How to build your own voice file

About

This is an in-depth description how to build your own voice files. It requires a little more than basic computer knowledge.

General steps

First, you need the [language].lang file for your language. Bring this up to date if necessary, and fill in the voice: values for those entries that have them in the current english.lang file (the master of all .lang files). The voice files are then created by performing the following 4 steps:

1. Convert each voice: value into a .wav file. This can be done either by using Text-to-speech software, or recording a human voice reading the items. Of course, only the first alternative can be automated. The VOICE_PAUSE entry is a special case, see detailed description below.
2. (Optional, but recommended.) Trim the .wav clips so that there is (almost) no leading and trailing silence. This helps both to keep the voice file small and to have fluent spelling and number speaking.
3. Compress all .wav snippets into .mp3 using lame or another encoder.
4. Assemble the final .voice file from all these .mp3 snippets. A special tool (provided here) is needed for this. If the voice file gets larger than 1.5 MB, go back to step #3 and choose a higher compression.

Using Windows

This is the almost fully automated way I use for building the voice files, using AT&T Natural Voice and Microsoft SAPI5 voices. You will need at least some knowledge about VBScript programming to make this work. Some knowledge about regular expressions is also useful.

Prerequisites

- At least one SAPI5 (much preferred) or SAPI4 (use only if absolutely necessary) compatible voice installed.
- The wavtrim tool, enhanced version.
- The _voice_pause.wav file.
- lame.exe in the search path
- The voicefont tool.
- The MakeVoices VBScript

Prepare your environment

1. Make a directory for building voice files.
2. Put the following files into this directory: Wavtrim.exe, Voicefont.exe, _voice_pause.wav and MakeVoices.vbs
3. Open a command prompt, change into your build directory and try to start lame from there. If it does not work, lame is not in your path. You then have 2 options: (1) Put lame into a directory which is in the path (I use C:\Windows). (2) Put lame.exe into your build directory.
4. Put the current .lang files for the language(s) you intend to build voice file(s) for into the build directory as well. Also add the current english.lang. English.lang is the master of all language files, needed to determine the correct language id number order.

Adapt MakeVoices.vbs to your needs

The languages and voice names that the script should build voice files from are currently hard coded in the script, mostly because there are some more parameters which need manual adjustment anyway.

Open the file in a text editor and go to the section that starts at line 60. It is titled "The actual work". There are already several subroutine calls present in that section, left there to be used as examples. A line starting with a single quote (apostrophe) is a comment, i.e. you can use this to temporarily disable a subroutine call without removing it.

The subroutine to use for building your intended voice file depends on whether your voice is SAPI5 or SAPI4 compatible. For SAPI5 voices it is `MakeVoiceFile`, for SAPI4 voices it is `MakeVoiceFileSAPI4`. These subroutines take the following parameters:

```
MakeVoiceFile <language>, <voice name>, <sample format>, <noise floor>, <lame parameters>
MakeVoiceFileSAPI4 <language>, <voice name>, <noise floor>, <lame parameters>
```

Parameter	Description
`language`	The language that the voice file is built for. This needs to be the same as the name of the .lang file, without the .lang extension. Of course, this should also match the language of the voice
`voice name`	The name of the SAPI voice to use. Note that you need the name as known in Windows, which is not always obvious. If you don't know this, you can use the provided ListVoices.vbs VBScript. Run this from a command shell, with cscript.
`sample format`	(SAPI5 only) Defines the sample format that the TTS engine uses (mono/ stereo, sample rate, sample width). Use the predefined constants to select this. 16 bit mono settings are strongly recommended. The optimal sample rate depends on the TTS engine used. Best results are achieved if the sample rate matches the internal sample rate of the engine. For the Microsoft voices, this is SPSF_22kHz16BitMono, and for AT&T Natural voices it is SPSF_32kHz16BitMono. If you use another TTS engine, you'll need to experiment.
`noise floor`	The maximum sample value that will be treated as silence by the enhanced wavtrim tool. Not all TTS engines use digital silence, and the noise floor may even vary across different voices using the same engine, so this parameter needs adjusting. Convert an arbitrary text snippet into a .wav clip, load it into a sample editor, and check the beginning and/or end for the noise floor. The value is expressed as an absolute 16 bit integer sample value (0 dB equals 32767).
`lame parameters`	The parameter string handed over to lame for compressing the .wav clips. See next table.

Lame parameters example:

```
-V 4 -B 64 --resample 12 --scale 0.6 --vbr-new -t -S
```

Parameter	Description
`-S`	Be silent, i.e. don't write to the console. As the output of the lame command is neither seen nor stored, **you'll always want this.**
`-t`	Do not write lame tag. The lame tag would only waste space, so you should **always set this.**
`--scale <factor>`	Scale the input volume by <factor>. The factor 0.6 is recommended. Higher values make the voice UI louder, lower values make it more quiet. Don't use values >1, or the voice will probably get distorted.
`--vbr-new`	Use VBR compression, new algorithm. Using VBR delivers good quality without wasting space, so you

	should **always use it.**
`-V <quality>`	Select the VBR quality. <quality> can assume a value between 1 (best quality, largest file) and 9 (worst quality, smallest file). The recommended value for speech is 4; you can go up to 6 if the voice file gets too large. Higher values lead to noticeable compression artefacts. If you would need a setting above 6 to get the voice file small enough, lower the resample frequency instead.
`--resample <freq>`	Resample the .wav input to <freq> kHz. Allowed values are 48, 44, 32, 24, 22, 16, 12, 11 and 8. **Always set this**, to ensure all .mp3 clips use the same frequency. You will probably have to use relatively low values, because otherwise the voice file will become too large. If you need to go below 16, you'll need to set the -B parameter as well, because of some limitations in the MAS MP3 decoder chip. Do not use 11, for the sake of player owners (also due to the MAS limitations)!
`-B 64`	Do not produce frames with a bitrate >64 kbps. Set this when using resample frequency settings of 12 or 8. See above why.

Build your voice file(s)

Now that you prepared everything, you are ready to try building your first voice file. Open a command prompt, change to your build directory, and start the build process with
`cscript MakeVoices.vbs`
Be sure to use cscript.exe for running the script, not wscript.exe. The script contains some WScript.Echo statements, which would block as a message box if run with wscript.exe. That is the reason why you can't simply run this by double clicking, or you would need to register cscript.exe as your default VBScript interpreter. There is a Microsoft knowledge base article describing how to do this.

Be prepared that the build process can take rather long if you have to use a SAPI4 voice. SAPI4 voices are handled in realtime, so speaking the full vocabulary may take 15 minutes and longer. Sometimes SAPI4 voices do also produce dropouts, that's why SAPI4 is not recommended.

If all went well, you will now have one or more .voice files in your build directory. Check the size(s). If the file(s) is/are larger than 1.5 MB, you'll need to adjust the lame parameters to produce (a) smaller file(s). Voice files that are larger than 1.5 MB may not load depending on how much buffer space is available on the box. Use a higher -V setting and/or reduze the sample frequency to achieve this.

If your voice file is small enough, you should now try it on your box.

(Optional) Adjust the pause

The pause clip is special, in that it does neither get generated by the TTS engine, nor get trimmed for obvious reasons. The provided pause file is 300 msec long, but you can change it. The sampling frequency does not matter, because lame will resample it if needed (that's why I recommend to always set the --resample parameter), and resampling will never cause aliasing effects because the pause clip only contains digital silence.

Adjusting pronunciation

If you test your voice file, you will most probably notice that some words are pronounced wrong. There is a mechanism provided in the script to adjust these words. You may have noticed (if not building one of the voices already known to the script), that it emits a warning that the voice is unknown. This is because you did not yet define a replacement table for such adjustments.

In order to create and use such a list, you need to edit two places within the script. First, add a function that creates such a list for your voice. The replacement list definition functions are located towards the end of the script. In order to start with an empty list, simply duplicate the very last function `Gen_default` and rename it. My observation is that all voices for one language of a specific engine tend to need the same adjustments, e.g. all German AT&T voices can share the same function, so the suggested function name is `ATT_de_de` in this case. If you rename the function, you also need to rename the return variable name within the function to read the same.

Now you need to connect this function to the voice. Add a line to the `InitReplaceLists` subroutine. Refer to the lines that are already there to see how to do this.

If you try to build again, you should no longer get the warning about the voice being unknown. You can now add entries to the replacement list. The replacement list uses regular expressions, refer to the other lists how to do this. Adjust and rebuild until everything sounds as intended.

If you now have a nice .voice file, you may want to share it with other users by uploading it to the VoiceFiles topic, especially if you created a voice file for a language that is not yet supported. If possible, try to maintain your voice file, i.e. update it from time to time to reflect changes in the .lang file. If you had to update the .lang file first, consider sharing this as well (filing a patch if you don't have commit access). The .voice file may be of limited use without a current .lng file whch needs to be generated from your current .lang file.

If you added a new replacement list to the script, it would be nice to share this too. Simply upload the script to this wiki topic, replacing the previous version. The wiki uses attachment versioning, so previous versions are always available.

Subsequent builds (when the .lang file has changed) are quite easy. Simply put the updated .lang file(s) into the build dir, and start the script again. From time to time you may need to adapt the lame parameters (if the .voice file becomes too large) or the replacement list (if a new word is pronounced wrong).

Using Linux

Yet to be written by a Linux expert ;-)

Tools

Tool	Description	UPDATED
wavtrim.c	Enhanced wavtrim tool, originally written by Jörg Hohensohn. The enhanced version does not only trim digital silence, but scans the file from either end until it finds a sample with a higher value than the given noise floor value. Then it trims this end, leaving 10 msec of the silent part there in order to not cut away faint beginnings/ends.	
wavtrim.exe	Enhanced wavtrim tool, Windows executable.	
voicefont.cpp	Voicefont tool, written by Jörg Hohensohn.	2005-01-31
voicefont.exe	Voicefont tool, Windows executable.	2005-01-31
_voice_pause.wav	Pause .wav file (300 msec).	
MakeVoices.vbs	VBScript for automated .voice building.	2005-02-01
ListVoices.vbs	VBScript that lists all installed SAPI5 voices.	
ListVoices_SAPI4.vbs	VBScript that lists all installed SAPI4 voices.	