore it is important to ensure the same conditions (sound reflections, time of the day, hardware, etc.) during the sessions.

The choice of the narrator is also very important. Voice professionals are the best choice. The appropriate people for this job usually come from television or radio environment – someone who earns living by speaking.

As already mentioned, the database is not recorded at once, because the narrator would get tired and the recording quality would deteriorate.

The recording has to be further processed to get the final database. It has to be replayed to check for any errors. The next step is a process called labeling. This means that the diphone boundaries have to be set and marked down; otherwise the machine would not know where to find them.

There are several labeling methods. The easiest and slowest one is manual labeling. From automatic or semi automatic methods of labeling, we mention labeling based on DTW and acoustic model labeling.

The labeling should also include equalizing the recording samples. We do the amplitude equalizing because it is not desirable to have the volume change in the middle of a word – that is what would happen if two diphones have different sound level. They have to be attached at the same amplitude level.

The amplitude equalizing is not the only one technique we use for quality enhancement of the synthesized speech. We consider also a phase equalizing, which is even more important. All the boundaries have to be in approximately same phase. If the phase would be opposite, we could experience lot of noise and various acoustic clicks [13].

The results of these steps are an acoustic database – in our case in a WAV format. An index file is a natural part of the database. It is actually a list of diphones with their boundaries in WAV file.

B. *Implementation of diphone synthesizer*

The design of the synthesizer is shown on Fig. 5. In this figure the principle of the speech synthesis has been described in a very simple form, and it shows how the synthesizer works.

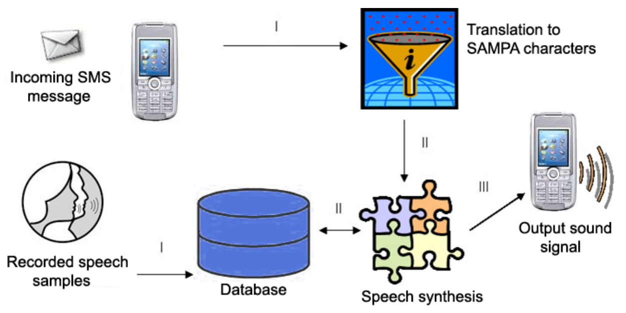


Fig. 5 synthesizer design

The input text has to be synthesized into the speech. But at first it has to be broken down into so called SAMPA alphabet. SAMPA (Speech Assessment Methods Phonetic Alphabet) is a phonetic translation which uses only printable ASCII characters. In the first step all characters are retyped to SAMPA. In the second step the result from the first step is retyped according to all rules for pronunciation for Slovak language. These rules were taken from [15]. For example word *košeľa* will be written as „*ko(S)E(L)a*“.

Before we get to the synthesis, the database has to be loaded. In our case, this is done right after the start of the application. As the database is in WAV format, the program has to find the data part and save it in the memory. It also loads the index file with the list of phonemes and diphones and its boundaries. In the tables (Table I and Table II) are examples of the part of index file with phonemes and also diphones.

Explanation to the Table I:

- *SAMPA* phoneme – the name of the phoneme from the index file. The file contains all phonemes in the same order like they are stored in the database.
- *Offset* – the value represents a start of a group of diphones that have the same first phoneme.
- *Begin* – represents the first sample in database for this phoneme.
- *End* – represents the last sample in database for this phoneme.
- *Middle* – represents the middle sample in database for this phoneme.

TABLE I
EXAMPLE OF INDEX FILE FOR PHONEMES

SAMPA phoneme	Offset	Begin	End	Middle
a~	277423	51098	55381	53259
O	202295	33532	34505	34041
U_^	231356	40543	42386	41450

TABLE II
EXAMPLE OF INDEX FILE FOR DIPHONES

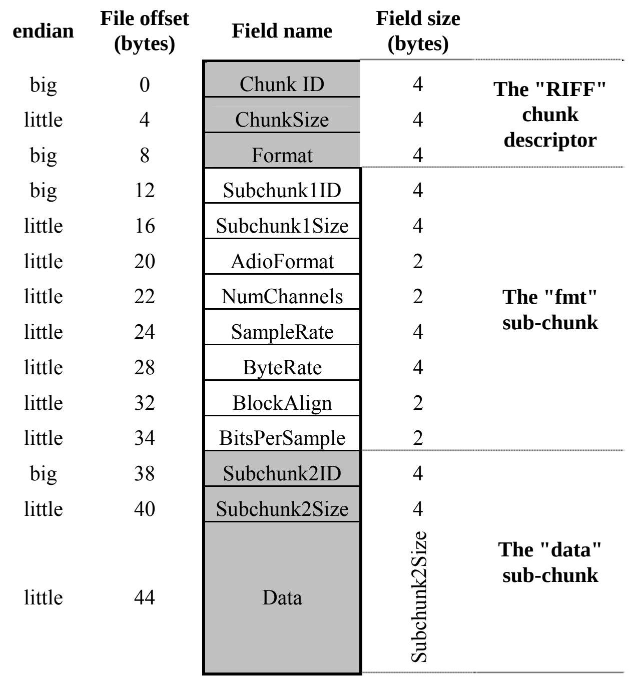
SAMPA diphone	Begin	End
O _	2326	4334
O x	4335	5771
U_^ t	3561	4505

Explanation to the Table II:

- *SAMPA diphone* – the name of the diphone from the index file. The file contains all diphones in the same order like they are stored in the database.
- *Begin* – represents a start of the diphone in the group of diphones starting with the same phoneme (it’s like local number). It means if we want to know the number where the diphone begins in the database (global), we have to add this value and the value of offset from the Table I. For example the phoneme U_^ starts on a sample number 40543 (Table I). But if we want to know where the group of diphones is starting with U_^, we have to go to the sample number 231356 (offset). Than according second letter in the diphone the relevant samples are found. For the diphone U_^ t we get values like that: beginning add offset from Table I and begin from Table II (231356+3561 = 234917).
- *End* – represents where the diphone in the database ends in the group of diphones starting with the same phoneme (it is like a local number). The global value for the end we get in the same way like for the beginning, it means the adding offset from Table I and end from Table II (231356+4505 = 235861).

The algorithm of the synthesis is very simple, because the text is in SAMPA and the database is loaded. For each diphone we check if it is in the database. If it is so, samples are taken from the database and stored. If diphone is not in database, it is replaced with the two phonemes from which a diphone is composed. For each phoneme we do not take all samples, just half of them. It is because like we said before, the duration of the diphone should be approximately of the same length as the phoneme. For example, for the diphone En (we suppose it is not in the database) samples for E are taken from “*middle*” to “*end*”, samples for n are taken from “*begin*” to “*middle*”. We do this for the whole text which should be synthesized. When the program finishes going through the whole text, and audio output is created and written in a WAV file.

The WAV file consists of two basic parts, a header and a data segment. The header consists of so called RIFF descriptor, which contains general file information and a part called FMT. This part describes the format of the data part. The format of a WAV file can be seen on Fig. 6.



endian	File offset (bytes)	Field name	Field size (bytes)	
big	0	Chunk ID	4	The "RIFF" chunk descriptor
little	4	ChunkSize	4	
big	8	Format	4	
big	12	Subchunk1ID	4	The "fmt" sub-chunk
little	16	Subchunk1Size	4	
little	20	AdioFormat	2	
little	22	NumChannels	2	
little	24	SampleRate	4	
little	28	ByteRate	4	
little	32	BlockAlign	2	
little	34	BitsPerSample	2	
big	38	Subchunk2ID	4	The "data" sub-chunk
little	40	Subchunk2Size	4	
little	44	Data	Subchunk2Size	

Fig. 6 Wave file format

The fields of the WAVE format have the following meaning:
ChunkID – contains ASCII characters of RIFF,
ChunkSize – the length of the file except first two fields (size – 8 bytes),
Format – contains ASCII characters of WAVE,
Subchunk1ID – contains ASCII characters of FMT,
Subchunk1Size – 16 for PCM,
AudioFormat – 1 for PCM, linear quantization, other values for non linear compression,
NumChannels – 1 mono, 2 stereo,
SampleRate – sample frequency (Hz) – 16 000,
ByteRate –
*SampleRate*NumChannels*BitsPerSample*/8 – 32000,
BlockAlign – 2,
BitsPerSample – number of bits per sample – 16,
Subchunk2ID – contains ASCII characters for data,
Subchunk2Size –
size of sub chunk*, Samples*NumChannels*BitsPerSample*/8,
Data – values of stored data from -32 768 to 32 767.

After creating the output WAVE file, this file can be played on the mobile phone.

C. *Synchronization with face animation*

The synthesized text has to be synchronized with the speaking face.

We use an annotation file (*.ano) which is a simple text file containing phonemes with its time marks (boundaries). The values are determined individually for each phoneme or the phoneme from the index file. From this file we take numbers of samples for the beginning and the end and using the sampling frequency determine the start time and the length of the maximum.

In the table (Table III) is a sample of this file for the Slovak word „*kráľov*“.

TABLE III
EXAMPLE OF ANOTATION FILE

SAMPA	Internal phoneme	Begin [s]	Rise time [s]	Maxim length [s]	Weight
k	k	0.4560	0.0298	0.0298	0.5000
r	r	0.5156	0.0647	0.0647	0.5000
a:	a	0.6451	0.0194	0.0389	0.5000
L	l	0.7034	0.0192	0.0192	0.5000
O	o	0.7417	0.0157	0.0157	0.5000
U_^	u	0.7731	0.0480	0.0480	0.5000

Explanation to the Table III:

- *Internal phoneme* – represents the character of the viseme (a shape of the mouth for the face animation) which the application uses for selection. It is made by a conversion from SAMPA using a mapping table of SAMPA characters to visems.
- *Begin* – time when the viseme starts
- *Rise time* – time period, when the mouth or the phoneme are opened to maximum
- *Maxim length* – a duration of phoneme’s maximum. Visually, this is the time, when the mouth does not move (or the movement is not significant)
- *Weight* – from <0,1>. The higher the weight, the more visible the viseme [16].

V. EVALUATION

In order to evaluate the implemented system two aspects must be taken into account; the achieved quality of the talking head, and the computational load of the synthesis process on the mobile phone.

Since at SMS reading, when user cannot use the visual modality, the quality of the animation is not as crucial as the quality of the synthesized speech, we have used standard speech MOS tests to evaluate the quality of the talking head. The MOS results are obtained by averaging the results of a set of subjective tests where a number of listeners rate the heard speech quality. The MOS ranges from 1 (worst) to 5 (best). Unfortunately, the results of these tests are not completed so far, and thus are not suitable for publication, however the temporal results seems promising. More than seventy amateur evaluators generated 3.3878 MOS.

The computational footprint of the application may be viewed from two sides: its size and its speed. Since the limitations on memory space have greatly diminished in the last years, we were able to improve the quality by enhancing the speech coding using PCM coded 16 kHz/16 bit sampled signal. The comparison of the size of the database with other available TTS for mobile devices is shown in Table IV. It may be seen that our database is the largest one; however, it is due to its phoneme/diphone content and higher quality signal coding (e.g. PCM instead of A-law).

TABLE IV
SIZE OF DATABASES OF SOME MOBILE TTS SYSTEMS

Database	**Storage size**
Hungarian TTS [7] 11 kHz /8 bit, A-law (diphone)	1.5 MB
Flite (mobile Festival) [9] 8 kHz (diphone)	< 4MB
LAIPTTS [8] (Mons (Belgium) diphone database)	4.7MB
Presented synthesizer 16 kHz /16 bit, PCM (phoneme/diphone)	6.16MB

Although that in [7] is argued that it is favorable not to load the database into the memory, [8] expects diphone processing to be accelerated if the database is taken into RAM. Having speed as a priority and following the [8] assumption we load the database into the memory at the start of the application. We have tested and compared the speed of our synthesis in a mobile phone and the results are given in Table V. The leading synthesis' times of our synthesizer are not just results of faster CPU, but they are consequence of the database load into RAM, and the direct PCM coding. Plus, it must be emphasized that the results for our synthesizer were generated in an emulator; the real phone performs even faster, however, in this case it is hard to measure the delay with a high accuracy. Similarly as in [7] and [8], we have observed that noticeable delays occur when a short sentence precedes a long one

TABLE V
COMPARISON OF SYNTHESIS TIMES OF SOME MOBILE TTS SYSTEMS (IN SECONDS PER SENTENCE). SHORT SENTENCE HAS 50-100 CHARACTERS, LONG SENTENCE HAS 100-400 CHARACTERS

System	**short sentences**	**long sentences**
Hungarian TTS [7] Nokia 6680 (220MHz)	0.15	0.467
Presented synthesizer Nokia N95 (330MHz)	0.171	0.271
LAIPTTS [8] PPC (80 MHz)	**approx. 7 sec. long sentences** < 3	.

VI. CONCLUSION

In this paper we presented a Slovak language speech synthesis application for a mobile phone. The application is combined with a speaking face animation and presents a possible future development for future mobile communication.

The type of the created application is Java Midlet, very similar to an applet class, but assigned for cell phones. Application is installed from JAR file. JAR is an archive and contains compiled classes.

The application can be very easy modified for needs of a mobile phone. This will be necessary for older types of mobile phones, which have limited memory for applications - in this case the database is loaded like external file, and it is not a part of the JAR file. This philosophy results in a smart modification of the database in case of any required changes.

In the future, with the expansion of the mobile phone memory, the database can be extended, because the present database contains only the most used diphones of Slovak language. With higher memory space we shall be able to cover not only all diphones in the database but to add some more frequent triphones, as well. Adding some triphones may solve the problem of unnatural sounds in the concatenation of problematic pairs of diphones. That can improve the quality of output acoustic signal; the speech will be more fluent and natural.

REFERENCES

[1] Comerford, L., Frank, D., Gopalakrishnan, P., Gopinath, R., Sedivy, J. The IBM Personal Speech Assistant, International Conference on Acoustics, Speech, and Signal Processing, 2001. Vol. 1, pp. 1-4. Utah. 2001.

[2] Gaudissart, V., Ferreira, S., Thillou, C. SYPOLE: Mobile Reading Assistant for Blind People, Proc. of the 9th SPECOM 2004. St. Petersburg, Russia. 2004.

[3] Dusan, S., Gadbois, G., J., Flanagan, J. Multimodal Interaction on PDA's Integrating Speech and Pen Inputs, EUROSPEECH 2003. pp. 2225-2228. Geneva, Switzerland. 2003.

[4] Kondratova, I. Speech-enabled Mobile Field Applications, Internet and Multimedia Systems and Applications, IMSA 2004. Hawaii, USA. 2004.

[5] Roessler, H., et al. Multimodal interaction for mobile environments. In Proc. of International Workshop on Information Presentation and Natural Multimodal Dialogue. Verona, Italy. 2001.

[6] Johnston, M., Bangalore, S., Vasireddy. G. MATCH: Multimodal Access To City Help, in Workshop on Automatic Speech Recognition and Understanding. Madonna di Campiglio, Italy. 2001.

[7] Tóth, B., Németh, G. Challenges of Creating Multimodal Interfaces on Mobile Devices, Proceedings of ELMAR-2007. pp. 171-174. Zadar, Croatia. 2007.

[8] Keller, E. Simplification of TTS Architecture vs. Operational Quality. Proceedings of EUROSPEECH '97. Paper 735. Rhodes, Greece. 1997.

[9] Black, A., Lenzo, K. Flite: A Small Fast Run-Time Synthesis Engine, 4th ISCA Speech Synthesis Workshop. pp. 157-162. Scotland. 2001.

[10] Németh, G., Kiss, G., Tóth, B. Cross Platform Solution of Communication and Voice / Graphical User Interface for Mobile Devices in Vehicles, Biennial on DSP for in-Vehicle and Mobile Systems. Portugal. 2005.

[11] Gros, J., Mihelic, F., Pavesic, N., Zganec, M., Mihelic, A., Knez, M., Mercun, A., Skerl, D. The Phonectic SMS Reader, Text, Speech and Dialogue: 4th International Conference, TSD 2001. Železna Ruda, Czech Republic. 2001.

[12] Farrugia, P. J. Text to Speech Technologies for Mobile Telephony Services, MSc. Thesis, Faculty of Science, University of Malta. pp. 160, 2005.

[13] Black, A. W., Lenzo, K. A. Building Synthetic Voices, Language Technologies Institute, Carnegie Mellon University, Retrieved October 12, 2007, from http://festvox.org/bsv.

[14] Vajaš, M., Rozinaj G. Facial Animation During Speech Performance, IASTED International Conference on Communications, Internet and Information Technology (CIIT 2004). St. Thomas, USA. 2004.

[15] Cerňak, M., Rozinaj, G. Forward Masking Phenomenon in Concatenative Speech Synthesis, 4th EURASIP Conference EC-VIP MC'03. pp. 691-694. Zagreb. 2003.

[16] Vajaš, M. Modelling of Talking Face, MSc. thesis, FEI STU Bratislava. 69 pp. 2005. In Slovak.

[17] Michalko, O. Diphone Synthesis of Slovak Speech, MSc. thesis, FEI STU Bratislava. 69 pp. 2007. In Slovak.