5-1-2004

Synchronizing Keyframe Facial Animation to Multiple Text-to-Speech Engines and Natural Voice with Fast Response Time

William H. Pechter
Dartmouth College

Synchronizing Keyframe Facial Animation to Multiple Text-to-Speech Engines and Natural Voice with Fast Response Time

A Senior Honors Thesis
Submitted to the faculty
in partial fulfillment of the requirements for the
Degree of Bachelor of Arts in Computer Science

Dartmouth Computer Science Technical Report TR2004-501

by
William H. Pechter
Dartmouth College
Hanover, NH
May 2004

_____
Lorie Loeb

_____
Hany Farid

_____
Stephen Linder

Abstract

This thesis aims to create an automated lip-synchronization system for real-time applications. Specifically, the system is required to be fast, consist of a limited number of keyframes with small memory requirements, and create fluid and believable animations that synchronize with text-to-speech engines as well as raw voice data.

The algorithms utilize traditional keyframe animation and a novel method of keyframe selection. Additionally, phoneme-to-keyframe mapping, synchronization, and simple blending rules are employed. The algorithms provide blending between keyframe images, borrow information from neighboring phonemes, accentuate phonemes b, p and m, differentiate between keyframes for phonemes with allophonic variations, and provide prosodromic variation by including emotion while speaking. The lip-sync animation synchronizes with multiple synthesized voices and human speech. A fast and versatile online real-time java chat interface is created to exhibit vivid facial animation.

Results show that the animation algorithms are fast and show accurate lip-synchronization. Additionally, surveys showed that the animations are visually pleasing and improve speech understandability 96% of the time. Applications for this project include internet chat capabilities, interactive teaching of foreign languages, animated news broadcasting, enhanced game technology, and cell phone messaging.

Table of Contents

Chapter 1 – Introduction.....1

Chapter 2 – Phonetic Transcription and Phoneme Alphabets 3
2.1 Phonetics and Phonemes.....3
2.2 The International Phonetic Alphabet: Phonemic versus Phonetic Transcription.....3
2.3 Choosing Phonemic Transcription.....4
2.4 Phoneme Alphabets Used.....4

Chapter 3 – Lip-Sync Animation and Diphone Durations 5
3.1 Text-to-Speech Engines, Diphones, and Prosodromic Variation 5
3.2 Interfacing Festival Text-to-Speech with Animation 6

Chapter 4 – Articulation of Phonemes and Choosing Eight Keyframes for Animation 8
4.1 Keyframe Assignment 8
4.2 Articulation of Consonants 10
4.3 Articulation of Vowels.....11
4.4 Articulation of Diphthongs 12
4.5 Creating Strip Files with In-Betweens.....12

Chapter 5 – Using X-sheets and Loading into the Master Data Structure 14
5.1 Parsing Phoneme and Word Data Files 14
5.2 X-Sheets and Aligning the Data 15
5.3 Master Data Structure 17

Chapter 6 - The Simple Blending Function – Smoothing Transitions between Keyframes.....19

Chapter 7 – Complex Blending Rules 23
7.1 Blending Within Phonemes – Assigning Multiple Keyframes to Diphthong Phonemes 23
7.2 Modifying/Grouping Blending Rules 24
7.3 Adding Prosodromic Variation.....24
7.4 Stealing Frames.....26
7.5 Accentuating Phonemes /b/, /p/, and /m/ 26
7.6 Allophonic Variation and the Decision Algorithm for Phonemes /d/, /l/, 27 and /v/.....27
7.7 Evaluation 27

Chapter 8 – Applications 29
8.1 Introduction.....29
8.2 Java Implementation of Animation over IP.....29

8.3 Speech-to-Text Voice with Animation over IP 29
8.4 Scalability 30
8.5 Future Work 30

Appendix A 32

Appendix B 35

List of Figures

Figure 1.1 Project Overview 2

Figure 3.1. Phoneme and Diphone Wave Data 6
Figure 3.2 Phoneme Data to JWAnimate 7

Figure 4.1 Phoneme Assignment 8
Figure 4.2. Keyframes 9
Figure 4.3. Places of Articulation (Hall 2004) 10
Figure 4.4. American English Vowel Space (Venditti 2002) 11
Figure 4.5. Strip-File 12

Figure 5.1. Master Data Structure 18

Figure 6.1. Keyframe Assignment and Frame Durations for “I love you” 19
Figure 6.2. Strip File Offsets 20
Figure 6.3. Before Simple Blending 21
Figure 6.4. After Simple Blending I 22
Figure 6.5. After Simple Blending II 22

Figure 7.1. Blending within Phonemes for “hide” 24
Figure 7.2. Frame Durations and Strip File Offsets for “I love you” using Simple Blending and without prosody 25
Figure 7.3. Frame Durations and Strip File Offsets after Addition of Blending within Phonemes and Prosody 25
Figure 7.4. Stealing Frames 26
Figure 7.5. Accentuating Phonemes b, p, and m 27

Figure 8.1. Avatar-over-IP 29
Figure 8.2. New Keyframes 31

List of Tables

Table 3.1. Phoneme Alphabets and Pronunciation Dictionaries.....7

Table 4.1. Consonant Phoneme-to-Keyframe Mapping 11
Table 4.2. Vowel and Diphthong Phoneme-to-Keyframe Mapping.....12
Table 4.3. Summary of Phoneme-to-Keyframe Mapping 13

Table 5.1. X-Sheet 1 15
Table 5.2. X-Sheet 2 16

Table 6.1. Keyframe Assignment to “I love you” 19

Chapter 1 – Introduction

Humans learn to use auditory and visual cues to comprehend speech from an early age. Lip-reading – one type of visual cue – is not only a tool for the deaf; people with normal hearing use visual information to aid in the understanding of audio signals. Studies reveal that when audio degradation is present, visual cues are particularly helpful (Waters 1994).

Animated avatars – virtual representations of human figures – are capable of improving the comprehension of speech by offering accurate lip-synchronization of speech. Both text and speech drive facial animation systems. Text-driven facial animation involves the use of a text-to-speech engine to provide both a synthesized voice and phoneme timing data for animation rules. Speech-driven facial animation is achieved by obtaining phoneme timing data directly from the speech signal.

Lip-sync animation involves synchronizing timing data derived from speech and the formulation of animation rules. Currently available software employ 3D facial models, complex wire-frames, skin and muscle wrappers, and manipulation of facial movement ("Façade" 2001). Typically, 3D characters require complex lip-sync animation rules, varying adjustment, and training rules to enable the mouth to work correctly (Kakumanu et al. 2001). Additionally, existing facial animation software lacks visual simplicity. Lip-sync animation performed with elaborate faces often causes unreasonable expectations of the character, and an elevated attention to the inaccuracies caused by synchronizing animation to speech. The complex faces may look cluttered, unappealing, and the synchronization appears awkward.

Conversely, when less complex characters are successfully animated, the simplest and slightest facial movements are more fully appreciated by the viewer. These simpler images leave more to the viewer's imagination, and are thereby easier to accept and to identify with. For example, virtual pet keychains are popular because people can relate to the simple images. Thus, the employment of carefully constructed simple images provides the viewer with enhanced overall enjoyment of the animation.

Accordingly, simple images make the facial animation system uncomplicated as fewer images are required to convey highly effective speech. Facial animation rules relying on fewer keyframe transitions are easier to implement. Additionally, the facial animation system benefits from decreased memory requirements. This aids in making the system fast, especially if images are preloaded. By utilizing a small number of keyframes displaying simple images, the original software developed for this thesis produces faster lip-synchronization that is also highly enjoyable for the viewer.

The requirement of accurate synchronization between speech and animation complicates even the simplest facial animation rules. The phoneme timing data obtained from text-to-speech engines strictly constrains the facial animation model. However, by merely focusing on precise synchronization with phoneme timing data, the believability of many existing programs suffers, as certain phonemes are merely represented by opening and closing the mouth (Comet 1998). While this thesis exploits the framework of strict synchronization timing rules, it manipulates them is such a way as to incorporate timing adjustments for more believable lip-sync animation.

Thereby, this thesis aims to create an automated lip-sync system for real-time applications. Specifically, the system is required to be fast, use a limited number of

keyframes, and create fluid and believable animations that synchronize with text-to-speech engines as well as raw voice data.

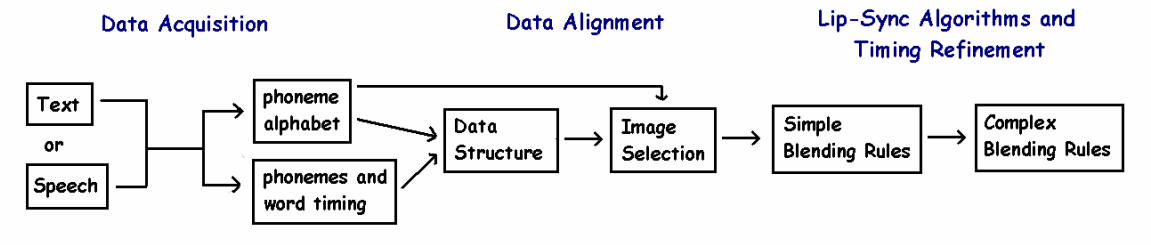


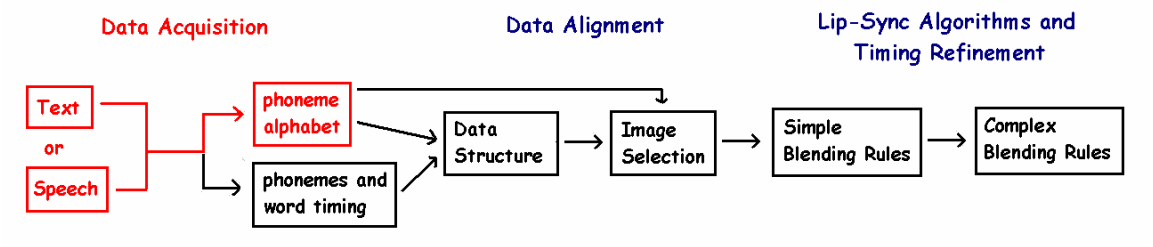
Figure 1.1 Project Overview

Chapters 2 and 3 of this thesis describe the input stage: Chapter 2 outlines the process of narrowing down speech sounds to be used as input for the facial animation, and Chapter 3 explains voice synthesizer selection as well as the phoneme data chosen as input for the facial animation.

The image selection stage is described in Chapters 4 and 5. Here, the techniques of narrowing down to a limited number of keyframes by examining articulation points are described. Additionally, these chapters describe the process of keyframe-to-phoneme mapping and the creation of strip-files. Chapter 5 describes the creation of X-sheets and the subsequent loading of phoneme and word timing data into a Master Data Structure.

Chapters 6 and 7 describe the novel lip-sync algorithms and timing refinements. Chapter 6 outlines the simple blending algorithm and the synchronization method. Then, Chapter 7 presents the addition of more complex rules to increase fluidity and believability of speech. Here, the complex blending algorithms are described – blending within phonemes, stealing-frames, accentuating phonemes b, p and m, deciding between keyframes for phonemes with allophonic variations, and providing prosodromic variation by including emotion while speaking.

Finally, Chapter 8 describes the original Avatar over IP animation program written in Java. In this program, all of the lip-sync and emotion algorithms are applied, and users are able to run the program over the Internet via a chat interface. Multiple voices and avatars may be selected. Applications and future work is discussed.



Chapter 2 – Phonetic Transcription and Phoneme Alphabets

2.1 Phonetics and Phonemes

Synchronizing animation to the spoken word requires an understanding of phonetics, the study of spoken language and speech sounds. Developing a standardized way to transcribe English sounds is difficult as the English language is not written as it is spoken: there is no 1-to-1 mapping between letters and sounds. In English, there are silent letters ('knife'), combinations of the same letters that have different pronunciation ('tomb' and 'bomb'), combinations of different letters that have the same pronunciation ('see' and 'sea'), along with numerous other linguistic subtleties. To solve these pronunciation discrepancies, a standardized pronunciation guide is required. Additionally, while many words are easily differentiable for a fluent English speaker, the exact pronunciation of foreign words, or names, may be difficult to discern. Similarly, if the speaker uses a different dialect, this information is not sufficiently transcribed using written English.

Of the 26 letters in the English alphabet, 5 (a,e,i,o,u) are vowels, and the remaining 21 are consonants. In phonetics, there are around 24 consonant sounds and 19 vowel/diphthong sounds, totaling approximately 43 sounds that encompass the English language; these distinct sounds are called phonemes (Singh 1976). The exact number of phonemes varies between phoneme alphabets, as some alphabets group certain sounds together (making the alphabet smaller than 43), or have separate symbols representing variations on the stress applied to a phoneme based on its placement within a word or sentence (making the alphabet much larger than 43). Most phoneme alphabets, however, tend to have around 43 phonemes. One of the most widely known standard phoneme alphabets is the International Phonetic Alphabet.

2.2 The International Phonetic Alphabet: Phonemic versus Phonetic Transcription

By means of phonetic transcription, the International Phonetic Alphabet (IPA) provides one of the most technical ways to transcribe orthography (the written word) into phonetic representation. Phonetic transcription differs from phonemic transcription in that phonemic transcription simply combines acceptable variations of the 43 phonemes, whereas phonetic transcription provides representation of specific variations of a phoneme. Each variation of a phoneme is called an allophone, and is usually caused by the phoneme's position in a word (Singh 1976). Single consonants, produced by creating constriction in the human breath channel, are thus differentiated into prevocalic (at the start of a word), intervocalic (between two vowels) or postvocalic (at the end of a word).

Similarly, vowels and diphthongs are subdivided into preconsonantal (at the start of a word), interconsonantal (in the middle of a word), and postconsonantal (at the end of a word). For example, the phoneme /p/ in the word ‘put’ is more aspirated than the phoneme /p/ in the word ‘apple.’ Thus, the exact phonetic transcription for the phoneme /p/ in the word ‘put’ would be [ph] because it is more strongly aspirated, and the phonetic transcription for the phoneme /p/ in the word ‘apple’ would be [p] because it is unaspirated. Deliberate aspiration of the phoneme /p/ in the word ‘apple’ by a native speaker would be phonetically transcribed as [p+].

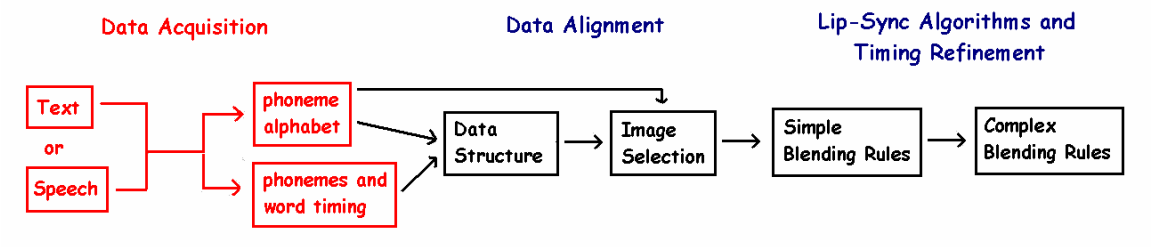
This precise representation is particularly useful in studying different dialects of a language, or the slight variations of speech in an unfamiliar language. In these cases, the IPA provides phonetic transcription as a way for one to carefully listen to sounds in order to be able to account for allophonic variations of phonemes (Bronstein 1998).

2.3 Choosing Phonemic Transcription

For the purposes of this project, phonemic transcription is utilized, allowing rules applied to just 43 phonemes to be applied to the entire English language. The rules increase in complexity when accounting for individual allophones and variations on phonemes based on their placement within a sentence. For this reason, allophonic variations are addressed during the phoneme timing refinement stage in Chapter 6, and not while obtaining speech data during the input stage. Additionally, applying animation rules to the broad phoneme description makes this project more scalable.

2.4 Phoneme Alphabets Used

Since the IPA uses non-ASCII symbols to represent phonemes and their variations, and as allophonic representation is not used during the input stage of this project, two other phoneme alphabets are utilized: WorldBet and ARPAbet, each a subset of the IPA (see Appendix A). Throughout this paper, the ARPAbet alphabet is used to represent phonemes, which will be delimited slashes. For example, the word ‘dog’ is transcribed as /d aw g/ using ARPAbet.



Chapter 3 – Lip-Sync Animation and Diphone Durations

3.1 Text-to-Speech Engines, Diphones, and Prosodromic Variation

Lip-sync animation utilizes the keyframe mapping described in Chapter 4 to synchronize video at a certain number of frames-per-second on top of natural English speech. Before this mapping is carried out, however, decisions need to be made as to which type of speech sounds should be assigned rules – should lip-sync animation rules be applied to phonemes, diphones (two phonemes), triphones (three phonemes), syllables, or morphemes (roots of words). This decision is based on looking at the trends of text-to-speech technology and exploring how it will fit with the animated face.

Numerous text-to-speech engines have been created, such as ATT Bell Labs, Microsoft TTS, Festival, and MBROLA. These engines use different methods for turning text into speech, and these methods need to be investigated before assigning lip-sync animation algorithms to the synthesized speech. For example, some text-to-speech programs use a large sound file database containing thousands of words (Pechter 1999). While these systems possess the advantage of word clarity and fixed word duration, the disadvantage lies in the difficulty of applying animation rules to sounds within words unless a separate animation is made for all of the words. Other text-to-speech engines have very large syllable and morpheme databases that can be very accurate, but suffer from the same problem: separate animations must be applied to thousands of speech segments.

However, the most commonly used method of performing text-to-speech is using a phoneme dictionary to predict phonemes and blending between diphone sounds. The concatenation of diphones by diphone blending is central to the MBROLA Project. The MBROLA project contains voice synthesizers for over one hundred dialects and languages. These voices take a list of phonemes as their input, along with prosodic information (duration of phonemes and piecewise linear description of pitch) to produce speech (“MBROLA” 2001). The voices are freely available and can be used with Festival Text-to-Speech.

There are many advantages to assigning lip-sync animation to the diphone blending system. First, there are generally only (43 phonemes)*(43 phonemes) = 1894 diphones. Thus a sound database of 2000 files is relatively small, so a 1-to-1 mapping of 2000 animations to their corresponding diphones would not be as difficult or time consuming as attempting the animation of a large database of words. Additionally, as the diphones are composed of specific phoneme cuts, and this information is readily output by the text-to-speech engine, these systems are more suitable for accurate lip-sync animation.

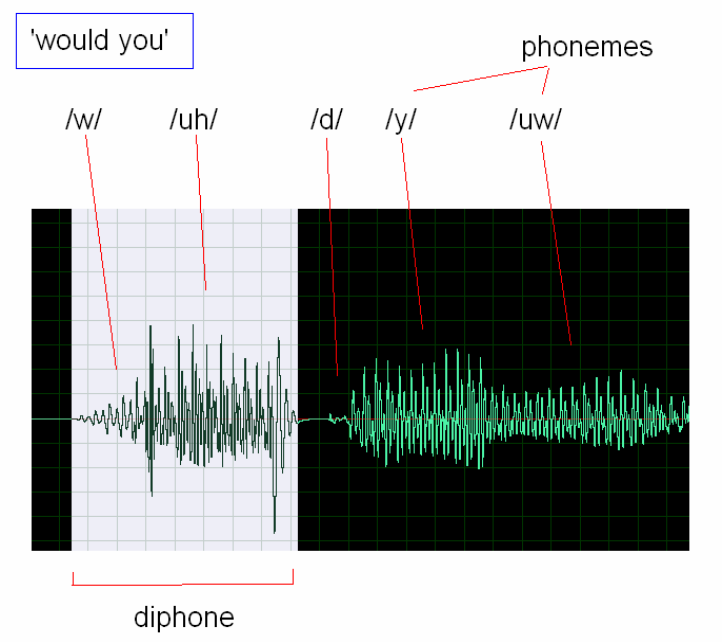


Figure 3.1. Phoneme and Diphone Wave Data

Obtaining a list of phonemes and phoneme durations is ideal for the lip-sync animation algorithm. The strategy consists of creating speech animation algorithms using only the information from the phonemes and phoneme cuts. Through this method, the lip-sync algorithm will work with text-to-speech programs with single phonemes, diphones, and morphemes, as long is there are phoneme cuts. Additionally, some freeware speech-to-text programs such as Baldisync are available for extracting phonemes and phoneme timing from wave files ("CSLU Speech Toolkit" 2003). This information makes synchronization with natural voice possible.

3.2 Interfacing Festival Text-to-Speech with Animation

In this project Festival Speech Synthesis System is used. The Carnegie Mellon University (CMU) and Oxford Advanced Learner's Dictionary (OALD) pronunciation dictionaries are used in conjunction with Festival synthesized voices to facilitate text to phoneme needs (see Figure 3.2). Text-to-speech synthesized voices use the phonemes to perform diphone blending with phonemes and phoneme duration output. Pronunciation dictionaries use different phoneme alphabets (see Table 3.1).

Table 3.1. Phoneme Alphabets and Pronunciation Dictionaries

Pronunciation Dictionary	**Phoneme Alphabet Used**	**English Variety**
CMU Dictionary	ARPAbet	American Standard English
OAL Dictionary	Radio	British
Moby Dictionary	WorldBet	American Standard English
Celex	IPA	British

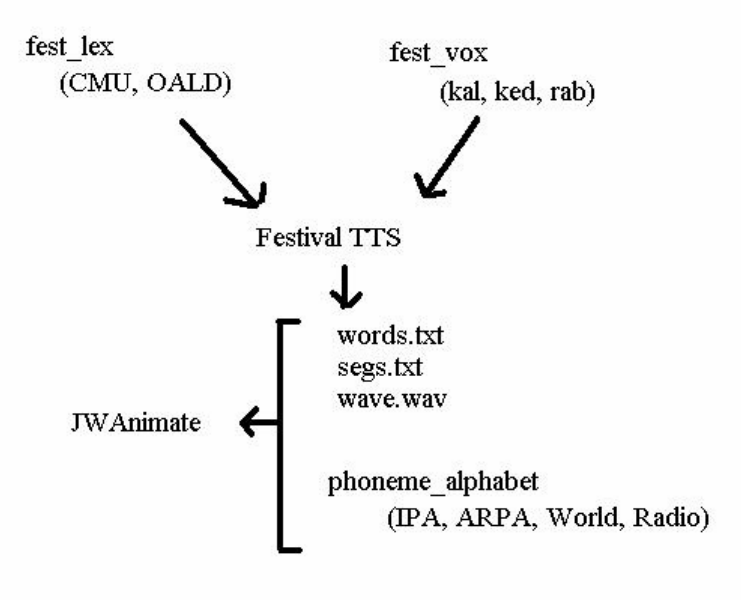
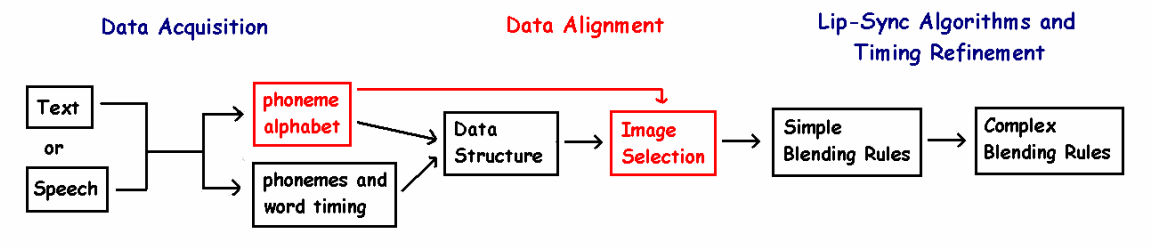


Figure 3.2 Phoneme Data to JWAnimate



Chapter 4 – Articulation of Phonemes and Choosing Eight Keyframes for Animation

4.1 Keyframe Assignment

Once phonemes and word timings have been extracted, phonemes are assigned to images in the image selection stage. The keyframes of the lip-synchronization animation are chosen with two main criteria in mind: speed and aesthetics. Based on these criteria, a small set of eight simple 2D drawings is chosen. Each keyframe shows only subtle tongue and teeth placement, facilitating blending between keyframes. A careful examination of the physical articulation of phonemes is used in deciding which keyframe to assign to each phoneme in the animated lip-sync of any word. The phoneme to keyframe mapping is summarized in Figure 4.1. Keyframe 3 is used most often as it represents the most versatile lip shape.

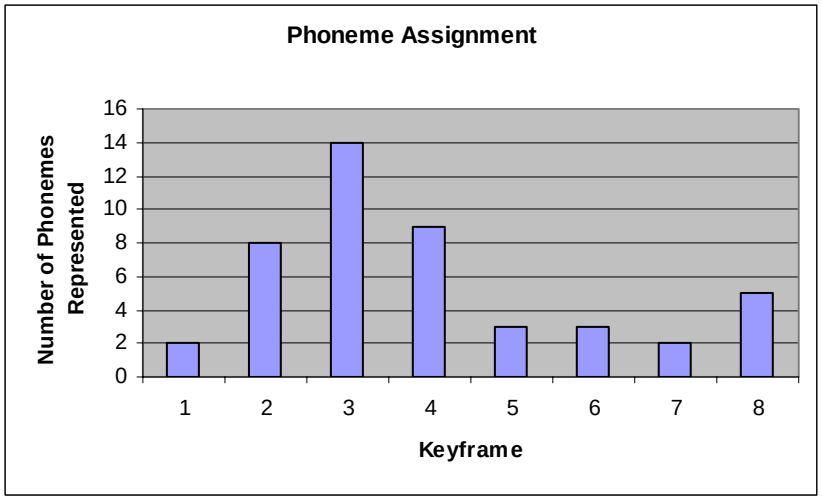


Figure 4.1 Phoneme Assignment

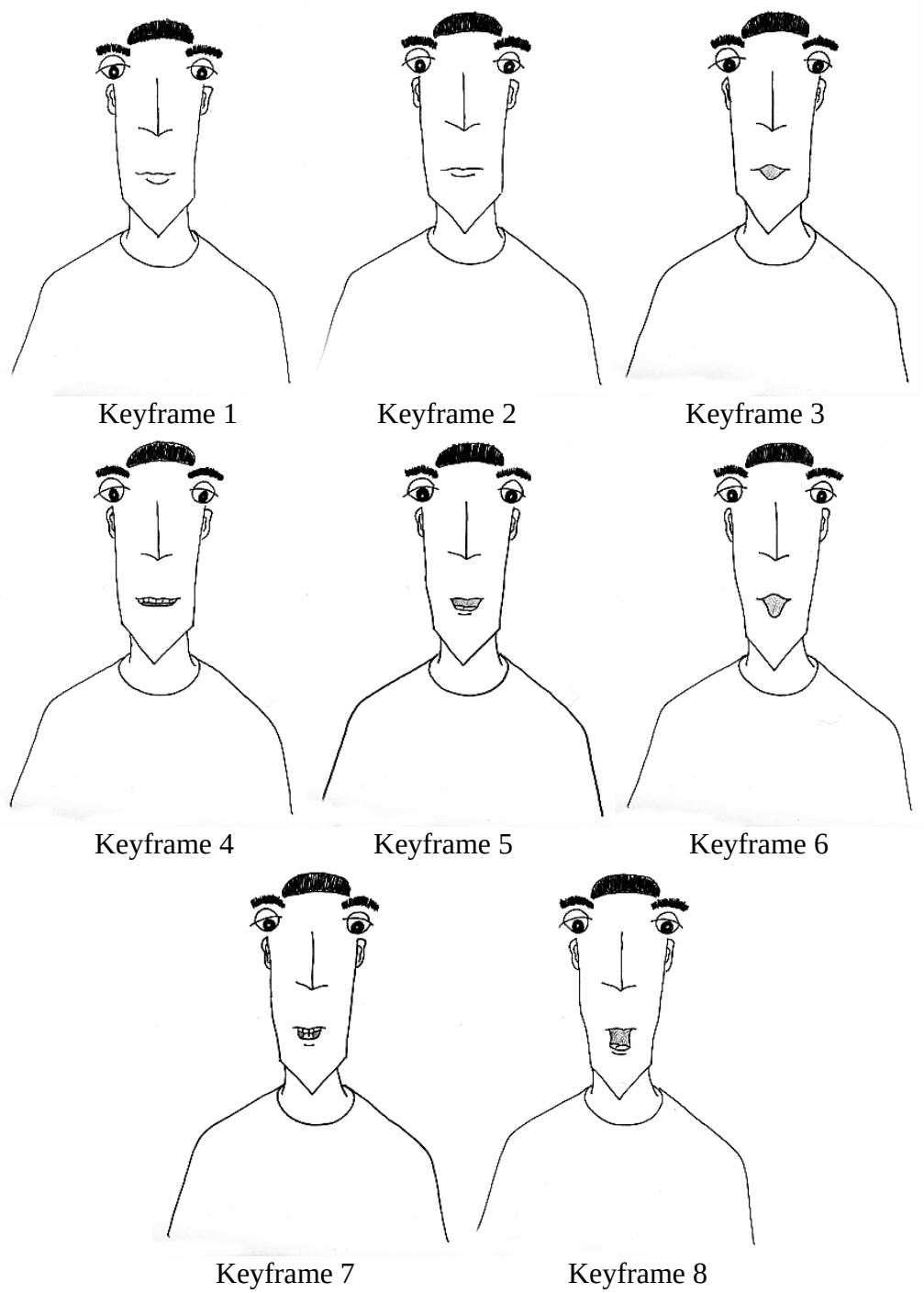


Figure 4.2. Keyframes

4.2 Articulation of Consonants

In order to map a small set of keyframes to consonant sounds, the phonemes are categorized based on the contact points and mouth shape during articulation.

Consonants have six main places of articulation within the mouth: bilabial (lips touching) labiodental (upper teeth touching lower lip), linguadental (tongue in between upper and lower teeth), alveolar (tongue anterior to upper teeth), palatal (tongue touching the hard palat), and velar (tongue touching the velum) (Singh 1976). These six contact points do not encompass all the variations for consonant phonemes; consonants pronounced in different English dialects and in other languages may be articulated in other contact points. In fact, certain phonemes in Arabic have a contact point deep down into the throat, where the tongue touches below the velum. Phoneticians believe that humans are capable of making as many as twenty six articulatory contact points (Singh 1976). Together, these account for the places of articulation of all the languages of the world. In order to create a keyframe set for a language other than English, the twenty six contact points can be grouped into the nearest key and still provide effective lip-sync, rather than using additional keyframes. Tests for this method of mapping keyframes to contact points of different languages is a future project.

However, for the purposes of this project, the six main points of articulation described above are considered when assigning keyframes to consonant phonemes, along with the exterior variations in lip formation.

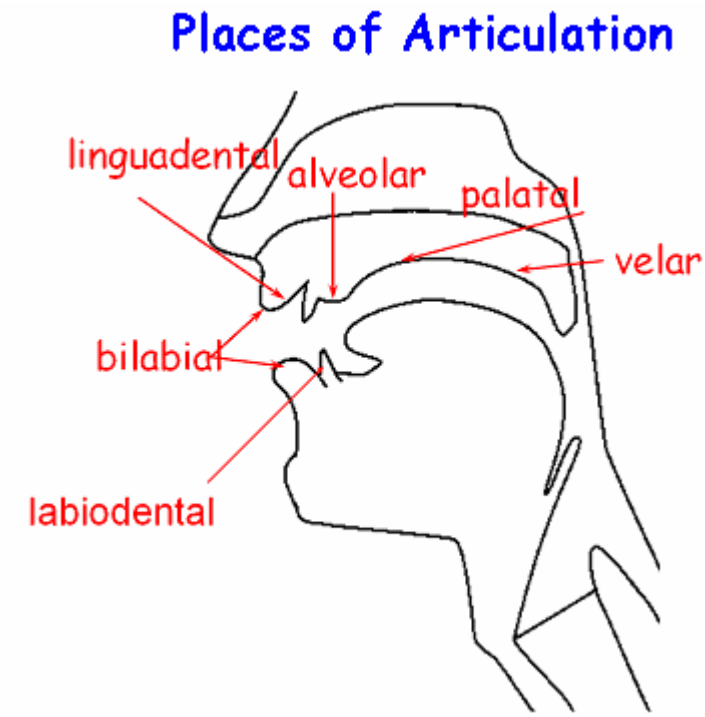


Figure 4.3. Places of Articulation (Hall 2004)

Table 4.1. Consonant Phoneme-to-Keyframe Mapping

Place of Articulation	ARPAbet Phonemes Represented	Matching Keyframe Set
Bilabial	(p, b, m, w)	(2, 2, 2, 3)
Labiodental	(f, v)	(2, (2 or 4))
Linguadental	(th, dh)	(2, 7)
Alveolar	(t, d, n, s, z, l, r)	(2, (2 or 4), 2, 4, 4, (3 or 7), 3)
Palatal	(sh, zh, ch, jh, y)	(3, 4, 3, 3, 3)
Velar	(k, g, h, ng)	(4, 4, 3, 4)

*Note – Choosing between keyframes for consonants /d/, /l/, and /v/ is discussed in chapter 7

4.3 Articulation of Vowels

In order to map a small set of keyframes to vowel sounds, the phonemes are categorized based on the position of the tongue within the mouth during articulation. Vowels have nine main places of articulation within the mouth. These nine points of articulation are considered when assigning keyframes to vowel phonemes, along with the exterior variations in lip formation.

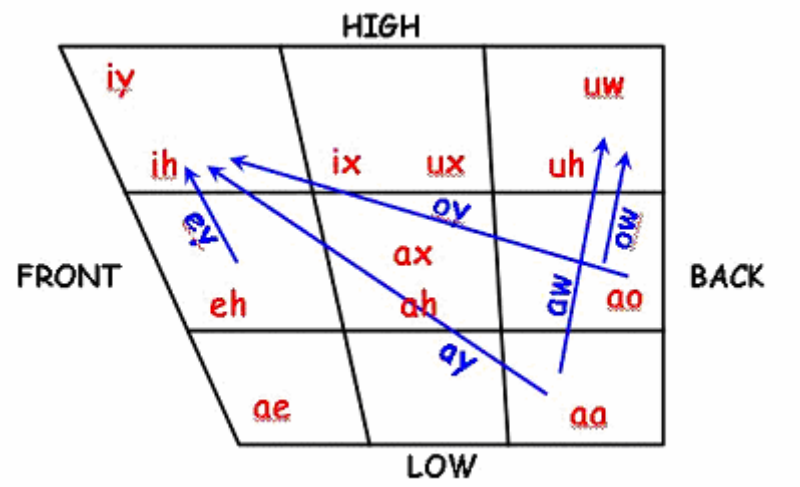


Figure 4.4. American English Vowel Space (Venditti 2002)

4.4 Articulation of Diphthongs

When matching the set of keyframes to diphthong sounds, the diphthongs are assigned one keyframe in some instances, and two keyframes in other instances. One keyframe is used when not enough time is available to perform blending within the phoneme. For example, if the sentence 'I love you' is spoken quickly, the word 'I' corresponding to the phoneme /ay/ may only have two frames designated to this sound. In this case, only keyframe 8 is assigned to this phoneme during the image selection stage. On the hand, if 'I love you' is spoken slowly, the phoneme /ay/ may have six frames designated to this sound. In this case, both keyframes 8 and 3 are assigned to this phoneme during the image selection stage.

Table 4.2. Vowel and Diphthong Phoneme-to-Keyframe Mapping

Diphthong	One Keyframe Mapping	Two Keyframe Mapping
AW	6	6 and 3
AX	6	6 and 3
AY	8	8 and 3
EW	8	8 and 5
OW	3	3 and 1
OY	3	3 and 1

4.5 Creating Strip Files with In-Betweens

Strip-files are image files containing multiple frames in order to blend one keyframe into another during the lip-sync animation. The strip-files used in this project are generated automatically using an optical flow morphing algorithm (Periaswamy and Farid 2003). The strip-files may also be generated manually by drawing the in-betweens between each of the 8 keyframes.

One strip file is needed between any two keyframes; thus, for these 8 keyframes, 28 strip files are needed, and in general,

$$\sum_{n=1}^{keyframes-1} n$$

is the total number of strip files required for a given number of keyframes.

For this project, four in-between images are used in each strip file, resulting in a final strip file consisting of six images.

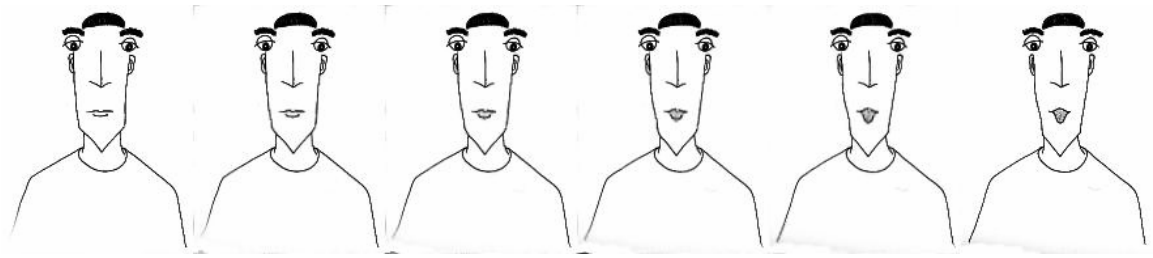
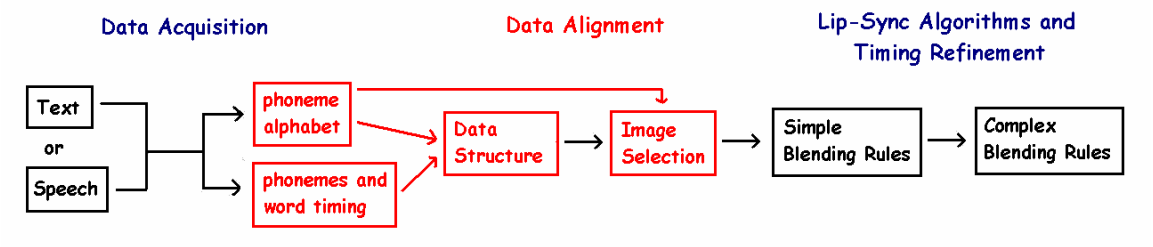


Figure 4.5. Strip-File

Table 4.3. Summary of Phoneme-to-Keyframe Mapping

	ARPABet	Example	Phonemic Transcription	Keyframe Mapping
1	AA	odd	AA D	8
2	AE	at	AE T	8
3	AH	hut	HH AH T	5
4	AO	ought	AO T	8
5	AW	cow	K AW	6 or (6 and 3)
6	AX	about	AX B AW T	6 or (6 and 3)
7	AY	hide	HH AY D	8 or (8 and 3)
8	B	be	B IY	2
9	CH	cheese	CH IY Z	3
10	D	dee	D IY	4 or 2
11	DH	thee	DH IY	7
12	EH	Ed	EH D	5
13	ER	hurt	HH ER T	3
14	EY	ate	EY T	8 or (8 and 5)
15	F	fee	F IY	2
16	G	green	G R IY N	4
17	HH	he	HH IY	3
18	IH	it	IH T	5
19	IY	eat	IY T	4
20	JH	gee	JH IY	3
21	K	key	K IY	4
22	L	lee	L IY	7 or 3
23	M	me	M IY	2
24	N	knee	N IY	2
25	NG	ping	P IH NG	4
26	OW	oat	OW T	3 or (3 and 1)
27	OY	toy	T OY	3 or (3 and 1)
28	P	pee	P IY	2
29	R	read	R IY D	3
30	S	sea	S IY	4
31	SH	she	SH IY	3
32	T	tea	T IY	2
33	TH	theta	TH EY T AH	2
34	UH	hood	HH UH D	3
35	UW	two	T UW	6
36	V	vee	V IY	4 or 2
37	W	we	W IY	3
38	Y	yield	Y IY L D	3
39	Z	zee	Z IY	4
40	ZH	seizure	S IY ZH ER	4
41	PAU	-	** This is a silent “pause” phoneme	“blink” emotion



Chapter 5 – Using X-sheets and Loading into the Master Data Structure

5.1 Parsing Phoneme and Word Data Files

Data files containing information from the text-to-speech engine need to be parsed for use with the animation. After a sentence is sent to the text-to-speech engine, three data files are obtained: a file with words and word-durations; a file with phonemes and phoneme-durations; and a wave file. The wave file contains the audio representation of the input sentence. The word-timing file contains a list of all the words in the sentence, and the time (seconds) in the wave file that each word ends.

```
#
0.3893 Would
0.5658 you
0.8181 like
0.9473 to
1.2219 see
1.2590 a
1.5993 movie
1.7339 with
2.0172 me
```

The phoneme-timing file contains a list of all the phonemes in the sentence, and the time (milliseconds) in the wave file that each phoneme ends.

```
#
0.2200 pau
0.2771 w
0.3367 uh
0.3893 d
0.4396 y
0.5658 uw
0.6372 l
0.7608 ay
0.8181 k
0.8963 t
0.9473 ax
1.0875 s
1.2219 iy
1.2590 ax
1.3473 m
1.4645 uw
```

```
1.5144 v
1.5993 iy
1.6576 w
1.7010 ax
1.7339 dh
1.8064 m
2.0172 iy
2.4660 pau
```

Parsing the word-timing and phoneme-timing files requires breaking down the files into four lists; a list of words (type String), a list of phonemes (type String), a list of word-timings (type double), and a list of phoneme-timings (type double).

Different types of word-timing and phoneme-timing files are produced for different text-to-speech engines. For example, Festival outputs phonemes and phoneme-timing into two columns, but MBROLA outputs in a different way. Thereby, files are parsed slightly differently, and the type of program must be stipulated before initializing the parsing process. Fortunately, the four lists described above represent are all the information required for the rest of the algorithm.

5.2 X-Sheets and Aligning the Data

As the algorithm requires a lot of data mapping, X-sheets are useful to visualize the data and align it to a usable form. Two main X-sheets are created using the four lists parsed from the data (described in 5.1). The first X-sheet has columns containing: which *phoneme_alphabet* is used (type String); the *phoneme* (type String); the *phoneme_timing* (type double); the *phoneme_duration* (type double); the *num_frames* assigned (type int), and the *keyframe_assigned* (type int). The second X-sheet contains columns for *words* (type String), *word_timings* (type double), *word_durations* (type double), *num_frames* (type int), and *emotion_mapping* (type int).

Table 5.1. X-Sheet 1

phoneme_alphabet	phoneme	phoneme_timing	phoneme_duration	Num_frames	keyframe_assigned
ARPAbet	pau	0.22	0.22	7	Blink
ARPAbet	w	0.2771	0.0571	2	3
ARPAbet	uh	0.3367	0.0596	2	3
ARPAbet	d	0.3893	0.0526	2	4
ARPAbet	y	0.4396	0.0503	2	3
ARPAbet	uw	0.5658	0.1262	4	6
ARPAbet	l	0.6372	0.0669	2	7
ARPAbet	ay	0.7608	0.1236	4	8
ARPAbet	k	0.8181	0.0573	2	4
ARPAbet	t	0.8963	0.0782	2	2
ARPAbet	ax	0.9473	0.051	2	6
ARPAbet	s	1.0875	0.14	4	4
ARPAbet	iy	1.2219	0.1344	4	4
ARPAbet	ax	1.259	0.0371	1	6

ARPAbet	m	1.3473	0.0883	3	2
ARPAbet	uw	1.4645	0.1172	4	6
ARPAbet	v	1.5144	0.0469	1	4
ARPAbet	iy	1.5993	0.0849	3	4
ARPAbet	w	1.6576	0.0583	2	3
ARPAbet	ax	1.701	0.0434	1	6
ARPAbet	dh	1.7339	0.0392	1	7
ARPAbet	m	1.8064	0.0752	2	2
ARPAbet	iy	2.0172	0.2108	6	4
ARPAbet	pau	2.466	0.4488	14	Blink
			Total:	77	

Table 5.2. X-Sheet 2

phoneme_alphabet	word	Word_timing	word_duration	Num_frames
ARPAbet	Would	0.3893	0.3893	12
ARPAbet	you	0.5658	0.1765	5
ARPAbet	like	0.8181	0.2523	8
ARPAbet	to	0.9473	0.1292	4
ARPAbet	see	1.2219	0.2746	8
ARPAbet	a	1.259	0.0371	1
ARPAbet	movie	1.5993	0.3403	10
ARPAbet	with	1.7339	0.1346	4
ARPAbet	me	2.0172	0.2833	22

*Note – first and last words include frames for the pause phonemes

The *keyframe_assigned* column in the first X-sheet is obtained by using a static array (the size of the *phoneme_alphabet* used) that assigns phonemes to keyframes. A function linearly takes each phoneme on the X-sheet as input, and fills in the corresponding keyframe in the keyframe column. Rules governing this phoneme-to- keyframe mapping are explained in Chapter 4.

The *phoneme_duration* column is obtained by aligning the *phoneme_timing* column. In the *phoneme_timing* column (the list obtained by the parsing step (5.1)), only the time (milliseconds) in the wave file that each phoneme ends is given. The useful phoneme_duration column contains the time in milliseconds that each phoneme is held. Subtracting one from adjacent rows in the *phoneme_timing* column allows the *phoneme_duration* data to be filled.

```
for (int j = 1; j < phoneme_timing.length; j++)
        phoneme_duration[j] = phoneme_timing[j] – phoneme_timing[j-1]
```

Finally, the *num_frames* column contains the exact number of frames assigned to animate each phoneme. Multiplying each element of the *phoneme_duration* column by the *frames_per_second* fills in the *num_frames* column. No element can be rounded to zero frames.

```
for (int j = 0; j < phoneme_duration.length; j++)
        num_frames[j] = (int) Math.round(phoneme_duration[j] * frames_per_second)
        if (num_frames[j] == 0)
                num_frames == 1;
```

The *num_frames* column is one of the most important in the animation algorithm as it gives the number of frames to work with for each phoneme. Thus, the rounding of *num_frames* in this function may need to be more accurate in the future. Possibly, a tally may keep track of a fractional number of frames as rounding occurs, and using this tally, frames may be added or subtracted.

5.3 Master Data Structure

For simplifying the implementation of the rest of the algorithm, and specifically in order to more easily insert emotions while speaking, the X-sheet is loaded into the Master Data Structure. This Master Data Structure is specifically designed for ease of use with speech rules, blending, emotion rules, and playing the finished movie. Classes are broken down as follows:

ParserLoader
myVideoFrame
myPhone
myWord

Parsing of input files, building X-sheets, and loading into the Master Data Structure is performed by the *ParserLoader* class. Building using the X-sheet starts with creating three doubly-linked-lists on top of each other. First, a doubly-linked list of *myVideoFrames* is created; the total number of frames in the animation is calculated by adding up the *num_frames* column. Next, a doubly-linked list of *myPhones* is created using the *phonemes* column of the X-sheet. Each *myPhone* is given a pointer to the start frame and to the end frame by moving through the list of *myVideoFrames* and using the *num_frames* column of the X-sheet. Finally, a doubly-linked-list of *myWords* is created using the *words* column of the second X-sheet, the *num_frames* column of the first X-sheet, and the *num_frames* column of the second X-sheet (see Figure 5.1).

Most importantly, this data structure directly corresponds to the synchronization of phoneme sounds to video frames. Additionally, following the pointers in this data structure allows one see where emotions can be animated for words. Changing the pointer assignment allows for easy insertion and deletion of frames, as well as changing the starting and ending frames for phonemes (and thereby words). Thus, rules at each level (frames, phonemes, and words) can be easily and accurately applied. Also, playing the finished movie is performed by simply traveling through the *myVideoFrames* at a number of *frames_per_second*. The runtime to parse, align in X-sheet, and load into data structure is O(numwords+numsegs).

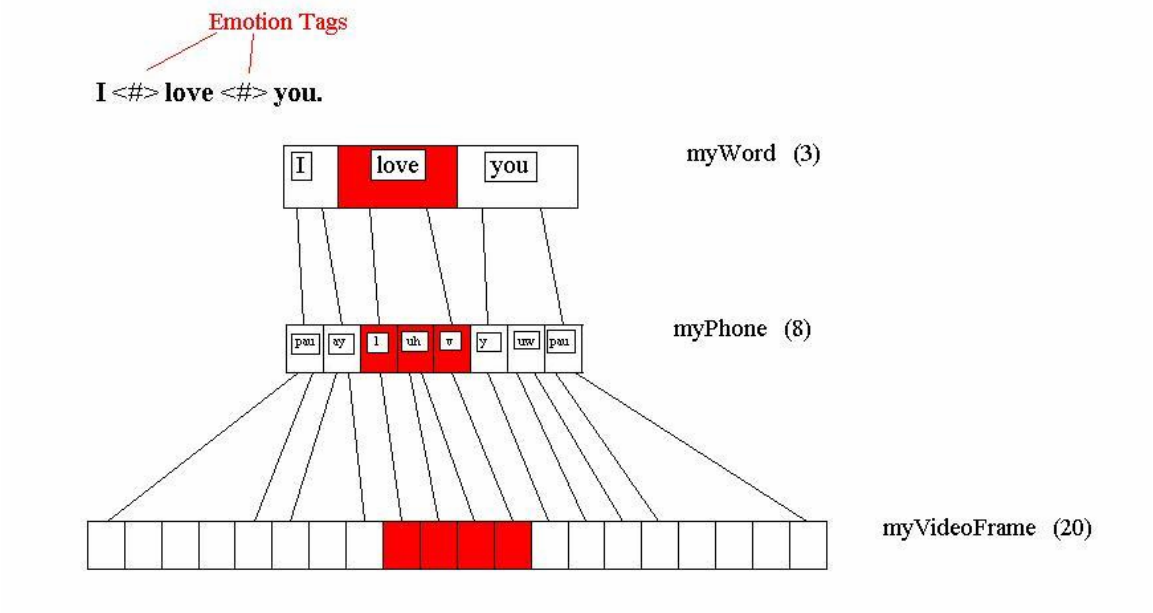
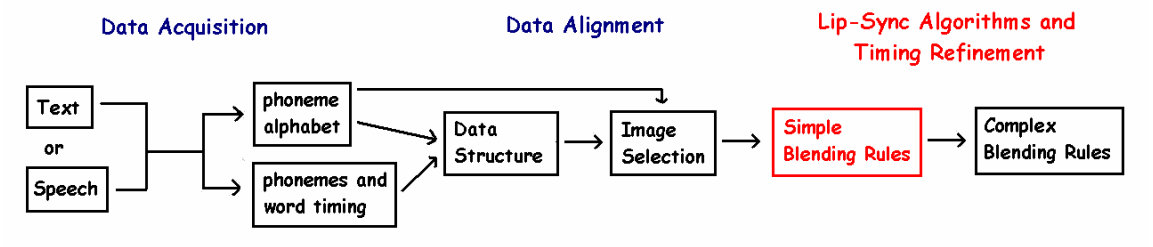


Figure 5.1. Master Data Structure



Chapter 6 – The Simple Blending Function – Smoothing Transitions between Keyframes

Once the image selection stage is completed, lip-sync is accurate but not very believable. For example, after the sentence 'I love you' goes through the image selection stage, keyframe assignment and timing data is available:

Table 6.1. Keyframe Assignment to "I love you"

Phoneme	Keyframe Assigned	Frames to Hold
Ay	8	4
L	7	3
Ah	5	5
V	4	2
Y	3	2
Uw	6	6

Phoneme-to-keyframe mapping is performed and keyframes are simply held for the duration of each phoneme (see Figure 6.1).

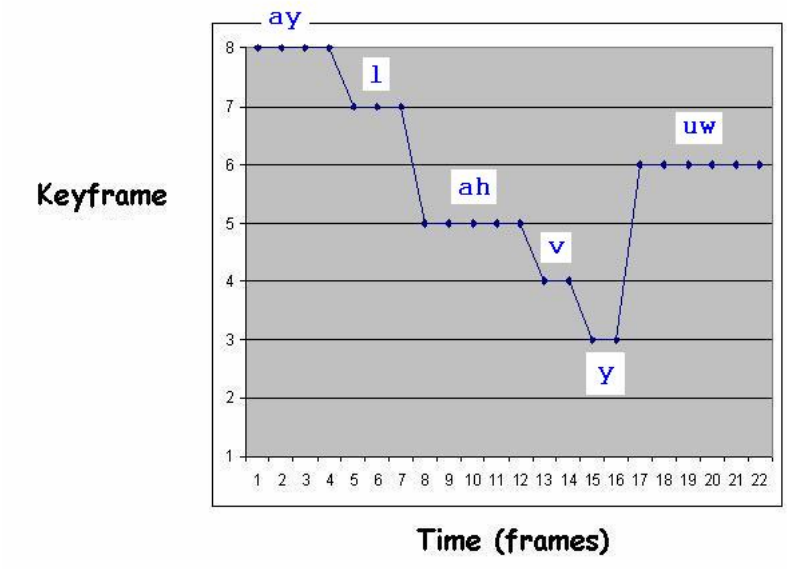


Figure 6.1. Keyframe Assignment and Frame Durations for "I love you"

This animation is synchronized with the audio, but it is clearly choppy; transitions between phonemes are too sharp. Thereby, simple blending rules are needed to increase the fluidity of the transitions.

Once the parsing and loading of the empty data structure have been performed, rules begin for simple blending. For the simple blending, each phoneme is first mapped to one keyframe ([hany1, hany8]). Next, the duration for each phoneme in seconds is rounded off to the nearest integer number of frames for each phoneme (num_frames). Diphone blending is performed by taking first_keyframe and its num_frames and applying it to the Blend function with the call int[] result = Blend(first_keyframe, next_keyframe, num_frames). For the last phoneme in a sentence, or before a /pau/ (pause phoneme) blend back to the first keyframe by calling int [] offsets = Blend(first_keyframe, 1, num_frames). The integer array of offsets and the filename will be used to obtain the frames when playing the animation. The filename and offset are given to each myVideoFrame. At the end of the rules, the animation is played by simply moving through the myVideoFrames at the number of frames_per_second.

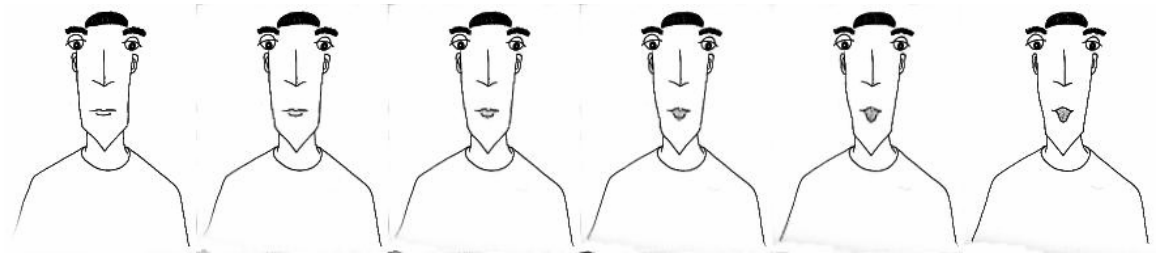


Figure 6.2. Strip File Offsets

```
int[] Blend(int first_keyframe, int next_keyframe, int num_frames)
{
 String filename ="hany"+first_keyframe+"_"+"hany"+next_keyframe+".JPG";

 int remainder = num_frames % (STRIP_SIZE-1);
 int quotient = frames / (STRIP_SIZE-1);

  if (frames <= 1)          {int[] res = {0}; return res;}
  else if (frames == 2)     {int[] res = {0,3}; return res;}
  else if (frames == 3)     {int[] res = {0,2,4}; return res;}
  else if (frames == 4)     {int[] res = {0,2,3,4}; return res;}
  else if (frames == 5)     {int[] res = {0,1,2,3,4}; return res;}
  else if (frames == 6)     {int[] res = {0,0,1,2,3,4}; return res;}
  else if (frames > 6)
  {
   int[] res = new int[frames];
   int count = 0;
   for (i = 0; i < INBETWEENS; i++) {
      for (j = 0; j < quotient; j++) {
       res[count] = i;
       count++;
      }
    if (remainder != 0)
      {
            res[count] = i;
```

```
                --remainder;
                count++;
            }
        }
        return res;
    }
    else {int[] res = {}; return res;}
}
```

Using this simple method, extremely accurate synchronization and smooth blending between keyframes is obtained. For each diphone, a strip file is obtained which contains in-betweens from the first_keyframe to the next_keyframe. The blending algorithm obtains the movie frames by selecting the frames which will be used for smooth transition towards the next_keyframe. The frame at index 5 will never be reached; instead, the next_keyframe will start with its own index 0 in the strip file. This method prevents duplication of frames when moving between phonemes.

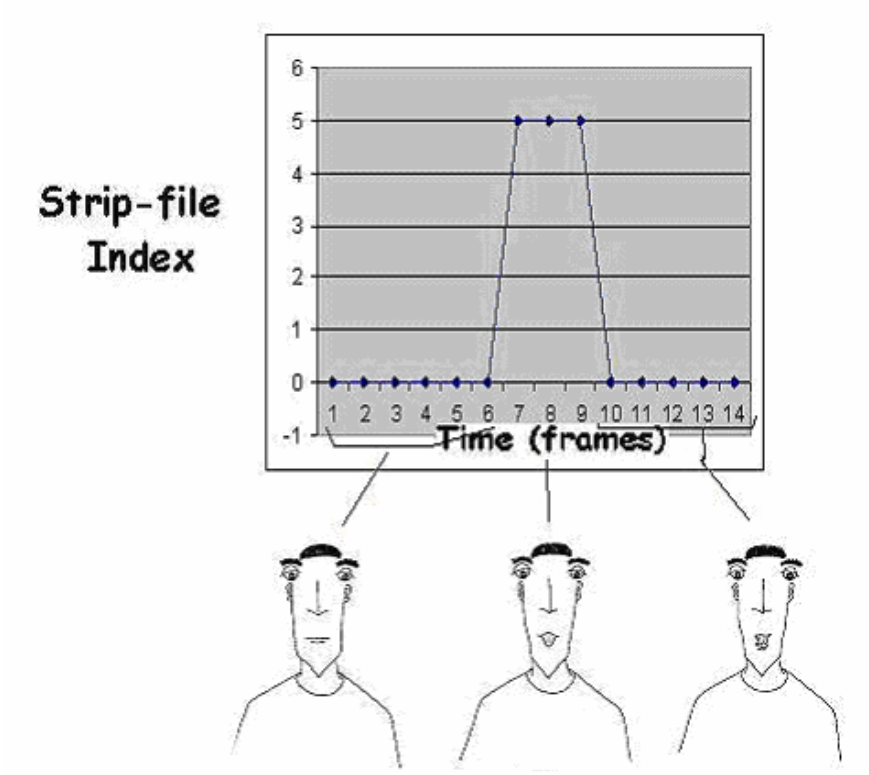


Figure 6.3. Before Simple Blending

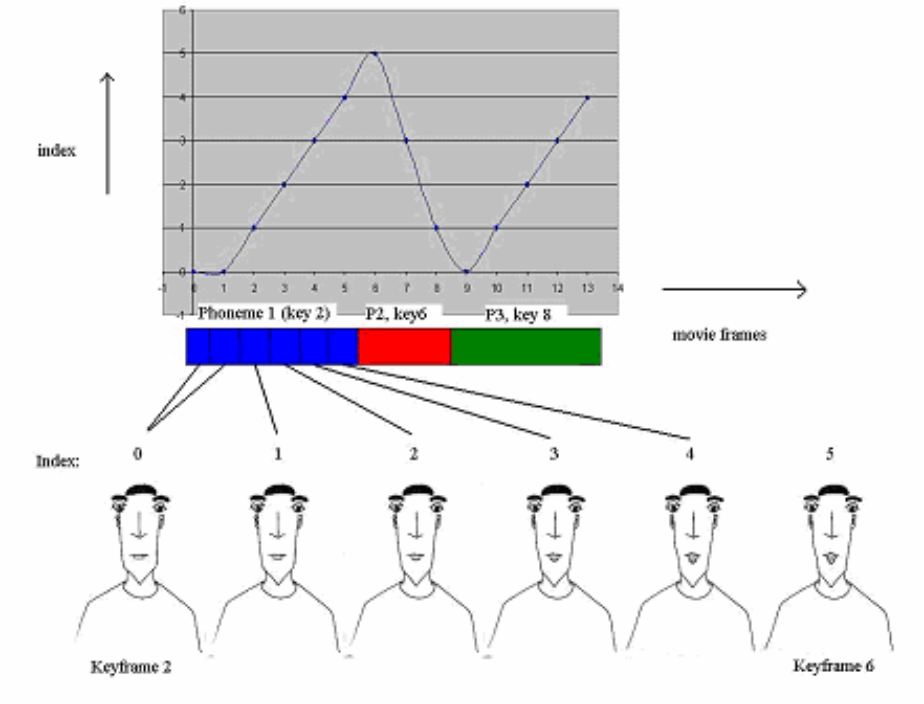


Figure 6.4. After Simple Blending I

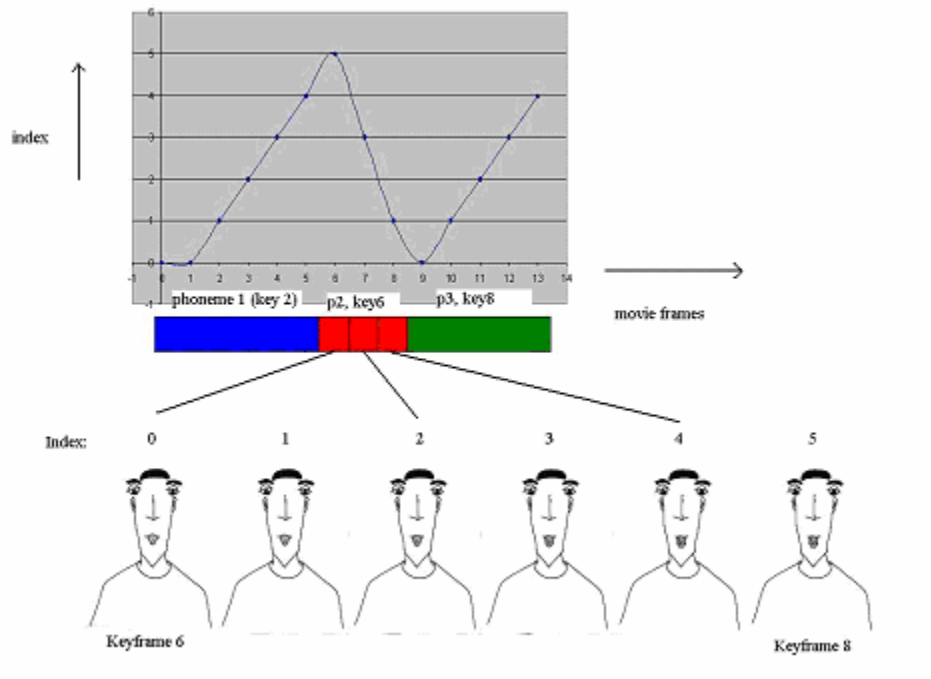
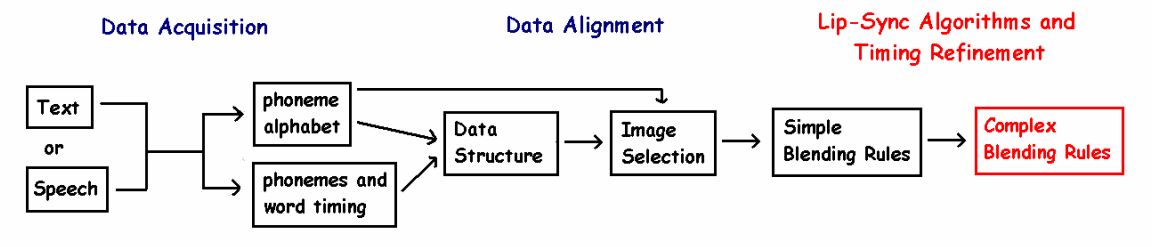


Figure 6.5. After Simple Blending II



Chapter 7 – Complex Blending Rules

7.1 Blending Within Phonemes – Assigning Multiple Keyframes to Diphthong Phonemes

After having achieved simple blending between phonemes, the believability of the facial animation is still not sufficient. Thereby, additional rules are implemented that clarify diphthong phonemes, take allophonic variation into consideration, provide prosodromic variation, and offer stricter timing refinements.

With the simple blending rules, diphthong phonemes are mapped to single keyframes. However, diphthongs are composed of the blending of two distinct sounds. For better synchronization and more realistic speech, two keyframes are mapped to each diphthong. For example, the phoneme /ay/ as in /hh ay d/ ('hide') is a combination of the vowels /aa/ (as in 'odd') and /ih/ (as in 'eat'). The keyframes assigned to /ay/ are 8 and 3, so blending within this phoneme should hit both of these keyframes within the num_frames allowed. By using the two separate keyframes, the diphthong phoneme is clarified, and the believability of the animation is significantly improved. However, if too few frames exist for this transition to look fluid, then only one keyframe is assigned (in this case keyframe 3).

```
if (num_frames <=2)
        Blend(keyframe 3, keyframe 1, num_frames);
else
        Blend(keyframe 3, keyframe 1, 2);
        Blend(keyframe 3, next_keyframe, num_frames – 2);
```

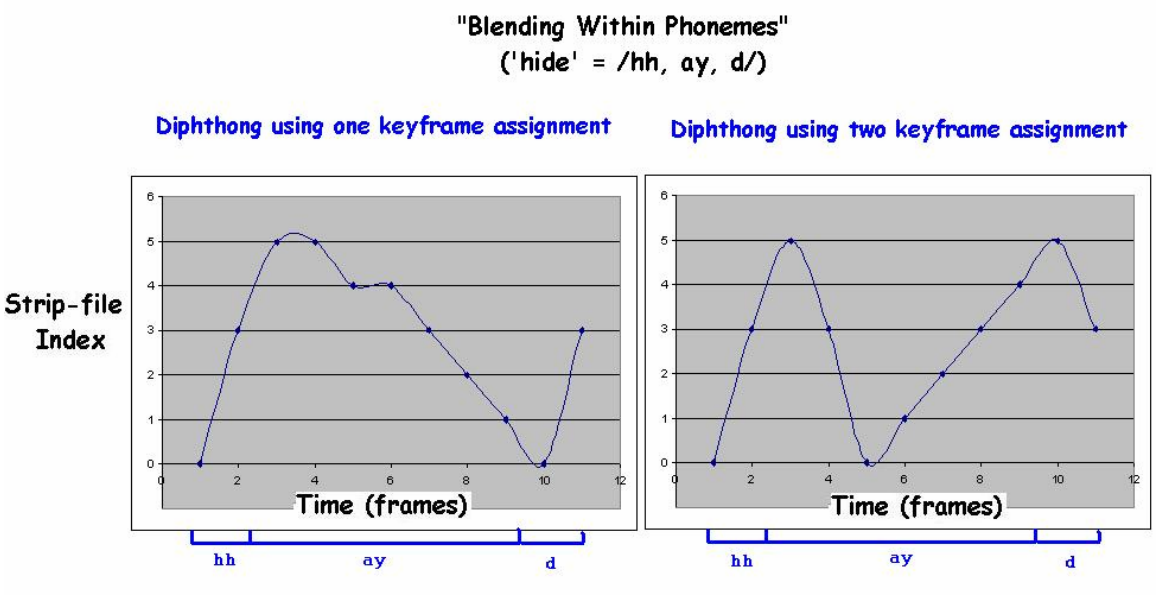


Figure 7.1. Blending within Phonemes for “hide”

7.2 Modifying/Grouping Blending Rules

Assigning multiple keyframes to certain phonemes adds complexity to the blending rules, thereby increasing the fluidity of phoneme transitions. It is then useful to group the phonemes in a phoneme alphabet into three categories: open mouths (mostly vowels), closed mouths (mostly consonants), and diphthongs. Single keyframe assignment goes to vowels and consonants, while double keyframe assignment goes to diphthongs. In most phoneme alphabets, six phonemes are assigned multiple keyframes – on the X-sheet these phonemes are mapped to the numbers 10-15. When blending rules are applied phoneme by phoneme, looking ahead to blend towards the next phoneme is slightly more complicated. For example, the word ‘boy’ has the phonemes /b oy/ and is mapped to (keyframe 2, key 10 [keyframe 3 to keyframe1]). Since the first phoneme is a consonant with one keyframe, and the next phoneme is a diphthong with two keyframes, one must first blend keyframe 2 to keyframe 3 within the num_frames for /b/, then blend within the phoneme /oy/. A similar problem occurs when blending from one diphthong to another. Thus eight functions are created to deal separately with all transitions between the three categories of phonemes (vowels, consonants, and diphthongs).

7.3 Adding Prosodromic Variation

Additionally, most text-to-speech engines provide a pause phoneme /pau/ that is useful for delaying speech between sentences. Most sentences begin and end with pauses, and pauses also occur after commas. In the simple blending, the /pau/ phoneme was assigned the keyframe 1, a neutral closed face. However, it is useful to add blending within this phoneme for such expressions as blinking or other mid-sentence emotions, adding prosody to the animation. The method for blending is similar to the diphthongs above

An example demonstrating blending within phonemes for pauses and diphthongs for the sentence ‘I love you’ /pau, ay, l, ah, v, y, uw, pau/ is shown in Figures 7.2 and 7.3.

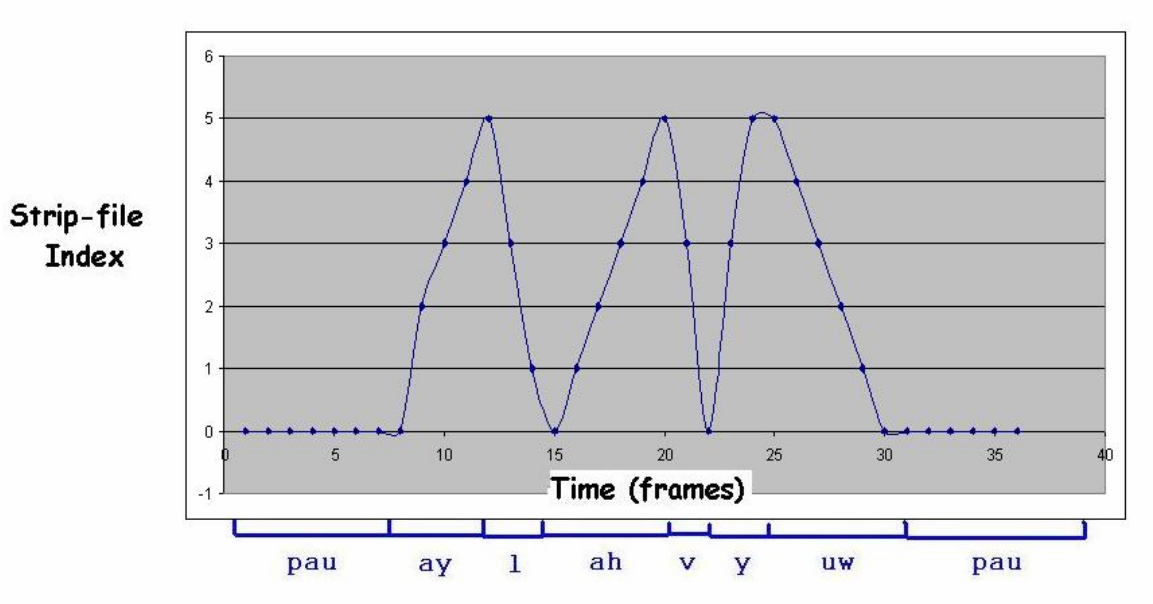


Figure 7.2. Frame Durations and Strip File Offsets for “I love you” using Simple Blending and without prosody

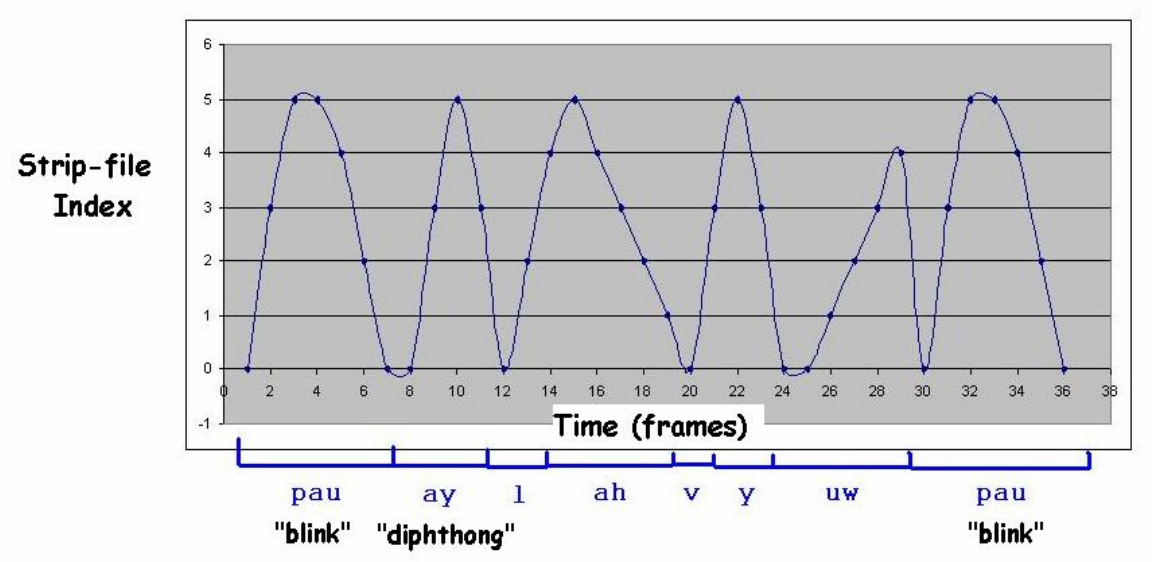


Figure 7.3. Frame Durations and Strip File Offsets after Addition of Blending within Phonemes and Prosody

7.4 Stealing Frames

Although the blending described so far is synchronized with the sound and appears to provide excellent transitions from phoneme to phoneme, problems occur when multiple phonemes in a row are mapped to the same keyframe. For example, the words 'hit tom' (as in 'he hit tom with the ball') are collectively composed of the phonemes /h ih t t aa m/ and are mapped to the keyframes (3, 5, 2, 2, 8, 2) respectively. In between the words 'hit' and 'tom', the same phoneme /t/ occurs twice. Previously, the keyframe 2 would be held for the 't' in 'hit.' In order to reduce the number of frames the keyframe is held, a method of stealing frames is used so that the same keyframe is not held very long. In the example above, the extra frames from the 't' in 'hit' can be assigned to the word 'tom' so that more frames are given for the blending from /t/ to /aa/.

The method stealing frames is provided in the blend function. When the blend function is called on phonemes that map to the same keyframe, pointers to the start and end frame are changed for each phoneme. The first phoneme receives only one frame, and the next phoneme is assigned the rest of the frames. Additionally, the number *framelength* of the second phoneme is incremented. When the blend function is then called for the second phoneme, more frames are available to start blending to the next phoneme.

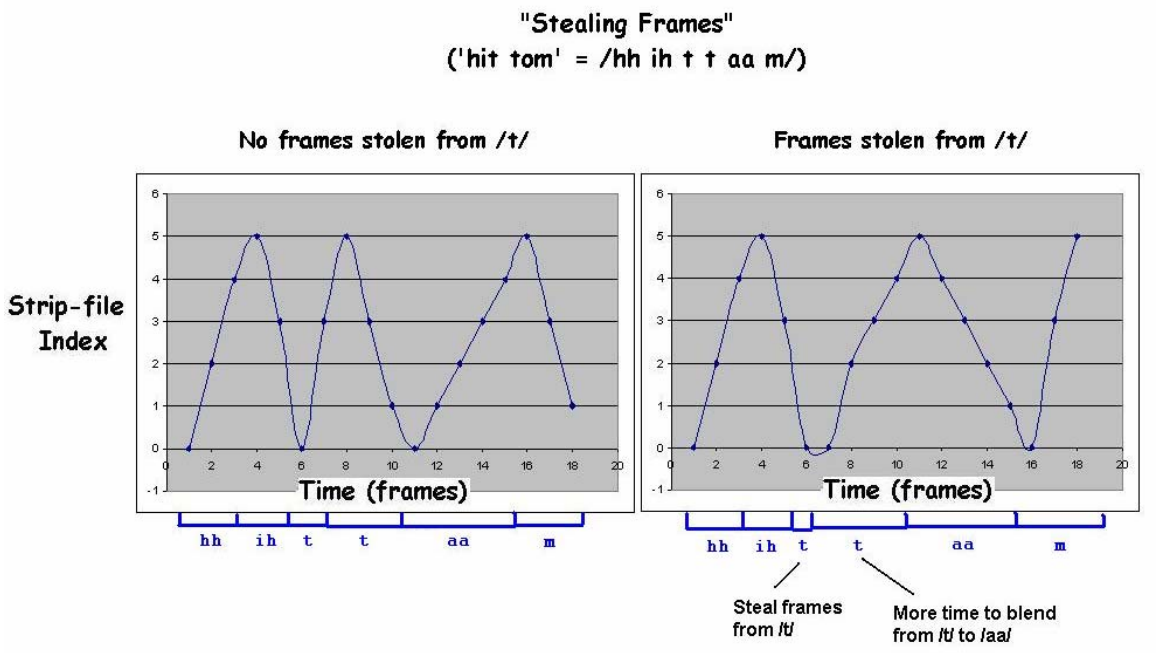


Figure 7.4. Stealing Frames

7.5 Accentuating Phonemes /b/, /p/, and /m/

The phonemes /b/, /p/, and /m/ are consonants (closed-mouth phonemes), such as in the words 'boy', 'pig', and 'man.' For these prevocalic consonant phonemes, words are

more accurately animated if the keyframe 2 is held for at least two frames (at 30 frames_per_second) before blending to the next phoneme. Hitting these phonemes for enough time thus improves the believability of the animation.

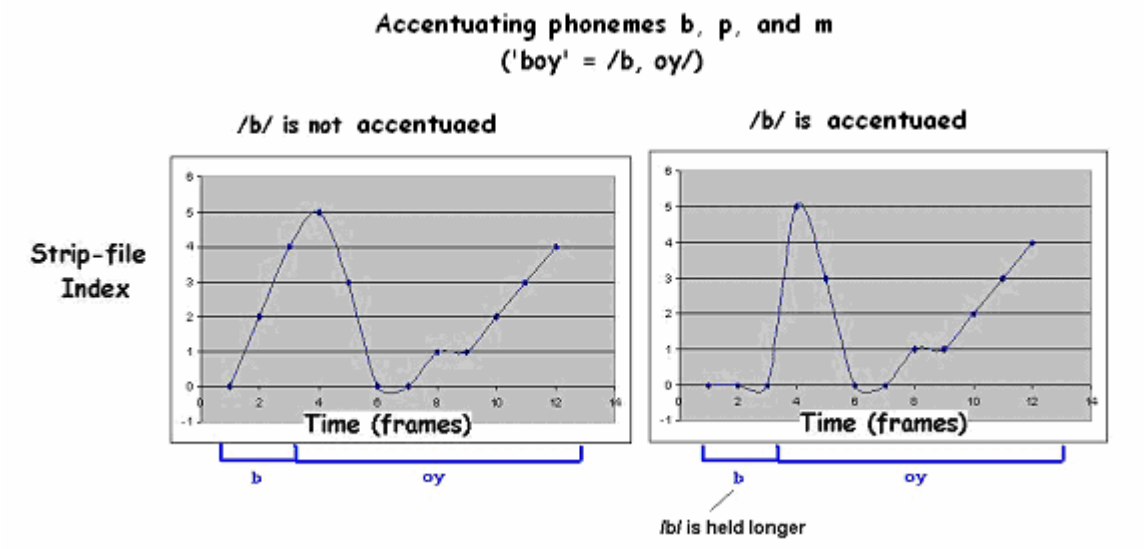


Figure 7.5. Accentuating Phonemes b, p, and m

7.6 Allophonic Variation and the Decision Algorithm for Phonemes /d/, /l/, and /v/

Allophonic variation – phonetic variation caused by a phoneme's position in a word – is addressed for the phonemes /d/, /l/, and /v/. These phonemes are chosen as it is found that these consonant phonemes appear drastically different when they appear in the prevocalic and postvocalic positions of a word. For example, the phoneme /l/ in the words 'love' and 'apple' appear different because /l/ is at the start of the word 'love' and the end of the word 'apple' – these are allophonic variations of the phoneme /l/. A decision algorithm is implemented during the phoneme-to-keyframe mapping for these phonemes. Thus, in assigning the keyframe for the phoneme /l/ there must be a decision whether /l/ should be assigned keyframe 7, or keyframe 3. A simple decision algorithm that chooses between two keyframes has been implemented as follows: during the loading process, myVideoFrames are given a boolean at_start which is set to true if the phoneme appears near the start of a word, and false if it appears at the end of a word. Based on this value, the keyframe is decided during phoneme to keyframe mapping.

7.7 Evaluation

The rules for the complex blending algorithm deal with blending within phonemes, stealing frames, accentuating phonemes b, p and m, and deciding between keyframes for phonemes with allophonic variations. All of these rules make the second blending algorithm highly effective. Animation is exactly synchronized with speech, so enunciation appears accurate, and speech appears smooth and fluid. Furthermore, all of

the rules put together run in O(num_phonemes + num_words) for the loading process, and O(num_phonemes) for implementation of all the rules.

The usefulness of the rules may be evaluated in a two main ways: qualitatively, by evaluating visual appeal, and quantitatively, by determining if the rules improve the understanding of certain sentences or words. Results show (see Appendix B) that in comparison to two other lip-sync animation programs, the thesis's program was preferred visually and in understandability. Sentences were chosen to test the different components of the rules: blending within phonemes, allophonic variation, stealing-frames, and prosodromic variation. 97% percent of the time, the lip movements improved the understanding of speech. Additionally, in comparison to the other programs, this program was more synchronous with the speech and fluid (see Appendix B).

Chapter 8 – Applications

8.1 Introduction

The fast runtime for the implementation of animated speech rules is extremely useful for animation over IP. In fact, the application of these rules is the fastest part of the entire animation-over-IP algorithm. It takes much longer for the text-to-speech engine to convert text into a wave file than it takes for the lip-sync animation rules to work in O(num_phonemes + num_words). Classes to perform the animation rules are written in Java Swing.

8.2 Java Implementation of Animation over IP

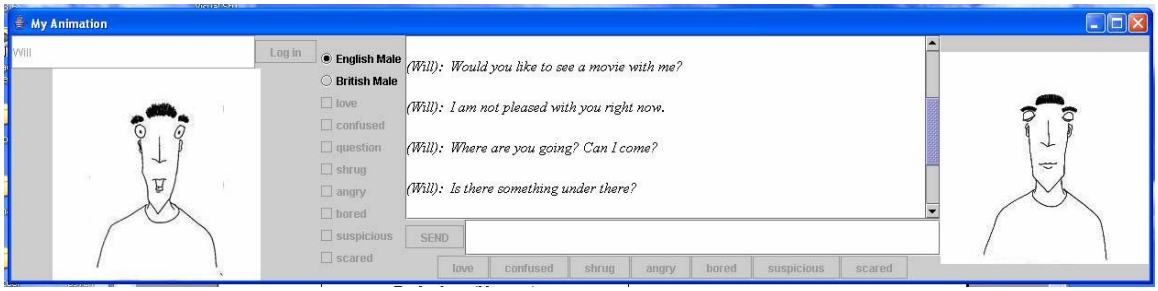


Figure 8.1. Avatar-over-IP

The users begin by logging into the chat program by running the JWAnimate applet. The user then chooses which voice the avatar should use. When typing into the chatbar, the user can include emotion tags to place emotions on words, at the end of sentences, or both (see Pechter 2004). After the tags are evaluated for emotion rules, the tags are extracted, and the raw text is given to the Festival program. The resultant wave, phoneme, and word duration files are then given to the class ParserLoader along with the voice type so that the phoneme alphabet is known. The ParserLoader class then parses the phoneme and word data and loads it into the Master Data Structure described in Chapter 5. Once the data structure is loaded, the animation rules are applied using class myRules. When the rules are finished, the frames are played at 30 fps along with the wave file. During the playing stage, emotions are displayed on top of the speech and at the end of sentences. Blinks after sentences also provide some prosodromic variation. When idle, the character blinks, shrugs, and looks from side to side.

8.3 Speech-to-Text Voice with Animation over IP

Lip-sync animation is also available with raw audio. Microphone-recorded speech is applied to a speech-to-phoneme API where word and phoneme are fit onto the wave file (see Pechter 2004). From here, the sound, word, and phoneme data files are manipulated similarly as described for the text-to-speech driven animation.

Applications for this speech driven animation are for the application of lip-sync to raw voice data on mobile forms. Here, the processor speed on a portable phone is significant. The algorithms described in this paper have the potential to work with a very small delay if the speech-to-phoneme part is done fast enough. The additional visual cues may even aid in the understandability of garbled speech caused by poor wireless phone connections.

8.4 Scalability

"Horsie the Horse" is an example of an avatar that was easily imported into the program (see Figure 8.2). Eight keyframes were chosen and strip-files were created by automatic generation of the in-betweens (Periaswamy and Farid 2003). Currently, this avatar may be selected in the Avatar over IP program. This application is clearly scalable, and many more avatars are expected to be added in the near future.

8.5 Future Work

Future work involves obtaining additional statistics on the effectiveness of the existing rules by means of surveys to see which ones have the most visual appeal, and which ones best improve the understandability of noisy speech. Additional rules may also be added to improve prosodromic variation. For example, some head swaying may be used at certain points in the sentence to improve visual appeal.

Furthermore, lip-sync rules can be geared to coincide with phonetics courses or the teaching of languages. Different languages and dialects may also be included in the near future, since voice synthesizers from MBROLA use the diphone blending method, which can easily be incorporated with the methods described in this paper.

Additionally, a toolkit for the general population can be created, whereby 8 keyframes (drawings, photographs, etc.) are selected by the user and input into the toolkit, which then automatically generates the files required for lip-sync animation. The avatars created by the toolkit can then be incorporated into Internet chat programs.

Finally, interactive games on mobile devices may be attempted. An interactive game need only store thousands of quotes on a database and feed them to the lip-sync animation program as necessary. In the future, users could even video conference by choosing an avatar and having the avatar lip-sync each person's raw voice.


Keyframe 1


Keyframe 2


Keyframe 3


Keyframe 4


Keyframe 5


Keyframe 6


Keyframe 7

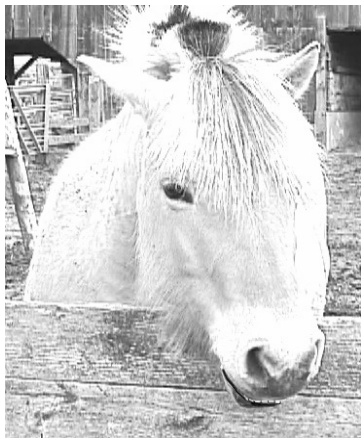

Keyframe 8

Figure 8.2. New Keyframes

Appendix A – Phoneme Alphabets

Table 1. Phoneme Alphabets (Venditti)

IPA Symbol	ARPAbet Symbol	Word	IPA Transcription	ARPAbet Transcription
[p]	[p]	parsley	['parsli]	[p aa r s l iy]
[t]	[t]	tarragon	['tærəgɑn]	[t ae r ax g aa n]
[k]	[k]	catnip	['kætnɨp]	[k ae t n ix p]
[b]	[b]	bay	[beɪ]	[b ey]
[d]	[d]	dill	[dɪl]	[d ih l]
[g]	[g]	garlic	['gɑrlɨk]	[g aa r l ix k]
[m]	[m]	mint	[mɪnt]	[m ih n t]
[n]	[n]	nutmeg	['nʌtmɛg]	[n ah t m eh g
[ŋ]	[ng]	ginseng	['dʒɪnsɨŋ]	[jh ih n s ix ng]
[f]	[f]	fennel	['fɛnl̩]	[f eh n el]
[v]	[v]	clove	[kloʊv]	[k l ow v]
[θ]	[th]	thistle	['θɪsl̩]	[th ih s el]
[ð]	[dh]	heather	['hɛðɚ]	[h eh dh axr]
[s]	[s]	sage	[seɪdʒ]	[s ey jh]
[z]	[z]	hazelnut	['heɪzl̩nʌt]	[h ey z el n ah t]
[ʃ]	[sh]	squash	[skwɑʃ]	[s k w a sh]
[ʒ]	[zh]	ambrosia	[æm'broʊʒə]	[ae m b r ow zh ax]
[tʃ]	[ch]	chicory	['tʃɪkɚi]	[ch ih k axr iy]
[dʒ]	[jh]	sage	[seɪdʒ]	[s ey jh]
[l]	[l]	licorice	['lɪkɚɨʃ]	[l ih k axr ix sh]
[w]	[w]	kiwi	['kiwi]	[k iy w iy]
[ɾ]	[r]	parsley	['pɑrsli]	[p aa r s l iy]
[j]	[y]	yew	[yu]	[y uw]
[h]	[h]	horseradish	['hɔrsrædɪʃ]	[h ao r s r ae d ih sh]
[ʔ]	[q]	uh-oh	[ʔʌʔoʊ]	[q ah q ow]
[ɾ]	[dx]	butter	['bʌɾɚ]	[b ah dx axr]
[ɾ̃]	[nx]	wintergreen	[wɪɾ̃ɚgrin]	[w ih nx axr g r i n]
[l̩]	[el]	thistle	['θɪsl̩]	[th ih s el]

IPA Symbol	ARPAbet Symbol	Word	IPA Transcription	ARPAbet Transcription
[i]	[iy]	lily	['lɪli]	[l ih l iy]
[ɪ]	[ih]	lily	['lɪli]	[l ih l iy]
[eɪ]	[ey]	daisy	['deɪzi]	[d ey z i]
[ɛ]	[eh]	poinsettia	[pɔɪn'sɛriə]	[p oy n s eh dx iy ax]
[æ]	[ae]	aster	['æstɚ]	[ae s t axr]
[ɑ]	[aa]	poppy	['papi]	[p aa p i]
[ɔ]	[ao]	orchid	['ɔrkɨd]	[ao r k ix d]
[ʊ]	[uh]	woodruff	['wʊdrʌf]	[w uh d r ah f]
[oʊ]	[ow]	lotus	['loʊrəs]	[l ow dx ax s]
[u]	[uw]	tulip	['tulɨp]	[t uw l ix p]
[ʌ]	[uh]	buttercup	['bʌrɚˌkʌp]	[b uh dx axr k uh p]
[ɝ]	[er]	bird	['bɝd]	[b er d]
[aɪ]	[ay]	iris	['aɪrɨs]	[ay r ix s]
[aʊ]	[aw]	sunflower	['sʌnflaʊɚ]	[s ah n f l aw axr]
[ɔɪ]	[oy]	poinsettia	[pɔɪn'sɛriə]	[p oy n s eh dx iy ax]
[ju]	[y uw]	feverfew	[fivɚfju]	[f iy v axr f y u]
[ə]	[ax]	woodruff	['wʊdrəf]	[w uh d r ax f]
[ɨ]	[ix]	tulip	['tulɨp]	[t uw l ix p]
[ɚ]	[axr]	heather	['hɛðɚ]	[h eh dh axr]
[ʉ]	[ux]	dude[1]	[dʉd]	[d ux d]

Table 2. Phoneme Alphabets II

	ARPABet	WorldBet	Example	Phonemic Transcription
1	AA	A	odd	AA D
2	AE	@	at	AE T
3	AH	^	hut	HH AH T
4	AO	>	ought	AO T
5	AW	aU	cow	K AW
6	AX		about	AX B AW T
7	AY	aI	hide	HH AY D
8	B	b	be	B IY
9	CH	tS	cheese	CH IY Z
10	D	d	dee	D IY
11	DH	D	thee	DH IY
12	EH	E	Ed	EH D
13	ER	3r	hurt	HH ER T
14	EY	ei	ate	EY T
15	F	f	fee	F IY
16	G	g	green	G R IY N
17	HH	h	he	HH IY
18	IH	I	it	IH T

19	IY	i:	eat	IY T
20	JH	dZ	gee	JH IY
21	K	kh	key	K IY
22	L	l	lee	L IY
23	M	m	me	M IY
24	N	n	knee	N IY
25	NG	N	ping	P IH NG
26	OW	oU	oat	OW T
27	OY	>i	toy	T OY
28	P	ph	pee	P IY
29	R	9r	read	R IY D
30	S	s	sea	S IY
31	SH	S	she	SH IY
32	T	th	tea	T IY
33	TH	T	theta	TH EY T AH
34	UH	U	hood	HH UH D
35	UW	u	two	T UW
36	V	v	vee	V IY
37	W	w	we	W IY
38	Y	j	yield	Y IY L D
39	Z	z	zee	Z IY
40	ZH	Z	seizure	S IY ZH ER
41	PAU	.pau	-	** This is a silent “pause” phoneme

Appendix B – Evaluation Surveys

Evaluation Survey for Avatar over IP
Three individuals were included in the survey.
"Yes" responses are recorded.

Sentence	**Do the lip movements improve the understanding of speech?**	**Do the lips move in synchronization with the speech?**	**Is the lip animation fluid?**
I like dogs and pigs.	3	3	3
The toy I know is not large.	3	3	3
I think the new crayon color is wicked cool.	3	3	3
He ate hay down by the bay, what do you say, okay?	2	3	3
Why I don't fly in the cave, I don't know.	3	3	3
Replay because the baby is sick.	3	3	3
The baby boy brought us a pig.	3	3	3
I just may do something, but I'll do it politely.	3	3	3
Baywatch is for people in Europe to gawk at hot girls.	2	3	3
Men of honor pay the price for the book.	3	3	3
Big maggots pose a threat to society.	3	3	3
He hit tom on the head with the broom	3	3	3
He wins something good.	3	3	3
Don't tell Lance, say yes or no.	3	3	3
Win nothing or report back to me.	3	3	3
Porch chairs stop moving.	3	3	3
I like the apple.	3	3	3
The dog dove and was mad.	3	3	3
The violin music is shows love is in the air.	3	3	3
Voices have distinction.	3	3	3
Don't become jaded, okay?	3	3	3
Would you like to sing with me today? I have time for you. It would be fun.	3	3	3
Are you available for an early lunch? I want some steak. Come with me!	3	3	3
Have a cheeseburger on me. You win. I lose.	3	3	3
The elephant at the zoo is huge! Do you think?	3	3	3
Totals:	97%	100%	100%

Evaluation Survey for VHost SitePal by Oddcast (“VHost SitePal”)
Three individuals were included in the survey.
“Yes” responses are recorded.

Sentence	**Do the lip movements improve the understanding of speech?**	**Do the lips move in synchronization with the speech?**	**Is the lip animation fluid?**
I like dogs and pigs.	0	0	0
The toy I know is not large.	0	0	0
I think the new crayon color is wicked cool.	0	0	0
He ate hay down by the bay, what do you say, okay?	0	0	0
Why I don’t fly in the cave, I don’t know.	0	0	0
Replay because the baby is sick.	0	0	0
The baby boy brought us a pig.	0	0	0
I just may do something, but I’ll do it politely.	0	0	0
Baywatch is for people in Europe to gawk at hot girls.	0	0	0
Men of honor pay the price for the book.	0	0	0
Big maggots pose a threat to society.	0	0	0
He hit tom on the head with the broom	0	0	0
He wins something good.	0	0	0
Don’t tell Lance, say yes or no.	0	0	0
Win nothing or report back to me.	0	0	0
Porch chairs stop moving.	0	0	0
I like the apple.	0	0	0
The dog dove and was mad.	0	0	0
The violin music is shows love is in the air.	0	0	0
Voices have distinction.	0	0	0
Don’t become jaded, okay?	0	0	0
Would you like to sing with me today? I have time for you. It would be fun.	0	0	0
Are you available for an early lunch? I want some steak. Come with me!	0	0	0
Have a cheeseburger on me. You win. I lose.	0	0	0
The elephant at the zoo is huge! Do you think?	0	0	0
Totals:	0%	0%	0%

Evaluation Survey for Famous3d ("Famous3d")
Three individuals were included in the survey.
"Yes" responses are recorded.

Sentence	**Do the lip movements improve the understanding of speech?**	**Do the lips move in synchronization with the speech?**	**Is the lip animation fluid?**
I like dogs and pigs.	3	3	3
The toy I know is not large.	2	3	3
I think the new crayon color is wicked cool.	1	2	2
He ate hay down by the bay, what do you say, okay?	2	2	2
Why I don't fly in the cave, I don't know.	2	2	2
Replay because the baby is sick.	2	2	2
The baby boy brought us a pig.	3	3	3
I just may do something, but I'll do it politely.	2	2	2
Baywatch is for people in Europe to gawk at hot girls.	2	3	3
Men of honor pay the price for the book.	2	2	2
Big maggots pose a threat to society.	3	3	3
He hit tom on the head with the broom	1	1	1
He wins something good.	1	1	1
Don't tell Lance, say yes or no.	2	2	2
Win nothing or report back to me.	3	3	3
Porch chairs stop moving.	2	2	2
I like the apple.	2	2	2
The dog dove and was mad.	2	2	2
The violin music is shows love is in the air.	3	3	3
Voices have distinction.	2	2	2
Don't become jaded, okay?	3	3	3
Would you like to sing with me today? I have time for you. It would be fun.	2	2	2
Are you available for an early lunch? I want some steak. Come with me!	2	2	2
Have a cheeseburger on me. You win. I lose.	1	1	1
The elephant at the zoo is huge! Do you think?	2	2	2
Totals:	69%	73%	73%

References

Brand, Matthew and Ken Shan. "Voice Driven Animation." Mitsubishih Electric Research Lab. Online Available: http:///cs.ucsb.edu/conferences/PUI/PUIWorkshop98/Papers/Brand.pdf. May 24, 2004.

Bronstein, J. Arthur. *American Dialect Society – Conference Papers on American English and the International Phonetic Alphabe*t. Tuscaloosa: University of Alabama Press, 1998.

Cohen, Michael and Dominic Massaro. "Modeling Coarticulation in Synthetic Visual Speech" Online Available: http://dipaola.org/stanford/facade/lipsync/geneva.pdf. May 24, 2004.

Comet, Michael. "Lip-Sync – Making Characters Speak" Online Available: http://www.comet-cartoons.com/toons/3ddocs/lipsync/lipsync.html. 1998.

"CSLU Speech Toolkit." Online Available: http://cslu.cse.ogi.edu/toolkit/. August 21, 2003.

E.J. Yannakoudakis and P.J. Hutton. *Speech Synthesis and Recognition Systems*. Chichester: Ellis Horwood Limited, 1987.

"Façade – Standford Facial Animation System." Online Available: http://dipaola.org/stanford/facade/presentation.html. March 19, 2001.

"Famous3d." Online Available: http://www.famous3d.com/web/index.html. May 12, 2004.

Hall, Currie Daniel. "Interactive Sagittal Section." Online Available: http://www.chass.utoronto.ca/~danhall/phonetics/sammy.html. May 23, 2004.

International Phonetic Association. "The International Phonetic Alphabet." Online Available: http://www2.arts.gla.ac.uk/IPA/fullchart.html. 1996.

Kakumanu, Osuna, Esposito, Bryll, Goshtasby, Garcia. "Speech Driven Facial Animation." *Department of CS and Engineering*, Wright State University (2001).

Lance, Williams. "Performance Driven Facial Animation." *Advanced Technology Group*. Apple Computer Inc. (1990).

Lewis, J.P. and F.I. Parke. "Automated Lip-Synch and Speech Synthesis for Character Animation." *Computer Graphics Laboratory*, New York Institute of Technology (1987).

"MBROLA." Online Available: http://tcts.fpms.ac.be/synthesis/mbrola.html. April 26, 2004.

Pechter, Joseph and William Pechter. "Hybrid Text-to-Speech 2000 Software." October 1999.

Pechter, Joseph. "Enhancing Expressiveness of Speech thorough Animated Avatars for Instant Messaging and Mobile Phones." Diss. Dartmouth College, 2004.

Periaswamy, Senthil and Hany Farid. "Elastic Registration with Partial Data." *Proceedings of the Second International Workshop on Biomedical Image Registration, Philadelphia, PA, 2003.* Hanover: Dartmouth College, 2003.

Periaswamy, Senthil and Hany Farid. "Elastic Registration of Intensity Variations." *IEEE Transactions on Medical Imaging* 22(7) (2003): 865-874.

Pfauntsch, Josef E. and Charles M. Shub. "Graphic Animation and Speech Synthesis Applications for the Auditory and Visually Disabled." University of Colorado (1988).

Pullum, Geoffrey and William Ladusaw. *Phonetic Symbol Guide*. Chicago: University of Chicago Press, 1986.

Singh, Sadanand and Kala Singh. *Phonetics – Principles and Practices*. Baltimore: University Park Press, 1976.

Venditti, Jennifer. "Phonetic Transcription, Context-dependent variation, and Intonation." Columbia Computer Science (September 123, 2002).

"VHost SitePal." Oddcast. Online Available: http://vhost.oddcast.com/vhost_minisite/products/sitepaltts.php. May 24, 2004.

Waters, Keith and Tom Levergood. "An Automatic Lip-Synchronization for Synthetic Faces." *Digital Equipment Corporation,* Cambridge Research Lab (1994).