MLSLib: A LIP SYNC LIBRARY FOR MULTI AGENTS AND LANGUAGES

H. Murakami*
H. Baba†
T. Noma*

*Department of Artificial Intelligence
Kyushu Institute of Technology
680–4 Kawazu, Iizuka, Fukuoka 820–8502, Japan
{h_mura, noma}@pluto.ai.kyutech.ac.jp
http://www.pluto.ai.kyutech.ac.jp/~noma/research/mlslib/

†Kyushu School of Engineering
Kinki University
11–6 Kayanomori, Iizuka, Fukuoka 820–8555, Japan
baba@fuk.kindai.ac.jp

ABSTRACT

This article presents MLSLib, a software library for human figure animation with lip syncing. The library enables us to easily use multiple TTS systems and multiple lip motion generators, and switch them arbitrarily. It also helps use of multiple speaking agents, possibly with different TTS systems and lip motion generators. The MLSLib is composed of three modules: LSSAgent, TTSManager, and FCP-Manager; The LSSAgent module provides unified simple APIs per single agent, independent of TTS systems and lip motion generators. The TTSManager and FCPManager manage TTS systems and lip motion generators, respectively. Both modules support standard sets of phonetic alphabets per language, and thus users are freed from TTS-dependent implementation of lip motion generators. Applications to multi-lingual agents and LOD in lip syncing are also presented.

Keywords: Lip Sync, human figure animation, TTS, LOD, MLSLib

1 INTRODUCTION

Lip motion greatly helps us to catch speech sounds, particularly in noisy environments. Similarly, in computer animation, virtual human's lip motion is helpful to viewers, and his/her lips should thus move synchronized with speech sounds. This we call *Lip Sync*.

To make agents speak arbitrary texts without prerecording, we usually use TTS (Text-To-Speech) systems[Dutoit97]. In TTS systems, a sequence of *phonemes* is first generated from an input text, and then sounds are synthesized from

the sequence. Lip syncing is thus realized by moving agents' lips according to the changes of phonemes. For example, in [Waters93], [Beskow95], and [LeGoff96], the phonemes are mapped to *facial control parameters*, possibly via *visemes*. A parametric facial model is then animated by the parameters with the effects of *coarticulation*[Cohen93]. Such systems are also commercially available now. Their extensions include image-based lip syncing[Ezzat98] and an approach to integrating facial images and synthesized speech organs[Ogata01].

These existing researches and systems, however, make an implicit assumption that the TTS, the facial model, and the facial control parameter generation algorithm[Cohen93] are all fixed, and do not refer to the use of multiple TTS systems, models, and algorithms. This prevents multi-lingual lip syncing, use of various facial models, and adaptive generation of control parameters, and simultaneously enforces detailed knowledges of individual TTS systems and facial control parameter generators (FCPGs) upon application programmers. In addition, since use of multiple speaking agents is not directly supported, programmers have often to elaborate appropriate program codes to let multiple characters speak.

To solve these problems, programmers should have a software environment that satisfies the following requirements:

(1) **Management and easy switching of multiple TTS systems and FCPGs**:
Directly managing different types of TTS systems and FCPGs and switching them can be a burden upon application programmers. If this requirement is satisfied, they can easily use multiple TTS systems and FCPGs in their applications.

(2) **Resolving differences in the capability of TTS systems and FCPGs**:
In general, different TTS systems have different capabilities, which have a great influence on software development. For example, some TTS systems can have multiple instances in a program, and others can only have a single instance. In the former, an individual instance can be assigned to each agent; while in the latter, the single instance should be shared by all agents. Resolving such differences will increase productivity in developing Lip Sync applications.

(3) **Independence of TTS systems and FCPGs**:
To use an arbitrary combination of TTS and FCPG, TTS systems and FCPGs should be mutually independent as much as possible. This property also helps programmers and, in addition, enables application systems to show wider variations with fewer TTS systems and/or FCPGs.

(4) **Support of Multiple Agents**:
For use of multiple agents, attributes of individual agents should be handled independently, and such data management should be supported by the software environment.

(5) **Unified and Simple APIs**:
Unified and simple APIs hiding detailed data management help programmers to develop Lip Sync applications easily.

This paper presents MLSLib (Multilingual Lip Sync Library), a TTS-based Lip Sync library satisfying the above requirements. The MLSLib enables application programmers to use multiple agents, TTS systems, FCPGs, and languages freely, and then greatly helps them to develop Lip Sync applications.

In the next section, the architecture of our MLSLib is discussed in detail. Its implementation issues and experimental results are discussed in Section 3. Finally Section 4 concludes this paper with some comments on future work.

2 SYSTEM ARCHITECTURE

2.1 System Overview

The MLSLib consists of three types of modules: TTSManager, FCPManager, and LSSAgent (Figure 1); The TTSManager manages all TTS systems used in a Lip Sync application and also resolves differences in their capability. The FCPManager (Facial Control Parameter Manager) manages all FCPGs used in the application. The LSSAgent (Lip Sync System Agent) manages the attributes for a single agent[1] and it also acts as an intermediary among user programs, the TTSManager, and the FCPManager. In the following subsections, these modules and their interactions are discussed in detail.

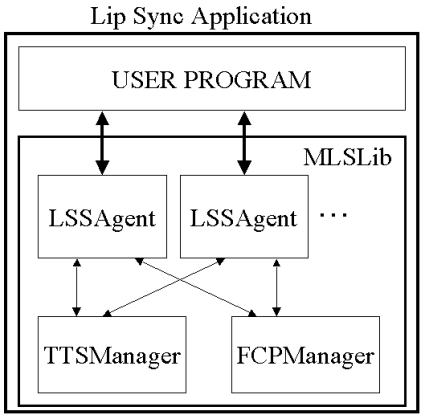


Figure 1: MLSLib Architecture

2.2 TTSManager

In general, TTS systems are different in many aspects. From application programmers' point of view, the following differences should be paid attention to:

(1) An application can have:
(a) multiple TTS instances, or
(b) only a single instance.

(2) A voice output function returns:
(a) immediately, or
(b) after the voice has been output.

(3) A TTS instance:
(a) occupies an audio device exclusively, or
(b) can share the audio device.

(4) Different TTS systems have different phoneme sets.

For example, Festival Speech Synthesis System[Black99] developed at the University of Edinburgh can create only a single instance and occupy an audio device exclusively, while L&H True Voice handled by MSWindows Speech API 4.0 [Microsoft98] can simultaneously have multiple TTS instances and then share the audio device.

To resolve these differences from programmers' viewpoint, the TTSManager is designed as follows: To settle the difference (1), the TTSManager has a three-layer structure as shown in Figure 2; Each node in the DLL layer manages a handler of an individual TTS, which is provided as Dynamic Link Library (DLL). Each node in the TTSE (TTS Entity) layer assigned to a TTS instance. Due to the difference (1), however, some TTS systems can only have a single instance even if multiple agents have to speak. To let multiple agents speak with such a TTS system, the TTSManager in our MLSLib has VTTS (Virtual TTS) layer, whose node is a *virtual* TTS instance corresponds to each agent. To set agent-dependent attributes such as pitch and speed, all programmers have to do is only to set the attributes through VTTS at the first time.

[1]LSSAgent modules are thus used per agent.

In case of a single-instance TTS, whenever the speaker changes, the TTSManager sets the attributes through TTSE on the background.

Because of the differences (2) and (3), our MLSLib does not use audio synthesizers provided by TTS systems. Instead, we developed a sharable synthesizer with DirectSound library on MS Windows. The synthesizer receives and plays multiple wave-form audio data generated by TTS systems.

To settle the difference (4), we first studied phoneme sets in various TTS systems. This revealed that common phoneme sets can be defined per language with simple mappings from TTS-dependent phonemes to common ones. We thus defined shared phoneme sets for Japanese and English, respectively. This enables programmers to use arbitrary combinations of TTS systems and FCPGs.

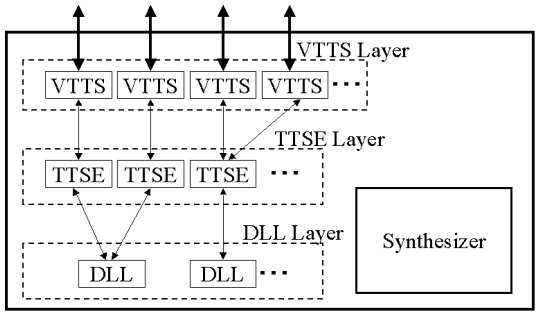


Figure 2: Three-Layer Structure of TTSManager

2.3 FCPManager

The FCPManager manages all FCPGs in a Lip Sync application. We make an assumption that FCPGs generate facial control parameters from a sequence of phonemes. For example, in an FCPG based on [Cohen93], control parameters are generated with coarticulation effects by considering dominance of the present and neighboring phonemes. Another FCPG simply generates parameters randomly without considering individual phonemes.

To make FCPGs independent of facial models, we recommend that FCPGs should represent (a vector of) facial control parameters as a weighted average of facial states, each of which corresponds to visemes. If appropriate sets of such faces are given in advance, such FCPGs can generate facial control parameters independent of facial models.

FCPGs in our MLSLib are in the form of DLL so that FCPGs can be added or deleted easily.

2.4 LSSAgent

In general, to develop Lip Sync applications, application programmers have to consider APIs of TTS systems and FCPGs, agent-dependent attributes such as pitch and speed, texts to be spoken and their corresponding phoneme sequences/audio data, and utterance timings including timer handling. The LSSAgent manages all of the above per agent, and then liberates programmers from their management.

2.5 Interactions among Modules

This subsection illustrates how the three modules and the user program make interactions with each other (Figure 3).

(I) **Generate a phoneme sequence and its audio data from a text**:

(1) Send a text to be spoken from a user program to the LSSAgent.

(2) Allocate memories to keep the phoneme sequence and its audio data.

(3) Send a text and the allocated memory pointers from the LSSAgent to the TTSManager.

(4) Generate and return a phoneme sequence and audio data and keep them in allocated memories.

(5) Return an ID of the text to be spoken.

(II) **Play the audio data**:

(1) Send the text ID from the user program to the LSSAgent.

(2) Transfer the audio data specified by the ID from the LSSAgent to the TTSManager.

(3) Play the audio data with a shared audio synthesizer, and send a start signal.

(4) Start a timer for measuring elapsed time of the agent's utterance, and keep the ID of the current text to be spoken.

(III) **Generate facial control parameters**:

(1) Send a memory pointer for returning the parameters.

(2) Measure the elapsed time of the agent's utterance.

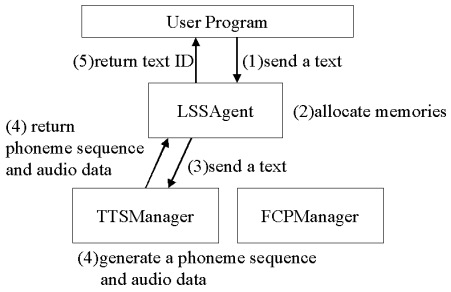
(3) Send the measured elapsed time and the current phoneme sequence from the LSSAgent to the FCPManager.

(4) Calculate and store the parameter values in the memory sent from the user program, which then receives the result.

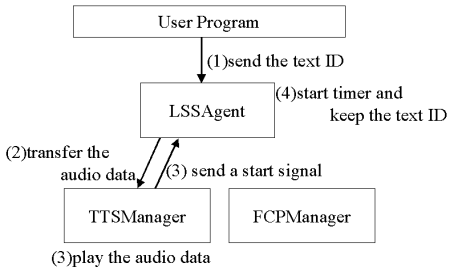
3 RESULTS

3.1 Implementation

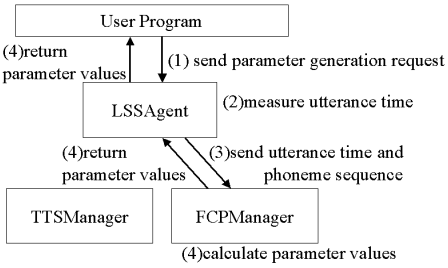
We developed the MLSLib with Visual C++ on MS Windows98. To use the MLSLib, user programs call public member functions in the LSSAgent class. The member functions are listed in Figure 4.



(I) Generate a phoneme sequence and its audio data from a text



(II) Play the audio data



(III) Generate facial control parameters

Figure 3: Interactions among Modules

3.2 Sample Program

A sample program in Figure 5 illustrates how to use the MLSLib.

(1) **Initializing agent/TTS/FCPG**
An agent (an instance of the LSSAgent) is created first ((a)). Next, TTS is created[2] and assigned to the agent with specified attributes ((b)). Furthermore, FCPG of specified attributes is assigned to the agent ((c)).

[2]The term "create" here includes initialization of TTSE and VTTS in the TTSManager.

CreateTTS	Create a TTS and assign it to an agent
GetState	Get a state whether an agent is speaking or not
SetPitch	Set an agent's voice pitch
SetSpeed	Set an agent's speech speed
GetTTSInfo	Get TTS capability information
TextToData	Change a text to a phoneme sequence and its audio data
DataToAudio	Play audio from audio data
CreateFCP	Create a FCPG and assign it to an agent
SetFCP	Set current FCPG to an agent
CalcParameter	Get current facial control parameters

Figure 4: Member Functions of LSSAgent Class (Extracts)

```
LSSAgent agent;                                              (a)
                ⋮

TTSINFO ttsi;
strcpy(ttsi.EngineName, "TTS ID");
strcpy(ttsi.VoiceName, "Voice Name");                        (b)
agent.CreateTTS(JAPANESE, ttsi);

                ⋮

FCPINFO fcpi;
strcpy(fcpi.EngineName, "FCPG ID");
strcpy(fcpi.InitFile, "file name");                          (c)
agent.SetFCP(agent.CreateFCP(fcpi));

                ⋮

int text_id;
text_id = agent.TextToData(JAPANESE, "text");                (d)
agent.DataToAudio(text_id);

                ⋮

double para[…];
while(speaking) {
    agent.CalcParameter(para);
    Draw agent('s face) with para;                           (e)
}
```

Figure 5: MLSLib Sample Program

(2) **Requesting agent to speak**
A user program obtains a text ID by sending a text to be spoken, and then requests the agent to speak it ((d)).

(3) **Drawing agent**
During speech, the user program obtains the current facial control parameters and draw the agent('s face) with the parameters, repeatedly ((e)). Note that the user program does not measure the elapsed time nor find the current phoneme.

3.3 Experiments

We made some experiments with our MLSLib as shown below. In these experiments, for English TTS systems, we used L&H TruVoice handled by MS Windows Speech API 4.0[Microsoft98] and Festival Speech Synthesis System[Black99] developed at the University of Edinburgh. For Japanese TTS, we also used L&H TruVoice. Sample movies of the experiments below can be found at our Web site shown on the cover.

3.3.1 Multiple Agents

We simulated a conversation between a father and his daughter. Voice attributes of both agents are independently managed by VTTSs in the TTSManager through the LSSAgent. In addition, our synthesizer enables their voices naturally overlap with each other.

3.3.2 Multi-lingual Agent

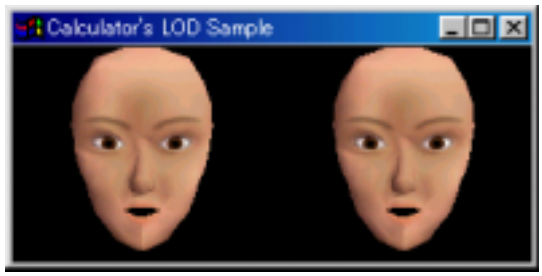
We sometimes want to animate a multilingual agent like an interpreter in computer animation. We simulated an agent speaking English and Japanese one after the other with our MLSLib, which enables us to use any TTS systems in a unified fashion and to switch one TTS to another very easily.

3.3.3 LOD for Lip Sync

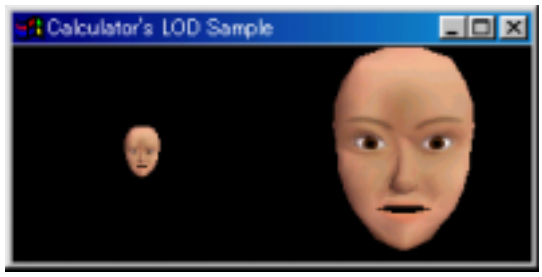
Recently, in computer graphics, LOD (Levels Of Detail) is considered important to balance the reality and rendering speed (e.g. [Funkhouser93] and [Hoppe96]). In typical LOD, more detailed models are used for neighboring objects to show their details; while simpler models are used for distant objects to reduce the rendering costs.

Similarly, in lip syncing, elaborate lip motions are required for neighboring agents to help viewers to understand what they are speaking. On the other hand, for distant agents, such costly lip motions are of little use since viewers cannot catch sounds from tiny lip motions on the screen. In such a case, all lip motions can convey is only whether the agents are speaking or not, and thus random lip motions are sufficient. We can also adopt an intermediate level where phonemes are roughly classified into, e.g., open lips or close ones.

Our MLSLib provides functions for switching FCPGs, and thus enables us to develop the above-mentioned LOD mechanism in Lip Sync applications. Figure 6 shows two images from a Lip Sync LOD sample program, where an agent's face on the left moves forward and backward from a viewpoint. (For comparison, the same face is drawn in the fixed size on the right.) In Figure 6(a), Cohen and Massaro algorithm[Cohen93] moves the lips of a near face; while in Figure 6(b), the lips move randomly for a far face. The FCPG is switched smoothly without interruption and delay.



(a) Face is near the viewpoint



(b) Face is far from the viewpoint

Figure 6: LOD in Lip Sync

4 CONCLUSIONS

This paper presented MLSLib, a software library for developing Lip Sync applications. It enables us to animate multiple agents speaking multiple languages and also use arbitrary combinations of TTS systems and FCPGs without detailed knowledges of individual TTS systems and FCPGs.

To show the effectiveness of our MLSLib, we developed some sample programs including animating multiple agents talking with each other, simulating multi-lingual agents, and realizing LOD in lip syncing. These programs are developed easily with simple APIs of the MLSLib.

For lip syncing, audio-visual systems, where natural facial animations are generated from audio track, have been developed by some authors including Bregler et al[Bregler97], Kuratate et al[Kuratate98], and Brand[Brand99]. In case of using TTS systems, however, their audio-based methods require wasteful translations via voice synthesis. Furthermore, their learning-based methods can only generate *natural* lip motions. Phoneme-based lip syncing can use any mapping from a phoneme sequence to lip motions,

and then easily animate faces, for example, in a *caricatured* fashion.

Our MLSLib freed application programmers from programming with system-dependent APIs of existing TTS systems. We, however, still have problems in semantics of attributes. For example, let us suppose that both TTS A and TTS B can have a pitch value from 0.0 through 1.0. But, in fact, pitch 0.5 in TTS A and in TTS B may have a different degree of highness/lowness of speaking voice. To solve such problems, we should extend the TTSManager to *standardize* these attributes as a future work.

ACKNOWLEDGEMENTS

This research is supported by Kayamori Foundation of Informational Science Advancement (K11 Res. IV No. 75), and Japan Society for the Promotion of Science (Grant-in-Aid for Scientific Research (C) 13680484).

REFERENCES

[Beskow95] Beskow, J., "Rule-Based Visual Speech Synthesis", *Proc. of EUROSPEECH '95*, Madrid, September 1995.

[Black99] Black, A. W., Taylor, P., and Caley, R.: *The Festival Speech Synthesis System System Documentation Edition 1.4, for Festival Version 1.4.0*, Centre for Speech Technology, University of Edinburgh, 1999.

[Brand99] Brand, M.: "Voice Puppetry", *Proc. of SIGGRAPH 99*, pp. 21–28, August 1999.

[Bregler97] Bregler, C., Covell, M. and Slaney, M.: "Video Rewrite: Visual Speech Synthesis from Video", *Proc. of AVSP '97*, Greece, September 1997.

[Cohen93] Cohen, M. M. and Massaro, D. W.: "Modeling Coarticulation in Synthetic Visual Speech," In: Thalmann, N. M. and Thalmann, D. (eds.), *Models and Techniques in Computer Animation*, Springer, pp. 139–156, 1993.

[Dutoit97] Dutoit, T.: *An Introduction to Text-to-Speech Synthesis*, Kluwer Academic Publishers, 1997.

[Ezzat98] Ezzat, T. and Poggio, T: "MikeTalk: A Talking Facial Display Based on Morphing Visemes", *Proc. of Computer Animation 98*, Philadelphia, PA, June 1998.

[Funkhouser93] Funkhouser, T. A. and Séquin, C. H.: "Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments," *Proc. of SIGGRAPH 93*, pp. 247–254, August 1993.

[Hoppe96] Hoppe, H.: "Progressive Meshes," *Proc. of SIGGRAPH 96*, pp. 99–108, August 1996.

[Kuratate98] Kuratate, T., Yehia, H. and Vatikiotis-Bateson E.: "Kinematics-based Synthesis of Realistic Talking Faces", *Proc. of AVSP '98*, pp. 185–190, Terrigal-Sydney, Australia, December 1998.

[LeGoff96] LeGoff, B. and Benoit, C.: "A Text-to-Audiovisual-Speech Synthesizer for French," *Proc. of Int'l Conf. on Spoken Language Processing*, Philadelphia, PA, October 1996.

[Microsoft98] *Microsoft Speech API 4.0 Documentation*, Microsoft, 1998.

[Ogata01] Ogata, S., Nakamura, S., and Morishima, S.: "Multi-modal Translation System —Modal Based Lip Synchronization with Automatically Translated Synthetic Voice—", *Proc. of Interaction 2001*, pp. 203–210, March 2001 (in Japanese).

[Waters93] Waters, K. and Levergood, T. M.: *DECface: An Automatic Lip-Synchronization Algorithm for Synthetic Faces*, Technical Report, DEC Cambridge Research Lab, September 1993.