CUTalk^{TM} – Cantonese Text-to-Speech System
Evaluation Version 2.0.0.3 (Oct 2001)

Introduction

CUTalk^{TM} is application software that can convert Chinese text into synthetic Cantonese speech. It includes a set of Application Programming Interface (APIs) that can be used in **MS Windows** or **Linux** based system. This APIs is developed in the C++ environment. Currently it supports Traditional Chinese (Big5 codes) and Simplified Chinese (GB) input. If the input strings are in English, the strings will be spelled ***(Only supported in Full Version)***. It also allows free switch between a male voice and a female voice, for both of which the rate of speaking can be altered by the programmer. The synthetic speech output is either played-back directly via the audio device (soundcard) or saved as an audio file. The most commonly used output file formats are supported.

CUTalk^{TM} Version 2.0.0.3 is a prototype that results from TTS research at Digital Signal Processing Lab., CUHK. Further development on this system is actively undergoing at the DSP Lab.

Package

There are 5 directories and one README.DOC file (this document) in the CD-ROM.

Directory Name	**Directory Contents**	**Description**
Data	**datab (Big5 Dictionary)** **datag (GB Dictionary)** **voice1.dat** **voice1.idx** **voice2.dat** **voice2.idx**	Audio data: "voice1.dat", "voice1.idx", "voice2.dat" and "voice2.idx" Linguist data: "datab" and "datag"
Example		An example of Visual C++ program showing the usage of the APIs
Inc	**cutalk.h**	Header file for the CTTS.
Lib	**cutalk.lib** **cutalk.dll**	Library files for CTTS.
Linux	**run** **speak.cpp**	Stuffs for Linux: Example programs:

	cutalk.h **cutalk.so.2.0.0.3**	"run" and "speak.cpp" Library files for CTTS "cutalk.h" and "cutalk.so.2.0.0.3"

CUTalk™ in MS Windows

(Windows 98, Windows 2000 and Windows NT)

`initCTTS()` ***initialize the CTTS***

Name:	int initCTTS(dataPath, gender)	
Input:	char *dataPath	• point to the path indicating the directory that storing the data files
	short gender	• indicate the gender of the voice
Returns:	1 if success 0 if failure	
Includes:	cutalk.h	
Type:	Initialization function	

♦ Description

The **initCTTS()** function is to initialize the TTS system by uploading the necessary audio and linguistic data. It must be called before the use of other functions.

Multi-threading is not supported in this function.

Parameter	Description
dataPath	Points to the string that indicates the full path (i.e. "e:\\progra~1\\CUTalk\\datafile") of the Data files.
gender	Indicates the gender of the voice data that will be initialized. Female: **F_VOICE** Male: **M_VOICE** Female & Male: **F_VOICE | M_VOICE** If the voice data of that gender is not initialized here, no sound of that gender can be played later. However, loading the voice of only one gender is a good way to save memory usage in the applications.

`class inputdata` ***store the parameters for the speech synthesis***

Name:	class inputdata	
Member:	short gender	• specifies the gender of the voice
	double sentence_rate	• indicates the speaking rate of the sound generated
	int font	• indicates the font type of the input characters
	int wav_type	• specifies the type of the sound data
	char *character	• points to the Chinese character string
	char wav_name[100]	• points to the string that storing the name (full path) of the output sound files
	short *output;	• points to the sound data after this function called
	HANDLE *hptr;	• points to the handle that controls the thread
	int num	• specifies the way of reading digit strings
	int punc	• specifies the way of handing punctuation
	int check	• indicates the process success or not
Includes:	cutalk.h	

♦ Description

The **inputdata** class stores all the necessary parameters for speech synthesis. These parameters are initialized in default stage after the class is declared and they can be changed using the function of **inputdata** class, which will be introduced in page 8-17.

Class Member	Description
gender	Indicates the gender of the voice data that will be spoken. Female: **F_VOICE** Male: **M_VOICE** If the voice data of that gender is not initialized by the function

	initCTTS() previously, that gender should not be assigned here. Default Value: **F_VOICE**
sentence_rate	Defines the speaking rate of the sound. This value should lie between **0.5** and **2**. Smaller value of it will make the sound spoken faster. Default Value: **1.2**
int font	Specifies the font of the input Chinese characters Big5 codes: **BIG5** GB codes: **GB** Default Value: **BIG5**
int wav_type	Indicate the format of the sound output. Microsoft wave files with 16k Hz sampling Frequency: **WAVE** Telephony speech with 8k Hz sampling Frequency: **MU** Raw (No Header) files with 16k Hz sampling Frequency: **RAW** Default Value: **WAVE**
char *character	Points to the Chinese Characters, which will be converted to transcriptions. Memory is allocated to this character point and the length of the Chinese Character string **must not** excess **2048 bytes**. Default Value: **"testing"**
char wav_name[100]	Points to the string that storing the name (full path) of the output sound files and the length of the string **must not** excess **99** bytes. Default Value: **NULL**
short *output;	If memory is allocated for this pointer, the sound data will be returned after calling of the function chinese2sound. The allocated memory must be sufficient for the output waveform. Otherwise, error will be resulted. Default Value: **NULL**
HANDLE *hptr;	If the address of the handle returned by the **CreateEvent** function is assigned to this pointer, the handle will be reset by the **SetEvent** function when the **chinese2sound** function or **chinese2wavfile** function is completed. Default Value: **NULL**
int num	If **1** is assigned to this variable, all the digit string (ASCII) in the

	input will be read as number. (e.g. "**1023**"will be read as "一千零二十三"). If **0** is assigned, the digit string will be treaded as digits only. (e.g. "**1023**" will be read as "一零二三"). Reading digit string as digit: **0** Reading digit string as number: **1** Default Value: **0**
int punc	If **1** is assigned to this variable, all the punctuation in the input string will be pronounced. If **0** is assigned, the punctuation will **not** be pronounced. Pronouncing the punctuation: **1** Not pronouncing the punctuation: **0** Default Value: **0**
int check	After completion of the function "**chinese2sound**" or "**chinese2wavfile**", this variable will return the status of the synthesis process (whether it is success or not). Success: **1** Fail: **0**

`inputdata::setgender()` ***Change the gender of the voice***

Name: short inputdata::setgender(gender);

Input: short gender

- specifies the gender of the voice

Returns: 1 if success
0 if failure

Includes: cutalk.h

Type: Setting Parameter function

♦ Description

The **setgender()** function is the function of class **inputdata**. It changes the parameter **gender** in the class **inputdata** and alternates the gender of the generated voice between male and female.

Parameter	Possible Values
gender	Female: **F_VOICE** Male: **M_VOICE**

`inputdata::setsentence_rate()` ***Change the speaking rate of the voice***

Name: short inputdata::setsentence_rate(sentenceRate);

Input: double sentenceRate

- indicates the speaking rate of the sound generated

Returns: 1 if success

0 if failure

Includes: cutalk.h

Type: Setting Parameter function

♦ Description

The **setsentence_rate()** function is the function of class **inputdata**. It changes the parameter **sentence_rate** in the class **inputdata** and adjusts the speaking rate of the generated voice.

Parameter	Possible Values
sentenceRate	Between **0.5** and **2**.

`inputdata::setwav_type()` ***Change the output format of the sound data***

Name: short inputdata::setwav_type(wav_type);

Input: int wav_type
- specifies the type of the sound data

Returns: 1 if success
0 if failure

Includes: cutalk.h

Type: Setting Parameter function

♦ Description

The **setwav_type()** function is the function of class **inputdata**. It changes the parameter **wav_type** in the class **inputdata** and hence the wave format of the generated voice.

Parameter	Possible Values
wav_type	Microsoft wave files with 16k Hz sampling Frequency: **WAVE** Telephony speech with 8k Hz sampling Frequency: **MU** Raw data (No header) with 16 kHz sampling Frequency: **RAW**

`inputdata::setfont()` ***Change the font of the input Chinese characters***

Name: short inputdata::setfont(font);

Input: int font

- indicates the font type of the input characters

Returns: 1 if success

0 if failure

Includes: cutalk.h

Type: Setting Parameter function

♦ Description

The **setfont()** function is the function of class **inputdata**. It changes the parameter **font** in the class **inputdata** and alternates the specified font of the input characters between Big5 and gb.

Parameter	Description
font	Big5 codes: **BIG5** Gb codes: **GB**

`inputdata::setnum()` ***Change the way of reading digit string***

Name: short inputdata::setfont(num);

Input: int num

- specifies the way of reading digit string

Returns: 1 if success

0 if failure

Includes: cutalk.h

Type: Setting Parameter function

♦ Description

The **setnum()** function is the function of class **inputdata**. It changes the parameter **num** in the class **inputdata** and alternates the way of reading digit string.

Parameter	Description
num	Reading digit string as number: **1**
	Reading digit string as digit: **0**

`inputdata::setpunc()` ***Change the way of handing punctuation***

Name: short inputdata::setpunc(punc);

Input: int punc

- specifies the way of handing punctuation

Returns: 1 if success

0 if failure

Includes: cutalk.h

Type: Setting Parameter function

♦ Description

The **setpunc()** function is the function of class **inputdata**. It changes the parameter **punc** in the class **inputdata** and alternates the way of handling the punctuation in the input string.

Parameter	Description
punc	Pronouncing the punctuation: **1** Not Pronouncing the punctuation: **0**

`inputdata::setcontent()` ***Change the content spoken by the voice***

Name: short inputdata::setcontent(character);

Input: char *character
- points to the Chinese character string

Returns: 1 if success
0 if failure

Includes: cutalk.h

Type: Setting Parameter function

♦ Description

The **setcontent()** function is the function of class **inputdata**. It changes the parameter **character** in the class **inputdata** and hence the content of the generated voice.

Parameter	Description
character	Any character strings with length no longer than **2048 bytes**. The character string must not excess **100** sentences and the sentences are separated by the Big5 punctuations.

`inputdata::setoutput()` ***Assign and Remove memory for the voice data***

Name: void inputdata::setoutput(output);

Input: short *output
- point to the variable which will store the output waveform

Includes: cutalk.h

Type: Setting Parameter function

♦ Description

The **setoutput()** function is the function of class **inputdata**. It assigns memory to the pointer **character** in the class **inputdata** or removes the memory from that. If the memory assigned is not sufficient, error may be resulted. An estimate of the needed memory is: number of characters in the class member **character** × 32000 + 1024 bytes.

Parameter	Possible Values
output	**Address** returned by calloc fucntion or malloc function. **NULL** indicates no memory is allocated.

`inputdata::setwav_name()` *Change the name of the output file*

Name: short inputdata::setwav_name(wav_name);

Input: char *wav_name

- points to the string that storing the name (full path) of the output sound file.

Returns: 1 if success

0 if failure

Includes: cutalk.h

Type: Setting Parameter function

♦ Description

The **setwav_name()** function is the function of class **inputdata**. It changes the parameter **wav_name** in the class **inputdata** and hence specifies the name of the output file.

Parameter	Possible Values
wav_name	Any string with length shorter than **99 characters**

`inputdata::sethandle()` *Assign and Remove handle the multi-threading*

Name: void inputdata::sethandle(hptr);

Input: HANDLE *hptr

- points to the handle for multi-threading

Includes: cutalk.h

Type: Setting Parameter function

♦ Description

The **sethandle()** function is the function of class **inputdata**. It assigns handle to the pointer **hptr** in the class **inputdata** or removes the handle from that.

Parameter	Description
hptr	**Address** returned by CreateEvent fucntion. **NULL** indicates no handle is assigned.

`chinese2sound()` ***Convert Chinese characters into Cantonese speech and playback the speech***

Name: void chinese2sound(ptr);

Input: void *ptr
- points to the class inputdata

Returns: Nil

Includes: cutalk.h

Type: Playing Sound function

♦ Description

The **chinese2sound()** function converts the Chinese characters stored in variable **character** in the class **inputdata** into sound. The sound will be played by the default audio player of the Windows operating system if the no memory has been assigned to the pointer **output** in the class inputdata. Otherwise, the voice data will be returned to **output** and no sound will be played. The parameters specifying the different properties of the speech is stored in the class **inputdata** and can be adjusted by the functions of class **inputdata**.

Multi-threading is supported in this function and this function can be enabled by assigning a handle to the hptr of class **inputdata. _beginthread** function can be used to start a new thread. In a particular moment, only **8** threads can be opened including the thread opened by **chinese2wavfile()** function. After the operation is finished, the assigned handle will be reset.

When **chinese2sound()** is completed, a value will be returned to **num** in the class **inputdata.** If **num** is **1**, **chinese2sound()** completed successfully. Otherwise, the synthesis process failed.

Parameter	Possible Values
ptr	Address of class inputdata.

`chinese2wavfile()` ***Convert Chinese characters into Cantonese speech and stores the voice data in a file***

Name: void chinese2wavfile(ptr);

Input: void *ptr
- points to the class inputdata

Returns: Nil

Includes: cutalk.h

Type: Generation function

♦ Description

The **chinese2wavfile()** function converts the Chinese characters stored in variable **character** in the class **inputdata** into sound and stores the output in the file specified in wav_name in the class **inputdata**. If no file name is specified by the wav_name, a randomly generated file name will be employed. The parameters specifying the different properties of the speech is stored in the class **inputdata** and can be adjusted by the functions of class **inputdata**.

Multi-threading is supported in this function and this function can be enabled by assigning a handle to the hptr of class **inputdata. _beginthread** function can be used to start a new thread. In a particular moment, only **8** threads can be opened including the thread opened by **chinese2sound()** function. After the operation is finished, the assigned handle will be reset.

When **chinese2wavfile()** is completed, a value will be returned to **num** in the class **inputdata.** If **num** is **1**, **chinese2wavfile()** completed successfully. Otherwise, the synthesis process failed.

Parameter	Possible Values
ptr	Address of class inputdata.

`chinese2xscript()` ***Converts Chinese characters into Cantonese syllabic transcription***

Name:	int chinese2xscript(xscript, ptr);	
Input:	char **xscript	• stores the outputted transcription
	class inputdata *ptr	• points to the class inputdata
Returns:	the number of the transcription	
Includes:	cutalk.h	
Type:	Conversion function	

♦ Description

The **chinese2xscript()** function converts the input Chinese characters stored in variable **character** in the class **inputdata** into transcriptions and stores them in the **xscript**. The transcription can be passed to **xscript2sound()** function or **xscript2wavfile()** function. The parameters specifying the different properties of the voice is stored in the class **inputdata** and can be adjusted by the functions of class **inputdata**.

Multi-threading is **not** supported in this function.

Parameter	Possible Values
xscript	**Address** returned by calloc fucntion or malloc function.
ptr	**Address** of class inputdata.

`xscript2sound()` ***Convert Cantonese transcription into speech and playback the speech***

Name:	int xscript2sound(txtScript, numScript, ptr);	
Input:	char **txtScript	• points to the transcription
	int numScript	• specifies the number of transcription
	class inputdata *ptr	• points to the class inputdata
Returns:	1 if success	
	0 if failure	
Includes:	cutalk.h	
Type:	Generation function	

♦ Description

The **xscript2sound()** function converts the transcription stored in txtScript into sound. The sound will be played by the default audio player of the Windows operating system if the no memory has been assigned to the pointer **output** in the class inputdata. Otherwise, the voice data will be returned to **output** and no sound will be played. The parameters specifying the different properties of the speech is stored in the class **inputdata** and can be adjusted by the functions of class **inputdata**.

Multi-threading is **not** supported in this function.

Parameter	Possible Values
txtScript	**Addresses** of the pointers point to the transcription. It can be obtained by calling **chinese2xscript** function.
numScript	The number of transcription. If the transcriptions are from the **chinese2xscript()** function, numScript is equal to the returned value of **chinese2xscript()** function.
ptr	**Address** of class inputdata.

`xscript2wavfile()` ***Convert the transcription into speech and stores the speech in a file***

Name: int xscript2wavfile(txtScript, numScript, ptr);

Input:

char **txtScript	• points to the transcription
int numScript	• specifies the number of transcription
class inputdata *ptr	• points to the class inputdata

Returns: 1 if success
0 if failure

Includes: cutalk.h

Type: Generation function

♦ Description

The **xscript2wavfile()** function converts the transcription stored in txtScript into sound and stored the output in the file specified in wav_name in the class **inputdata**. If no file name is specified by the wav_name, a randomly generated file name will be employed. The parameters specifying the different properties of the speech is stored in the class **inputdata** and can be adjusted by the functions of class **inputdata**.

Multi-threading is **not** supported in this function.

Parameter	Description
txtScript	**Addresses** of the pointers point to the transcription. It can be obtained by calling **chinese2xscript** function.
numScript	The number of transcription. If the transcriptions are from the **chinese2xscript()** function, numScript is equal to the returned value of **chinese2xscript()** function.
ptr	**Address** of class inputdata.

`termCTTS()` ***terminate the CTTS***

Name: void termCTTS()

Input: None

Returns: None

Includes: cutalk.h

Type: Termination function

♦ Description

The **termCTTS()** function is used to terminate the TTS operation. It essentially removes the audio and linguistic data that have been uploaded by **initCTTS()**.

Multi-threading is not supported in this function.

CUTalk™ in Linux

Audio data and Linguist data:

Use the data files stored in the directory "Data".

Function calls:

All the function calls discussed in the MS Windows version can be applied to the Linux version. Except that

i) Multi-thread function of Chinese2sound() and Chinese2wavfile() **is** started by pthread_create() but **not** _beginthread().

ii) The class member "**HANDLE *hptr**" of class **inputdata** is omitted. Since this is not necessary for multi-threading in Linux.