



Centrum voor Wiskunde en Informatica

CharToon 2.1 extensions; Expression repertoire and lip sync

Zs.M. Ruttkay, A.D.F. Leli re

Information Systems (INS)

INS-R0016 July 31, 2000

Report INS-R0016
ISSN 1386-3681

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

CWI is the National Research Institute for Mathematics and Computer Science. CWI is part of the Stichting Mathematisch Centrum (SMC), the Dutch foundation for promotion of mathematics and computer science and their applications.

SMC is sponsored by the Netherlands Organization for Scientific Research (NWO). CWI is a member of ERCIM, the European Research Consortium for Informatics and Mathematics.

Copyright © Stichting Mathematisch Centrum
P.O. Box 94079, 1090 GB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

CharToon 2.1 Extensions

Expression Repertoire and Lip Sync

Zsófia Ruttkay, Alban Lelièvre

CWI

P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Email: zsofia.ruttkay@cwi.nl

ABSTRACT

CharToon is a modular system to design and animate 2¹/₂D faces and other graphical objects. This report contains the extensions made for version 2.1, effecting only the Animation Editor module. The new features allow the re-usage of a repertoire of expression snapshots and animations and automatic generation of lip-sync from phoneme and/or viseme sequences.

1998 ACM Computing Classification System: D.2.2, H.5.1, H.5.2, I.3.8, J.5.

Keywords and Phrases: animation, lip-sync, graphical user interface.

Note: CharToon was designed and implemented under the project INS3.4 'Facial Animation'. CharToon is proprietary software of Stichting Mathematisch Centrum and is protected by international copyright laws. An on-line version of this report with colour figures is available from <ftp://ftp.cwi.nl/pub/CWIreports/INS/INS-R0016.ps.Z>. More on-line material, including movies is available from <http://www.cwi.nl/CharToon>.

1. Introduction

CharToon is a modular system to design and animate $2^{1/2}D$ faces and other graphical objects. The three modules of CharToon, **Face Editor**, **Animation Editor** and **Face Player** are meant, respectively, to design faces, to make animations for them, and to play the animations. The architecture of the system, its usage as well as the functionalities of the individual modules are discussed in the manual of the released version 2.0 [5]. We will use in this document concepts and terminology introduced there.

In this document we explain the new facilities in version 2.1. All the new features are of Animation Editor, which make it possible to re-use a repertoire of expressions and animations, and to generate lip sync automatically from a sequence of visemes. We also document a couple of new possibilities to view parameters.

In order to be able to produce a viseme sequence from a phoneme sequence, we provided some auxiliary programs, originally developed to map the Dutch phoneme sequences we received from the IPO institute [2]. However, by editing the content of some files, these programs can be used for mapping to other viseme sets, and can be adapted to mapping different phoneme sets and sequences.

2. Re-using expressions and animations

2.1. Reanimate at a given time

In CharToon 2.0 the **Reanimate** function of Animation Editor, to be activated by the menu `File/Reanimate`, makes it possible to re-use pieces of animations made for a face with a different profile. The reanimation may happen by looking for matching IDs or matching labels in the current profile and the profile of the animation to be re-used (see [5] Chapter 6.3.3 for details). However, when using **Reanimate**, the current animation is first erased, and then the content of the ‘imported’ animation is loaded. Hence, in CharToon 2.0 there is no possibility for incremental usage of pieces of animations, either in time for the same channels or for complementary set of channels.

This situation is improved in CharToon 2.1 by the **Reanimate at time** function, to be activated in Animation Editor by the menu `File/Reanimate at time`. The basic mechanism is the same as of **Reanimate**, namely an ‘.all’ complete animation is loaded to the current one. However, only those APs of the current animation are effected for which a matching AP has been found from the loaded animation. The insertion takes place at the time marker, if it has been set, otherwise at time 0. Only a time interval starting at the insertion time is effected. Namely, if the insertion has to take place at time t and an AP of the animation to be inserted is of length d , then in the corresponding channel of the current animation only KPs between t and $t+d$ get replaced (see Figure 1).

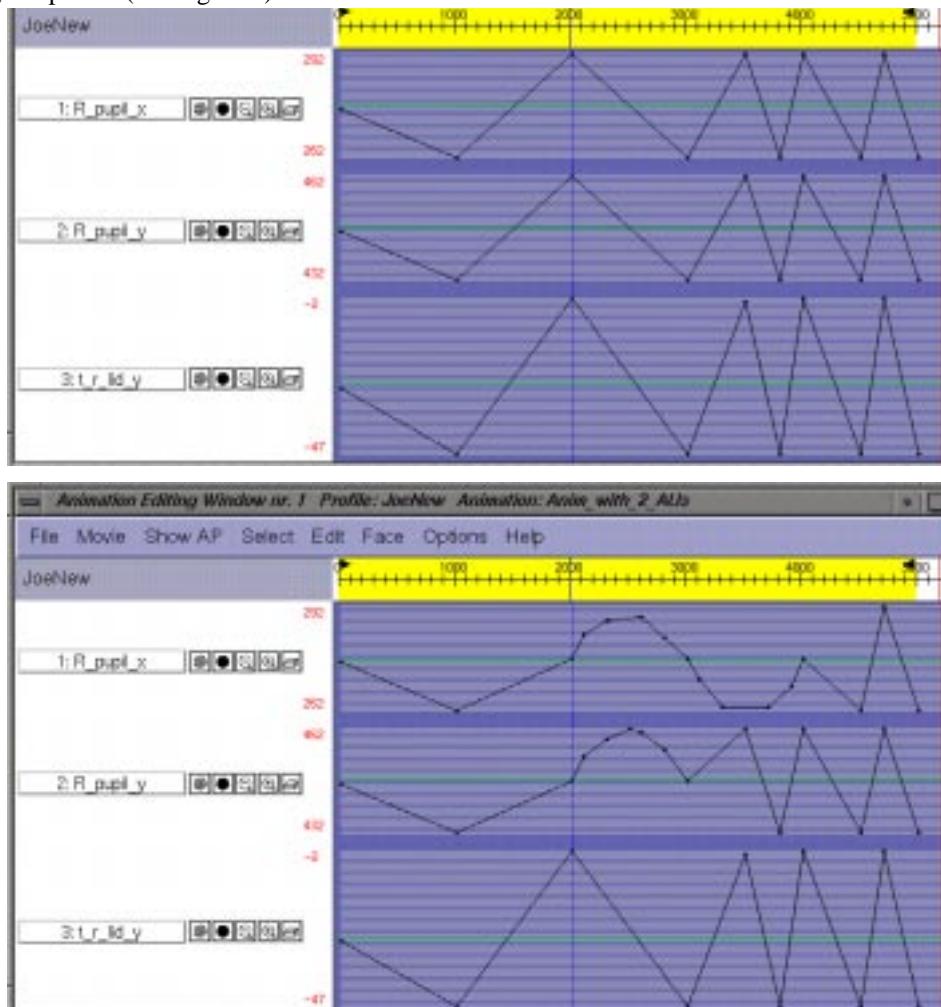


Figure 1. The effect of ‘Reanimate at time’: the animation before and after. Note that only the KPs in the first two APs have been replaced, after 2000 ms till 4000 and 3000 ms, respectively.

The **Reanimate at time** function is handy to insert pieces of animation at increasing time moments, as well as to superimpose animations effecting different part of the face. E.g. blinks can be added later to an animation which had nothing defined for the eyelids

2.2. Using the expression repertoire

The **Reanimate at time** function, as explained above, makes it possible to re-use pieces of animations. In principle, one can always do reanimation ‘by ID’ or ‘by label’. However, the user has to think ahead to consistently label the control points in order to be able to exploit later the reanimate by label possibility. Moreover, even if e.g. two faces, Face1 and Face2, have control points with pair-wise identical labels, it is not assured that a smile made for Face1 will have a similar effect on Face2. It requires careful design of both the faces and the expression to be able to reproduce a similar expression on a face by reanimating an expression made for another face.

The concept of **repertoire** [7] has been coined for re-usable elements in CharToon, both concerning facial features and animations. By using a facial feature and an animation repertoire, it is possible to re-use the animations from the repertoire in such a way that the semantics of the animations is also preserved. In other words, when reanimating a smile from the animation repertoire (made originally for a special face), the result will be again a smile on the current face, assuming that the current face was built up from facial repertoire elements. As hinted to above, the visible result of reanimation depends on the design of the two faces as well as on the design of the animation. Hence it is quite a challenge to compile a set of facial features — eyes, mouths, eyebrows, cheeks, etc. — of different complexity which lend themselves for re-using not only these components when building faces, but also re-using animations made for one of the components.

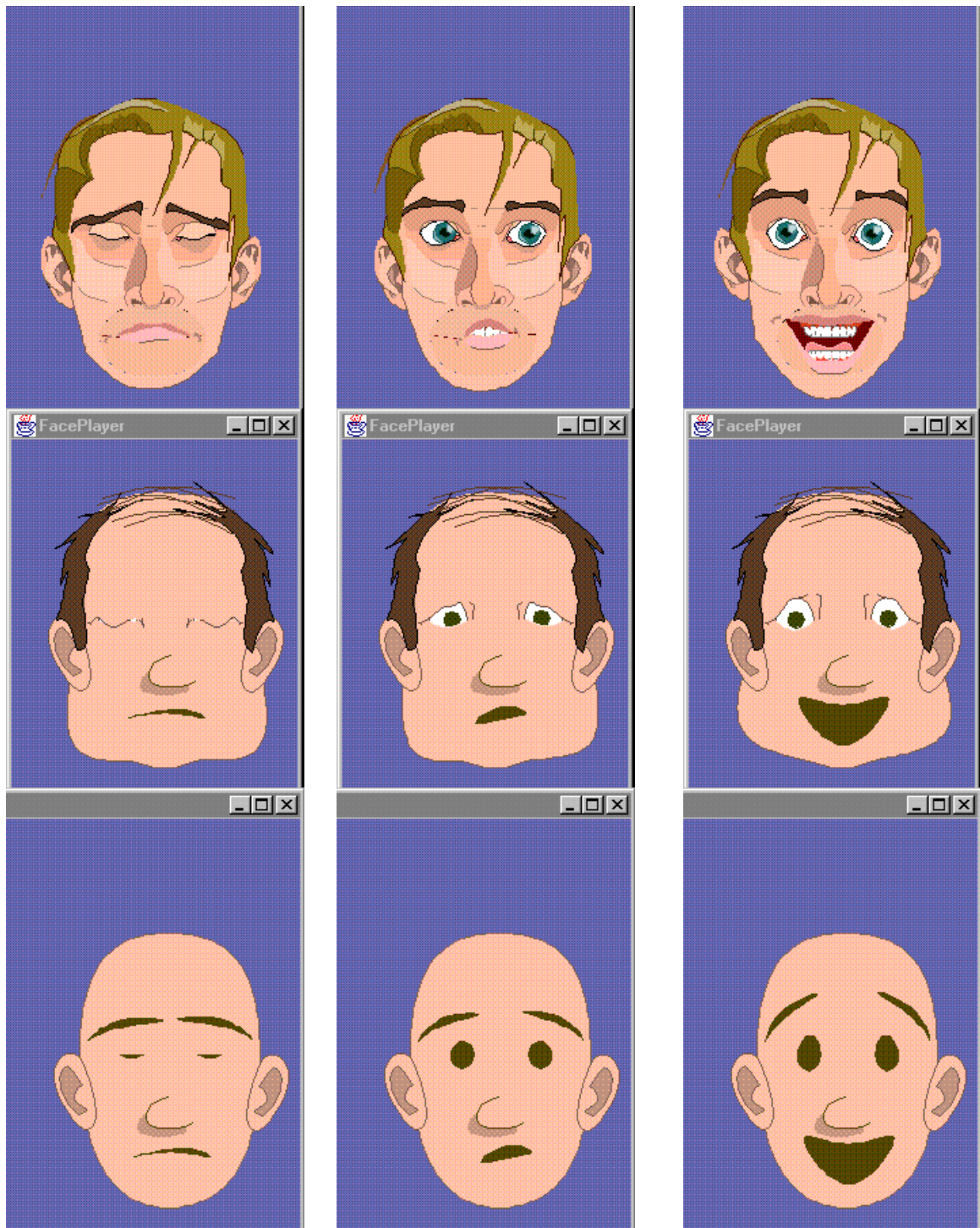
The facial feature components form the **facial repertoire**. There is a mouth repertoire, an eyes repertoire, etc., with a special, so-called **reference** design for each features. In each class, there are corresponding control points with identical labels. Simpler elements contain only a subset of the control points of the reference element.

For the face made of the reference facial features, an animation is made for different expressions, including the 6 basic expressions. An **animation repertoire** may contain snapshots (e.g. a surprised face) as well as time-dependent animations of expressions (a face turning from neutral to surprised and to neutral again). The animation repertoire files are .all files, to be found in the **CompleteAnimations** directory, containing information on the profile in addition to the animation.

Because of the consistent labelling of the control points, the animation repertoire can be used for any face made of facial feature repertoire elements. Moreover, as a result of careful design, it is also guaranteed that the re-used animations, originally made for the reference face, will have the same effect on other faces, see Figure 2. The full description of the provided repertoire elements and design recipes for their usage are discussed in [3]. The animation repertoire provided with CharToon 2.1 are all snapshots of variants of the 6 basic expressions and of some others, made for the **Generic** face. The files, all in subdirectories of **CompleteAnimations/Repertoire**, are listed in Appendix I. This repertoire can be extended or replaced by the user. (Remember that an animation repertoire may contain time-dependent animations too, not only snapshots.) Here we outline only how to make animations by using the (an) animation repertoire.

The **CompleteAnimations/RepertoireSamples.all** file contains an animation made by using most of the expressions in the **Repertoire**. This animation can be used by **Reanimate** to see how the expressions look like on a face (newly) made from facial repertoire elements.

It is possible to insert snapshots of expressions at given time moments, similar to key-framing. Automatically Animation Editor will provide linear interpolation between the KPs of the inserted snapshots. The user is free to refine the transition between the expressions, by inserting KPs. It is also possible to adjust the inserted expressions. The other possibility is to insert animations, e.g. a smile. Besides inserting expressions at different times, it is also useful to compile an animation by using repertoires for parts of the face. E.g. for a talking head, the eyebrow and eye expressions can be superimposed over the animation of the mouth. As of manipulating inserted animations, the same things apply as for snapshots.



Sadness/Burial_Face

Surprise/Astonishing_Surprise

Smile/Absolute_Joy

Figure 2. Three expression from the animation repertoire, shown on three faces: **SimpleHead**, **Medium-Head** and **Generic**. The faces were made up from (unmodified) elements of the facial repertoire.

2.3. Zoom in/out all

A new facility in CharToon 2.1 is to zoom in/out (see Chapter 6.4.1 in [5] for zooming) for all shown APs, resulting in increasing/decreasing the zoom status of each APs, by the following new menu items:

Show AP/Zoom in all zoom in for all shown APs,

Show AP/Zoom out all zoom out for all shown APs.

If all the APs had identical (e.g. ‘overview’) view, then by doing parallel zoom in/out the resulting views will be also identical for all APs. However, if the initial views were different, the difference will be preserved, unless an extreme view is reached.

3. Lip sync

3.1. Defining visemes

A **talking head** is a head with a mouth animated in such a way that the mouth movement corresponds to the audio spoken (or sung) by the head. In principle it was possible already in CharToon 2.0 to generate animation for the mouth by hand, frame by frame, defining in Animation Editor what mouth shape has to be shown at certain time moments. However, such a process is impossible in practice, partly because of the tedious work needed to make each mouth shape one by one, partly because of the ad-hoc hacking assumed to produce the individual mouth shapes. A general approach to overcome these difficulties is to use a given set of mouth shapes — the so-called visemes —, and to produce the animation for the mouth as a sequence of visemes from the predefined set. The issue of deciding the sequence and timing of the visemes according to audio is a research topic in itself. Without bothering about the related issues, here we assume that:

- the set of mouth shapes (visemes) to be used is agreed upon;
- the time sequence of the requested visemes is known, and given in the form of an ascii input file (see 3.2).

What remains for the user of Animation Editor is:

- to design the set of visemes to be used as mouth snapshots;
- to make an animation of the mouth according to the given viseme sequence.

A **viseme** is a snapshot of the mouth. It depends on the intended usage of the talking head, how many visemes should be used and how detailed they should be. E.g. if a news-reader head is to be looked at by hearing impaired people too, then very well articulated and refined mouth shapes (with tongue and teeth visible sometimes) are to be produced. Hence the head has to have a mouth which is capable of producing detailed deformations, and a big number (40-60) of visemes have to be provided as ready-made units for lip sync. On the other hand, if one would only like to give the impression of mouth movements during speech, a few (6-8) mouth shapes may be sufficient. The mouth can be simple, cartoonish, just having enough control possibilities to ‘make’ the low number of mouth shapes. There are different recommended viseme sets available for different languages. However, there is no single set accepted for English [1, 3, 8]. One should always have the possibility to design a new set too, for special purposes (e.g. singing).

Once the set of visemes to be used is agreed upon, the animator has to make with Animation Editor a corresponding mouth shape for each viseme, and save it as a complete .all animation (see [5] Chapter 6.3.1 for details) into the **viseme definition file** for the viseme. In practice, a viseme will be a snapshot, an animation with a single KP for each parameter for the mouth. (Some mouth parameters may take their neutral value, but these have to be given also explicitly.)

A viseme which was designed for a given mouth (head) will not, in general, work for other mouths. In principle one has to design each viseme of the viseme set for every different mouth. Note, however, that the predefined mouth repertoire had been provided in such way that a single viseme set can be re-used for all the provided mouth designs. (The issue is discussed in detail in [7].) However, the effect may not be perfect.

In this document we will use the ‘**ExtendedEnglish**’ **viseme set** for English, consisting of 47 visemes. The visemes were developed at CWI, based on visual clues from [8]. The corresponding files are provided with CharToon 2.1, and are in the **Visemes/Neutral** subdirectory. The file names and the corresponding mouth shapes are given in Appendix II. There are 6 other viseme sets available too, each of them is a variant of the original one, corresponding to mouth shapes of visemes while one of the 6 basic expressions is shown on the face (effecting also the mouth sometimes), see Figure 3 and 4. These variants are to be found in the **Visemes/Sad**, ... etc. subdirectories. These viseme sets were designed for the **GenMouth** reference mouth, but in such a way that they can be used with any of the mouth facial repertoire elements, to be found in the **UserDefinedComposite/Mouths** directory (see Figure 4 and 5). To experiment with viseme sequences, it is useful to take the **Visememouth2** mouth, which is an enlarged version of the reference mouth.

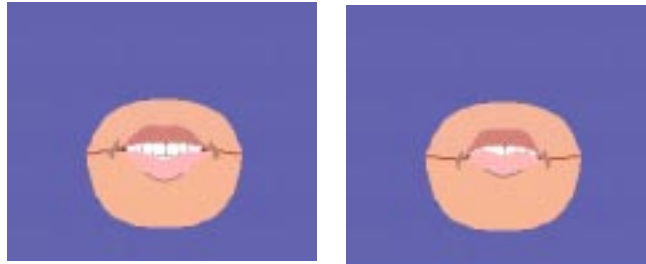


Figure 3. The variants of the **C_CHurch** viseme with different expressions: **C_CHurch_Smile** and **C_CHurch_Sad**. The mouth is **GenMouth**.

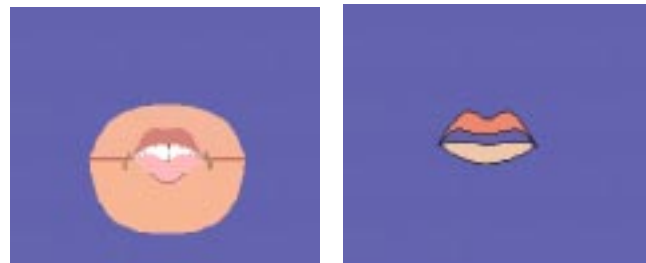


Figure 4. The **C_CHurch** viseme shown on the reference **GenMouth** and on the simpler **Mouth2**.



Figure 5. The **C_CHurch** viseme on profile and quart view variants of **GenMouth**.

3.2. The viseme profile and the viseme sequence files

If one wants to generate a mouth animation automatically, based on some viseme sequence, three kinds of input files are needed:

- a **viseme profile file**, an ASCII file with '.avprof' extension, listing the VID and the viseme definition file for the individual visemes to be used;
- the **viseme definition files**, ASCII files with '.all' extension, containing the definition of the individual visemes (the file names all must be listed in the viseme profile file);
- a **viseme sequence file**, an ASCII file with '.avseq' extension, containing a time sequence of visemes;

A **viseme profile file** is an ascii file (see Figure 6), to be produced by a text editor. The first non-comment line must be a positive integer, telling the number of visemes to be used. Then for each viseme an integer VID and a filename (pathname relative to the **Visemes** directory) is provided in separate lines. The filename is an “.all” file, containing a snapshot (the corresponding mouth shape), which had been designed earlier (see previous chapter). The file name should be chosen carefully and in a systematic way, giving an idea about the viseme it describes.

```
// Comments
// Viseme set for English
// The profile the visemes are designed for
// visememouth2 (mouth) profile, also in Generic head
// Nof visemes
47
// VID filename
1   Neutral/A_cAr.all
2   Neutral/A_mAp.all
3   Neutral/A_bAlt.all
4   Neutral/B_Boy.all
5   Neutral/C_CHurch.all
6   Neutral/D_Day.all
7   Neutral/E_bEAt.all
8   Neutral/E_bEd.all
...
```

Figure 6. Beginning of the **ExtendedEnglish.avprof** viseme definition file.

A **viseme sequence file** is an ascii file (see Figure 7), to be produced by some software outside of CharToon (see Chapter 4 for such a sw for a special case). This is the file containing the actual input, the sequence of visemes to be turned into an animation of the mouth in Animation Editor. Each line contains a time in milliseconds (integer, multiple of 100) and an VID (positive integer). The VID is meant to be an VID listed in the viseme profile file being used.

```
// Comments
// The viseme set to be assumed
// Time VID
0 47
100 1
200 17
300 24
400 47
500 19
600 1
700 20
800 13
900 19
1000 47
1100 43
1200 24
1300 24
1400 35
...
```

Figure 7. Beginning of the **SampleSequence.avis** viseme sequence file.

All the input files are to be put in the **visemes** subdirectory. It is possible to use subdirectories of **visemes**, then the relative path has to be given in the visemes profile file. This is recommended if different viseme profile file and viseme definition files are around.

3.3. Generating lip sync

One generates lip sync by using the new menus in *File/Viseme*. The menus offer file selection dialogs to choose the file to be loaded. First one has to load the viseme profile file to be used:

File/Viseme/Load viseme profile load a viseme profile file.

There is no visible result of loading a viseme profile file. What happens, though, is that the files listed in the viseme profile file are opened, and the visemes as snapshots are read into memory, for further usage. These visemes will be identified and referred to later by their VIDs.

If an error is encountered when reading in a viseme profile file or processing a file referred to by it, the error is reported.

If a viseme profile has been loaded with success, one may load a viseme sequence file:

File/Viseme/Load viseme sequence load a viseme sequence file,

As default, the viseme sequence is inserted at the beginning (time 0), and the rest of the effected channels is cleaned. If one wants to insert a viseme sequence at a given time, the time marker has to be set (see 6.6.2 of the manual for details). The time marker can be (re)set by the mouse in the ruler region

Mouse Press + ALT set time marker,
(=MouseMiddle)

Mouse Press + ALT +SHIFT remove time marker, if there was one at the time of mouse press.
(=MouseMiddle + SHIFT)

It is important to remember that the original content 'after the inserted visemes' will be erased. Hence one has to work from left to right when inserting different viseme sequences.

Note that the visemes in the .all files (referred to by the viseme profile file) are supposed to be made for (mouth) parameters which are present in the current profile. In other words, you must be working with the face which has the mouth parameters in the profile identical to the mouth parameters used to define the individual visemes. If this is not the case, some mouth labels will be reported as 'not found'. (Actually when inserting a viseme, a 'reanimate by label' action takes place, see [5] or details.) This is not a fatal error, but an indication that you are working with mismatching profile and viseme profile, a situation you should normally avoid. However, when working with a mouth from the provided mouth repertoire, the visemes do produce proper mouth shapes even if some of the APs (control points with proper labels) are not provided for the current mouth.

Once a viseme sequence has been loaded, it is allowed to fine-tune the resulting animation by hand editing. It is possible and advised to save intermediate work as ordinary animation. which can be used further as any other animation (loaded, edited, turned into movie,...)

Besides generating lip sync automatically, it is possible to load individual visemes at given times, by using the *File/Reanimate at time* function. (Note that the visemes are not in the **CompleteAnimations** subdirectory, which you get when calling the *File/Reanimate at time* menu, but in the **visemes** subdirectory.)

3.4. Possible further extensions

The lip sync facilities are meant to support first experiments with talking heads. Depending on further needs and eventual auxiliary software to be used for audio to viseme, phoneme to viseme or text to speech (to viseme) generation, different extensions and refinements could be made.

The major bottleneck of the current facilities for lip sync is that in the viseme sequence file the time of visemes must be multiple of 100 milliseconds. This is required because of **the (fixed) time granularity** of Animation Editor is 100 ms. If it turns out that this granularity is too rough to produce good-enough quality

lip sync, Animation Editor will be improved in such a way that the time granularity can be set to smaller steps.

Another useful feature would be to get some hints about the visemes read in from a viseme sequence file. It would be possible to extend AnimationEditor with a **visemes channel**, where symbols corresponding to the read-in visemes would be shown, or even edited by the user.

Finally, Face Player, when called from Animation Editor, does not play audio. This could be easily changed in the future, allowing **to watch and listen to a talking head** from Animation Editor.

4. Tools to map a phoneme sequence to a viseme sequence

In this chapter we explain how to use the auxiliary software tools (all written in Java 1.1), which were developed to be able to generate viseme sequences from phoneme sequences for Dutch language, according to the practice and expertise of IPO [2].

In the first stage of experiments, we ignored much information on coarticulation, present in the input phoneme sequence. As we had developed a viseme set meant for the English language, we also had to provide a tool to ‘approximate’ some typical Dutch phonemes by the English phonemes and corresponding visemes.

Three programs are to be used one after the other, each taking a single ASCII input file and producing an ASCII output file with the same name but a new extension. The format of the files can be derived from Figure 8. The process consists of the following steps.

1. The **ProcPhonemes** program takes a **.aoph original phoneme sequence file** (provided by IPO) and produces the **.apph processed phonemes file**. This program changes the original sequence in two respects:

- only a limited set of phonemes will be included in the output:
 - DIPHs, providing information on co-articulation between phonemes in the input are skipped;
 - for special phonemes ("Ei", "9y", "ai", "ui", "iu", "yu", "eu", "Ai", "Oi", "E:", "9y") also a second part dummy phoneme is generated ("Ei2", "9y2", "ai2", "ui2", "iu2", "yu2", "eu2", "Ai2", "Oi2", "E:2", "9y2"). The symbols used for the phonemes are according to the practice of IPO. Appendix VI gives some idea about their meaning.
- for each phoneme a single time is assigned, instead of a start time t and a duration d , as is in the original sequence. The time is:
 - t for plosives ("p", "t", "k", "b", "d", "g", "c", "tS", "dZ") and the first component of special phonemes ("Ei", "9y", "ai", "ui", "iu", "yu", "eu", "Ai", "Oi", "E:", "9y");
 - $t+0.75d$ for the second component of special phonemes;
 - $t+0.5d$ for the rest

2. The **SamplePhonemes** takes the processed phonemes file and samples it at 100 ms, producing the **.asph sampled phoneme sequence file** which contains phonemes at multiples of 100 ms (not necessarily at each discrete time, only for those where there was a phoneme prescribed close to the discrete time).

3. The **MapPhonemes** program takes a sampled phoneme sequence file and maps the phonemes to visemes, producing the **.avis viseme sequence file**. MapPhonemes takes the necessary mapping information from the predefined **visemetable file** which defines the one to one correspondence of phonemes to visemes. This file named **visemetable** should be available in the same directory where the program file is. The initial content of the file, used for the mapping of Dutch phonemes to visemes of the ExtendedEnglish viseme set, is given in Appendix VI. The file should be re-written with a file defining the intended mapping of visemes.

Each tool should be invoked as

```
java ToolName FileName
```

The `FileName` should be given without extension, and it is looked for in the **PhonemeSequences** subdirectory. The program looks for the given file (always in the **PhonemeSequences** subdirectory) with the required extension, and produces an output file with `FileName` but the new extension. See Figure 8 for a sample sequence of produced files.

The final output, an **.avis viseme sequence file** can be loaded to Animation Editor and will be turned into an animation of the mouth automatically, as explained in Chapter 3.3.

DefPhonSeq.aoph

//time	duration	phon/diph	symbol for phoneme/diph
0.00000	0.02000	PHON	.
0.00000	0.02937	DIPH	.!?!
0.02000	0.01500	PHON	ui
0.02937	0.04075	DIPH	?!I1
0.03500	0.06531	PHON	Ei
0.07012	0.04018	DIPH	I1k1
0.10031	0.06000	PHON	k
0.11031	0.09043	DIPH	k1b1
0.16031	0.05043	PHON	yu
0.20075	0.04068	DIPH	b1E1
0.21075	0.06656	PHON	E
0.24143	0.07706	DIPH	E1n1
0.27731	0.06700	PHON	n
0.31850	0.04656	DIPH	n1@1
0.34431	0.05187	PHON	@
0.36506	0.05193	DIPH	@1n1
0.39618	0.05212	PHON	n
0.41700	0.08275	DIPH	n1s1
0.44831	0.08643	PHON	s

DefPhonSeq.apph

10	.
20	ui
31	ui2
35	Ei
83	Ei2
100	k
160	yu
198	yu2
244	E
310	n
370	@
422	n
491	s

DefPhonSeq.asph

0	.
100	k
200	yu2
300	n
400	n
500	s

DefPhonSeq.avis

0	47
100	15
200	44
300	22
400	22
500	35

Figure 8. The **DefPhonSeq.aoph** input file and the corresponding output files generated by the phoneme sequence processing programs.

One can experiment with different viseme sets, as well as different mappings of phonemes to visemes. For the first, one has to design new viseme sets, either by modifying visemes in a given set, or from scratch. For the latter, one has to make a new visemetable file. One can experiment with ‘many to one’ mappings to see what happens if only a subset of a larger viseme set is used, or a viseme set (e.g. MPEG4) with a small number of visemes is used. The correspondence of different visemes, given in Appendix V, may be helpful for such experiments.

Acknowledgement

CharToon has been developed at CWI, in the framework of the FASE project, sponsored by STW under nr. CWI 166 4088.

We thank Jeroen Hendrix for making the English-Dutch viseme mapping table, Paul ten Hagen and Han Noot for several discussions on the topic and for comments on this report.

Appendix I

The animation repertoire: .all files in subdirectories of CompleteAnimations/Repertoire

Anger/

- Be_Careful_What_You_Say.all
- Teased_Pested.all
- Furious.all
- Annoyed.all
- Reproach.all
- Rage_Hate.all
- Deceived_BadMood.all
- Sulk.all

Disgust/

- Disgust_Amused.all
- Disgust_Mockery.all
- Disgust.all
- Revulsed.all
- Disgust_Moderate.all
- Violent_Disgust.all

Fear/

- I'm_Going_To_Be_Late.all
- I'll_Never_Do_It_Again.all
- Don't_Do_That.all
- Terror.all
- Fear_Waiting.all
- Fear.all

Sadness/

- Regrets.all
- Deceived.all
- What_a_Disaster.all
- Remembering_Souvenir.all
- Burial_Face.all
- Pitiness.all
- Sadness.all

Smile/

- Are_You_Joking.all
- Smiling_For_The_Camera.all
- Hey_Look_At_That.all
- Nasty_Idea.all
- How_Cute.all
- Adoration.all
- Derision_Smile.all
- Joy_Madness.all
- Absolute_Joy.all
- Pleased.all
- Smile.all

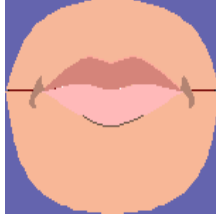

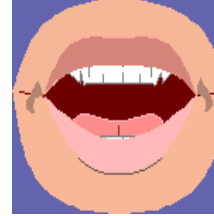
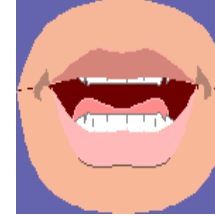
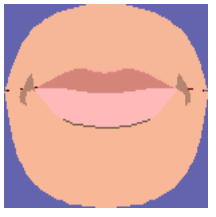
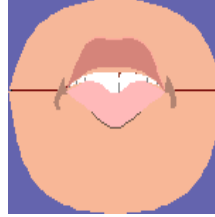
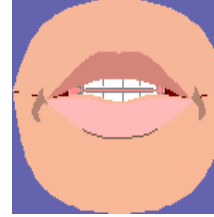
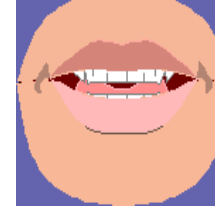
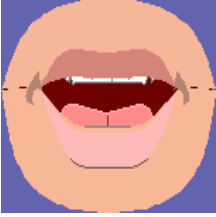
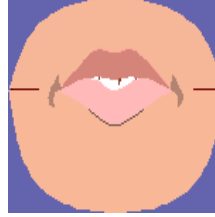
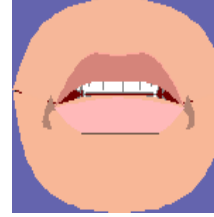
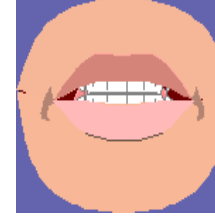

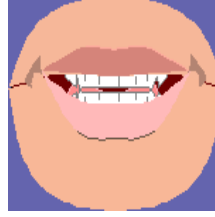
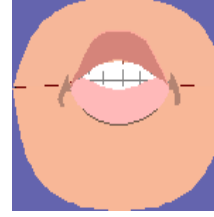
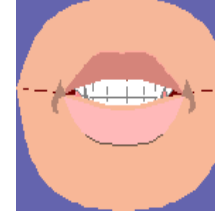
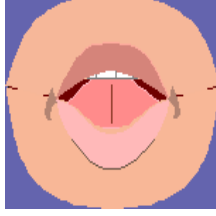
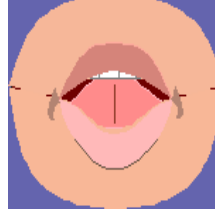
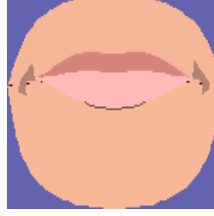
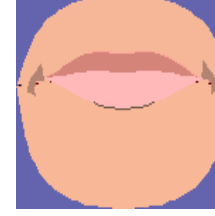
Surprise/

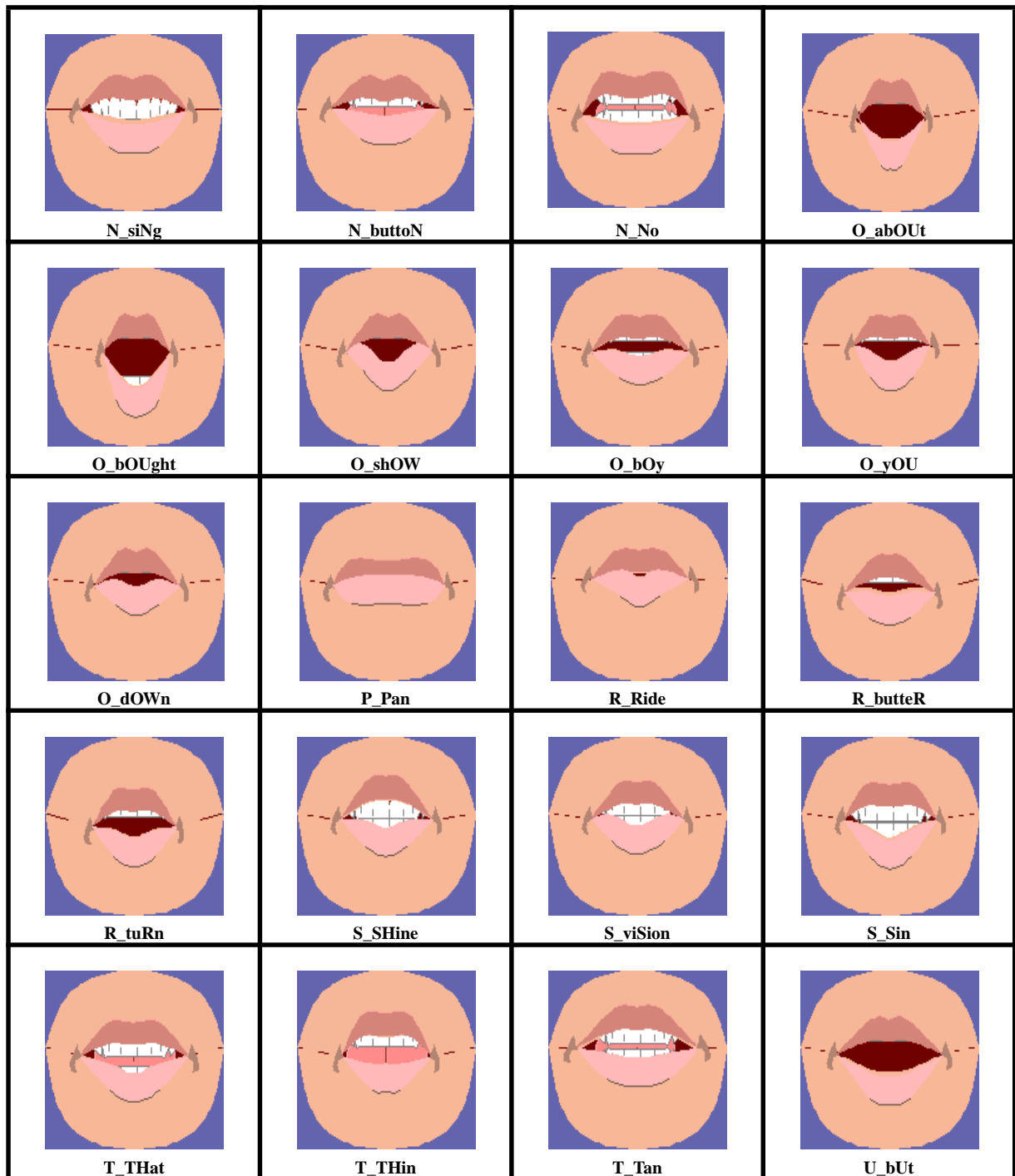
- Astonishig_Surprise.all
- Surprise_Slightly_Disgusted.all
- Surprise_Bad.all
- Surprise_Good.all
- Surprise.all

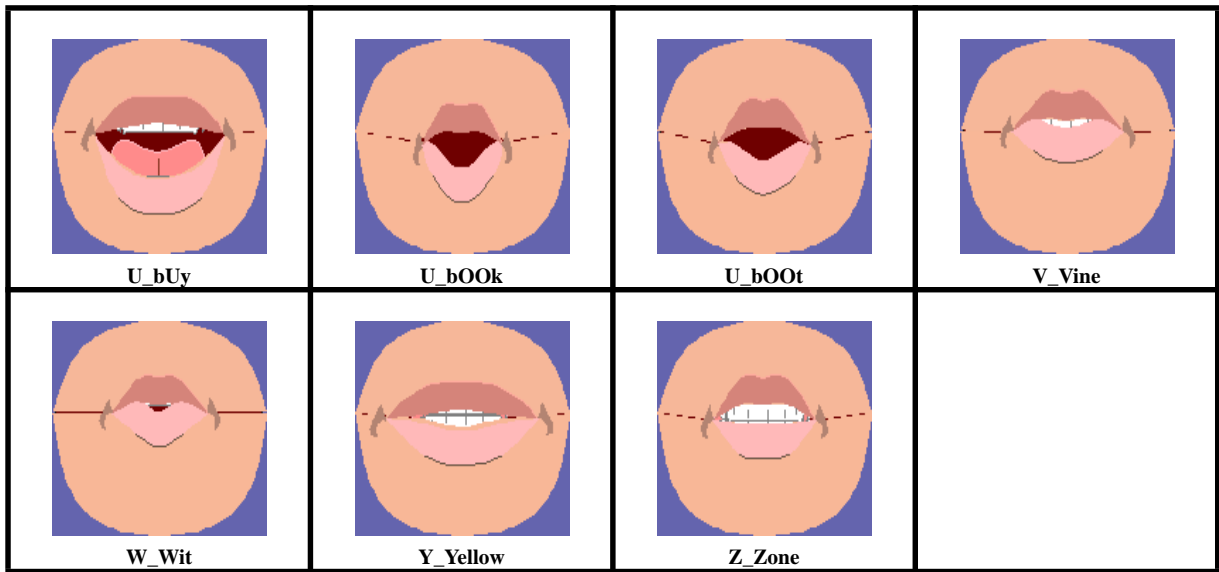
Others/

- Crying_Of_Joy.all
- Please_Help_Me.all
- Exhausted_Trying_To_Explain.all
- Bored.all
- Neutral.all
- Opportunity_To_Revenge.all
- Where_Is_The_Coffee.all
- Impressed.all
- Are_You_sure.all
- Incredulity_Contempt.all
- Attention_Concentration.all
- Do_Not_Know.all
- Not_Sure.all
- Listening_to_Nonsense.all
- Lightly_Sarcastic.all
- Smack.all
- Doubt_suspicion.all

Appendix II:
The ExtendedEnglish viseme set for neutral expressions, shown on the GenMouth mouth

			
0_Closed	A_cAr	A_mAp	A_bAI
			
B_Boy	C_CHURch	D_Day	E_bEA
			
E_bEd	F_Fine	G_glottalstop	G_Got
			
H_Head	I_bIt	J_Jungle	K_Can
			
L_Lovable	L_Let	M_bottoM	M_My





Appendix III

The DECtalk viseme set from [8]

#	DECtalk viseme	#	DECtalk viseme
0	SI_0_Silence	30	LX_30_pvocL
1	IY_1_bEAt	31	M_31_Met
2	IH_2_bIt	32	N_32_Net
3	EY_3_bAIt	33	NX_33_Sing
4	EH_4_bEt	34	EL_34_bottLe
5	AE_5_bAt	35	D_35_Debt
6	AA_6_pOt	36	EN_36_buttoN
7	AY_7_bUy	37	F_37_Fin
8	AW_8_dOWn	38	V_38_Vet
9	AH_9_bUt	39	TH_39_THin
10	AO_10_bOUt	40	DH_40_THis
11	OW_11_bOAt	41	S_41_Sit
12	OY_12_bOy	42	Z_42_Zoo
13	UH_13_bOOK	43	SH_43_SHin
14	UW_14_IUte	44	ZH_44_meaSUrE
15	RR_15_bIRd	45	P_45_Pet
16	YU_16_cUte	46	B_46_Bet
17	AX_17_About	47	T_47
18	IX_18_kisseS	48	D_48_Debt
19	IR_19_killERd	49	K_49_Kit
20	ER_20_bIRd	50	G_50_Get
21	AR_21_butteR	51	DX_51_baTTeR
22	OR_22_calOR	52	TX_52_laTin
23	UR_23_chURn	53	Q_53_glstOp
24	W_24_Wet	54	CH_54_CHurch
25	Y_25_Yet	55	JH_55_Judge
26	R_26_Red		
27	LL_27_Let		
28	HX_28_Head		
29	RX_29_pvocR		

Appendix IV
The MPEG4 viseme set based on [3]

#	MPEG4 viseme	example
0	none	
1	P_B_M	Put, Bed, Mill
2	F_V	Far, Voice
3	Th_Dh	THis, THat
4	T_D	Tip, Doll
5	K_G	Call, Gas
6	tS_dZ_S	CHair, Join, SHe
7	S_Z	Sir, Zeal
8	N_L	Lot, Not
9	R	Red
10	A:	cAr
11	E	bEd
12	I	tIp
13	O	tOp
14	U	bOOk

Appendix V

Correspondence of the ExtendedEnglish viseme set to DECtalk and MPEG4. For DECtalk, the correspondence indicates identical mouth shapes. For MPEG4, the many to one mapping is based on similar mouth shapes (MPEG4 ones approximating the ExtendedEnglish ones).

#	ExtendedEnglish	DECtalk	MPEG4
0	0_Closed	0	0
1	A_cAr		10
2	A_mAp		10
3	A_bAlt		10
4	B_Boy	46	1
5	C_CHurch	54	6
6	D_Day	35	4
7	E_bEAt	1	11
8	E_bEd	4	11
9	F_Fine	37	2
10	G_glottalstop	53	-
11	G_Got	50	5
12	H_Head	28	-
13	I_bIt	2	12
14	J_Jungle	55	6
15	K_Can	49	5
16	L_Lovable		8
17	L_Let		8
18	M_bottoM	31	1
19	M_My		1
20	N_siNg		8
21	N_buttoN		8
22	N_No	32	8
23	O_abOUt		10
24	O_bOUght	10	13
25	O_shOW		13(?)
26	O_bOy	12	13(?)
27	O_yOU		14
28	O_dOWn	8	13
29	P_Pan	45	1
30	R_Ride	26	9
31	R_butteR		9
32	R_tuRn	23	9
33	S_SHine	43	6
34	S_viSion		6
35	S_Sin	41	6
36	T_THat		3
37	T_THin	47	3
38	T_Tan		4
39	U_bUt	9	13(?)
40	U_bUy	37	10(?)
41	U_bOOk	13	14
42	U_bOOt		14
43	V_Vine	38	2
44	W_Wit	24	2(?)
45	Y_Yellow	25	6(?)
46	Z_Zone	42	7

Appendix VI

Mapping of Dutch phonemes to the ExtendedEnglish viseme set (content of the default visemeta-ble file)

```
//phoneme VID corresponding file (for information only)
// phoneme symbols are according to the practice of IPO
// * in place of phonemes means that there is no Dutch phoneme corresponding to the given English viseme
A      1      //A_cAr.all
{      2      //A_mAp.all
e      3      //A_bAIIt.all
b      4      //B_Boy.all
tS     5      //C_CHurch.all
d      6      //D_Day.all
i      7      //E_bEAt.all
E      8      //E_bEd.all
f      9      //F_Fine.all
-      10     //G_glottalstop.all
g      11     //G_Got.all
h      12     //H_Head.all
I      13     //I_bIt.all
dZ     14     //J_Jungle.all
k      15     //K_Can.all
L      16     //L_Lovable.all
l      17     //L_Let.all
*      18     //M_bottoM.all
m      19     //M_My.all
N      20     //N_siNg.all
=n     21     //N_buttoN.all
n      22     //N_No.all
Au     23     //O_abOUt.all
O      24     //O_bOUght.all
*      25     //O_shOW.all
OI     26     //O_bOy.all
*      27     //O_yOU.all
AU     28     //O_dOWn.all
p      29     //P_Pan.all
*      30     //R_Ride.all
*      31     //R_butteR.all
9      32     //R_tuRn.all
S      33     //S_SHine.all
Z      34     //S_viSion..all
s      35     //S_Sin.all
D      36     //T_THat.all
T      37     //T_THin.all
t      38     //T_Tan.all
@      39     //U_bUt.all
aI     40     //U_bUy.all
U      41     //U_bOOk.all
u      42     //U_bOOt.all
v      43     //V_Vine.all
*      44     //W_Wit.all
j      45     //Y_Yellow.all
z      46     //Z_Zone.all
.      47     //0_Closed.all
//
// substitutions1
// Some Dutch phonemes do not exist in English. However, an (English) viseme substitute is used.
y      32
a      39
```



```

e      8
o      24
2      32
Ei     3
//
// substitutions2
// The special phonemes are replaced by two substitutes, below the visemes corresponding to the 2 substitutes are listed.
9y     24
9y2    32
ai     39
ai2    7
oi     24
oi2    7
ui     42
ui2    7
iu     7
iu2    42
yu     32
yu2    44
eu     8
eu2    44
Ai     1
Ai2    7
Oi     24
Oi2    7
c      5
x      20
G      20
C      20
J      45
R      20
w      43
E:     8
E:2    45
9:     32
O:     24
{:"    8
// end

```

References

1. Ezzat, T., Poggio, T. (1998)
MikeTalk: A talking facial display based on morphing visemes. Proc. of Computer Animation'98, IEEE, Los Almos.
2. IPO (2000)
Practice of coding phonemes of Dutch audio sequences at the IPO institute, personal communication.
3. ISO (1998)
Information Technology - Coding of Audio-Visual Objects: Visual, ISO/IEC 14496-2 Committee Draft, Tokyo
4. Lelièvre, A. (2000)
CharToon Tutorial - Making an animated face from scratch or from the repertoire, Report INS-R00??, CWI, Amsterdam, to appear
5. Noot, H., Ruttkay, Zs. (2000)
CharToon 2.0 Manual, Report INS-R0004, CWI, Amsterdam
6. Ten Hagen, P., Noot, H., Ruttkay, Zs. (1999)
CharToon: a system to animate 2D cartoon faces, Short Papers Proceedings of Eurographics'99
7. Ten Hagen, P. (2000)
A Facial Repertoire for Animation , Short Papers Proceedings of Eurographics'2000, to appear
8. Waters, K., Levergood, T. (1993)
DECface: An automatic lip-synchronization algorithm for synthetic faces, Technical Report CRL 93/4, Digital, Cambridge.