TTSApp ReadMe

Benjamin ‘Benilda’ Key: **Ben.Key@YekNeb.com**

Contents

Introduction

TTSApp is a simple open source Text to Speech application for reading text files and the clipboard using SAPI5 or the Microsoft Speech Platform. It is a dialog box application for Microsoft Windows that is written using the **Microsoft Foundation Class (MFC) Library**. It runs on Microsoft Windows. I have only tested it on Microsoft Windows 10, but it should work on Windows 7 and up.

The application can be built to used either **Microsoft Speech API 5.3** or the **Microsoft Speech Platform** for Text to Speech services.

Table of Contents

Features

- Prosody manipulation: voice, rate, and volume.
- Speak text.
- Pause and Stop speech.
- Speak a text file.
- Save to MP3 or WAV file.
- Speak clipboard text (*Control+Shift+Windows+C*)
- Highlights each word as it is spoken.
- Customize font.
- Supports command line arguments for setting the voice and speaking a text file to an MP3 or WAV file.
- Support for the following file encodings: unspecified, UTF-8, UTF-16le, UTF-16be, UTF-32le, and UTF-32be. File encodings are detected using the **Byte Order Mark**.

Table of Contents

Building

To build the application use **Microsoft Visual Studio 2019**. To build the **Microsoft Speech Platform** configurations you need to install the **Microsoft Speech Platform - Software Development Kit (SDK) (Version 11)**. On 64-bit systems you should install both the x64 and x86 versions.

In addition, TTSApp depends upon the **Boost C++ Libraries** and the **LAME MP3 Encoder**.

The Boost C++ Libraries and LAME are available in **VCPKG**. **My fork of VCPKG** includes **InstallPackagesForWindows.py**, a Python script I use to automate the installation of the packages I use including the **Boost C++ Libraries** and the **LAME MP3 Encoder**.

To build TTSApp, do the following.

1. Install **Microsoft Visual Studio 2019**.
2. Install the **Microsoft Speech Platform - Software Development Kit (SDK) (Version 11)**.
3. Install the **Microsoft Speech Platform - Runtime (Version 11)** and at least one of the **Microsoft Speech Platform - Runtime Languages (Version 11)**.
4. Install **Python**.
5. Install **Git for Windows**.
6. Clone **My fork of VCPKG**.
7. Build VCPKG.

8. Run the ‘vcpkg integrate install’ command.
9. Run InstallPackagesForWindows.py to install the **Boost C++ Libraries** and the **LAME MP3 Encoder**.

Table of Contents

Motivation

I have low vision. However, my vision is not yet so poor that I need a fully featured screen reader such as **JAWS**, **NVDA**, or **Narrator**. I primarily need a text to speech application to read articles on the Internet. I need an application that allows me to select the text on a web site, copy it to the clipboard, and have it be spoken with the press of a keystroke. Thus far I have not been able to find an affordable application that has the features I need without adding bloat that I do not need. I eventually decided to write such an application myself.

Table of Contents

Command Line Options

```
{TTSApp} usage:

{TTSApp} [options...]:
  -h [ --help ]          Display this information.
  -?                     Display this information.
  -v [ --version ]       Display version information.
  -i [ --input ] arg     Text file to read and convert
to an MP3 or WAV file.
  -c [ --csv ] arg       CSV file containing information
on strings to read and
                         convert to MP3 or WAV files.
Each line of the CSV file
                         is assumed to consist of two
columns of data. The first
                         column is assumed to contain
the file name of the WAV
                         or MP3 file in which to save
the spoken text. The
                         second column is assumed to contain
the text that is to
                         be spoken.
  -o [ --output ] arg    When used with the input option,
this option specifies
                         the MP3 or WAV file to output
the spoken text to. If
```

```
                        the output file is not specified,
this parameter
                        defaults to the input file with
the extension renamed
                        to .wav.
  -r [ --read ] arg     Text file to read. Unlike the
input option, the read
                        option causes the text to be
spoken aloud.
  -V [ --voice ] arg    Voice name to use for speech.
  -w [ --wait ]         When used with the -i and -o
options, specifies that
                        TTSApp will wait until after
the MP3 or WAV file is
                        written before continuing.
  --rate arg            Voice rate. The valid range is
-10 to 10.
  --volume arg          Voice volume. The valid range
is 0 to 100.
```

Table of Contents

Usage

TTSApp is a MFC dialog box application. It contains a menu bar, a large Rich Edit window, a Prosody group box, a Speech group box, and a File group box.

The Prosody group box contains the Voice combo box and the Rate and Volume slider bars. The Voice combo box makes it possible to set the active voice. The Rate slider bar allows you to set the speech rate and the Volume slider bar allow you to set the speech volume.

The Speech group box contains the Speak button, the Pause button, the Stop button, and the Process XML check box. The Pause button and Stop button are initially disabled. They are enabled only while the application is speaking. The Speak button causes the application to begin speaking the text contained in the Rich Edit window using the selected prosody settings. Once the application begins speaking, the Pause button and Stop button are enabled. If you wish to pause or stop speech, you may do so by pressing these buttons. The Process XML check box controls whether or not **SSML** or **SAPI XML** tags will be processed.

The File group box contains the Open File button and the Save to sound file button. The Open File button causes an Open dialog box to be displayed to prompt a user for a .txt, .ssml, or a .xml file to open. Once the file is selected, the file will be opened and its contents will be displayed in the Rich Edit

window. The Save to sound file button causes the spoken text to be saved to an MP3 file or a WAV file instead of being spoken aloud. If the user first opened a file using the Open File button, the sound file name will be derived from the name of the opened file. Otherwise the user will be prompted to choose a file name using a Save dialog box.

The menu bar contains the following menus: File, Format, Edit, and Help.

The File menu contains the Open, Save to sound file, and Exit menu items. The Open menu item serves the same purpose as the Open File button. It is assigned to the accelerator key Ctrl+O. The Save to sound file menu item serves the same purpose as the Save to sound file button. It is assigned to the accelerator key Ctrl+S. The Exit menu item closes the application. It is assigned to the accelerator key Alt+F4.

The Format menu contains the Font, Font (Fixed Width), Decrease Font Size, and Increase Font Size menu items. The Font menu item causes a Font dialog box to be displayed, making it possible for the font used in the Rich Edit window to be customized. The Font (Fixed Width) also causes a Font dialog box to be display; however, it causes the Font dialog box to display only fixed width fonts. The Decrease Font Size menu item decrements the font size by one point. It is assigned to the accelerator key Ctrl+-. The Increase Font Size menu item increments the font size by one point. It is assigned to the accelerator key Ctrl++.

The Edit menu contains the following menu items: Cut, Copy, Paste, Delete, Clear, and Select All. The Cut menu item cuts

selected text in the Rich Edit window to the clipboard. It is assigned to the accelerator key Ctrl+X. The Copy menu item copies selected text in the Rich Edit window to the clipboard. It is assigned to the accelerator key Ctrl+C. The Paste menu item pastes text from the clipboard to the Rich Edit window. It is assigned to the accelerator key Ctrl+V. The Delete menu item either deletes selected text from the Rich Edit window or deletes the character after the caret from the Rich Edit window. It is assigned to the accelerator key Del. The Clear menu item clears all text from the Rich Edit window. It is assigned to the accelerator key Ctrl+Del. The Select All menu item selects all text in the Rich Edit window. It is assigned to the accelerator key Ctrl+A.

The Help menu contains the About TTSApp and Usage menu items. The About TTSApp menu item causes the about box to be displayed. It is assigned to the accelerator key Alt+F1. The Usage menu item causes a dialog box displaying the command line help to be displayed.

The **Command Line Options** section above gives a complete description of available command line options.

The command line options that are used for reading text files require further explanation.

The -i or –input options are used to allow the Save to sound file functionality to be controlled via the command line. It could be used in conjunction with the -o or –output to specify the file name of the MP3 or WAV file that is to be used or it could be used without the -o or –output command line options. When it

is used without the -o or –output command line options, a WAV file is created and the WAV file name is automatically derived from the text file name.

The -r or –read options are used to cause the text file to be spoken aloud instead of having the spoken text saved to an MP3 or WAV file.

There are multiple use case scenarios for TTSApp. The simplest is to type or paste text in the Rich Edit window, set the desired voice prosody options, and press the Speak button. The Pause and Stop buttons, which are initially disabled, are enabled while TTSApp is speaking. If you wish to Pause or Stop speech, you may do so using these two buttons.

TTSApp also functions as a clipboard reader. Simply copy text you wish to be spoken to the clipboard and press *Control+Shift+Windows+C*. The clipboard text will be placed in the Rich Edit window and spoken. I often use this functionality to read web pages. This was a primary motivation for writing this application.

TTSApp can be build with four variations, 32-bit SAPI support, 32-bit Microsoft Speech Platform support, 64-bit SAPI support, and 64-bit Microsoft Speech Platform support. The SAPI support variations are named TTSApp-SAPI.exe. The Microsoft Speech Platform support variations are named TTSApp-MicrosoftSpeechPlatform.exe.

TTSApp functions as a single instance application. Only one instance of each variation can run at a time. If a second instance

is started, any command line options are forwarded to the first instance and it is brought into the foreground.

The -w command line option is used when forwarding commands to another instance. Instead of simply shutting down when sending the requests to the first instance, it waits. This is specifically used in conjunction with the Save to sound file functionality to make scripts that save multiple text files to MP3 or WAV files one after another possible.

Table of Contents

Credits

Support for generating MP3 files was accomplished by using three libraries by **trodevel** and the **LAME MP3 Encoder**. Specifically, the following three libraries by trodevel are used.

- **convimp3**: Convenience MP3 library. Enables easy PCM to MP3 conversion and vice versa.
- **lameplus**: Very thin C++ wrapper for LAME library.
- **wave**: WAV file appender.

These libraries are released under the **GNU General Public License Version 3**.

I discovered these libraries thanks to the Stack Overflow article **Is there any LAME c++ wrapper/simplifier**.

Table of Contents

License

This software is licensed under **The BSD 2-Clause License**.

Table of Contents