A comparison of TTS engines for integration in Rockbox

Delyan Kratunov
15. Apr. 2010

Table of Contents

I.Procedure....2
A. Hardware....2
B. Versions used....2
C. Speed tests....2
D. Memory tests....2
II.General info....2
A. Application binary sizes....2
B. Conclusions....3
III.Speed tests....3
A. Results....3
B. Conclusions....3
IV.Memory tests....3
A. Results....3
B. Conclusions....3
V.Appendix....4
A. timeapp.py....4
B. Test No.1 “Alice”....4
C. Test No.2 “Alice/4”....5
VI.Samples....5
A. All samples use the Alice test file from above....5
B. flite – KAL (diphone)....5
C. flite – KAL16 (diphone)....5
D. flite – SLT (clustergen)....5
E. flite – AWB (clustergen, Scottish dialect, not used in the tests)....5
F. eSpeak – default voice....5

I. Procedure

A. Hardware

All tests were performed on a Fedora 12 (kernel 2.6.31.12-174.2.22.fc12.x86_64) machine with a Core Duo P9600 CPU (clocked at 2.66 GHz)

B. Versions used

flite-1.4-release December 2009

eSpeak text-to-speech: 1.26 09.Jun.07 (the last GPLv2 version)

C. Speed tests

I wrote a script, timeapp.py, that runs a given command 50 times through the os.system call (identical to the C system() call) and prints out the minimum, average, and maximum times. There is some overhead from the os.system call but it's constant, so the results are comparable.

D. Memory tests

I used valgrind's massif tool to investigate the amount of memory allocated (both stack and heap) and (more importantly) note the maximum amount for several scenarios.

When testing flite, I used the versions that had only 1 voice compiled (as opposed to the generic flite application that has all bundled voices compiled in), as that is likely to be the case for Rockbox and the generic app had slightly larger memory usage.

Command used:

```
valgrind --tool=massif --max-snapshots=1000 --stacks=yes --time-unit=B
```

II. General info

A. Application binary sizes

All sizes are of the stripped binaries (strip --strip-all).

It is important to note that flite links the voices statically while eSpeak separates them in a data directory to be processed at runtime. Thus, I have provided the sizes of flite linked with each of the default voices, as well as with all of them.

Description	Filename	Size
flite + the AWB voice (clustergen)	flite_cmu_us_awb	3.7 M
flite + the KAL voice (diphone)	flite_cmu_us_kal	2.7 M
flite + the KAL voice (sampled at 16kHz) (diphone)	flite_cmu_us_kal16	4.3 M
flite + the RMS voice (clustergen)	flite_cmu_us_rms	4.2 M
flite + the SLT voice (clustergen)	flite_cmu_us_slt	3.7 M
flite linked with all of the voices above	flite	19 M
eSpeak main binary + library	espeak + libespeak.so	12 K + 161 K = 183 K
All 5 English voices + phoneme data	espeak-data/	387 K

B. Conclusions

Clearly, flite's concatenative synthesis requires more data and thus more space. However, I personally feel that the difference in quality outweighs the difference in space used. Further, the clustergen voices can be made smaller (at the cost of degraded fluency), an area worth looking into.

III. Speed tests

A. Results

Description	Minimum	Average	Maximum	Output length	Avg. speed
flite (voice KAL) (Test No.1 “Alice”)	0.123165 s	0.130493 s	0.174572 s	61 s	467x realtime
flite (voice KAL16) (Test No.1 “Alice”)	0.171015 s	0.177551 s	0.239413 s	123 s	693x realtime
flite (voice SLT) (Test No.1 “Alice”)	4.371572 s	4.588671 s	5.672501 s	113 s	25x realtime
eSpeak (default voice) (Test No.1 “Alice”)	0.256182 s	0.267296 s	0.365111 s	114 s	426x realtime

B. Conclusions

Clearly, the simplest diphone voice proved to be the fastest. However, the results of the KAL16 voice are also very interesting – although the output was twice as long, its performance matched that of its simpler sibling. This goes to show the scalability of the diphone engine.

Clustergen voices seem to be rather slow (and, as will later be shown, memory-intensive), however, the results show the greatest fluency and overall quality (Samples).

The length of the generated output also shows that the basic diphone voice produces output that's too fast and might not be suitable for long texts. Note: eSpeak allows run-time selection of both output speed and pitch.

IV. Memory tests

A. Results

Description	Peak memory usage (Test No.1 “Alice”)	Peak memory usage (Test No.2 “Alice/4”)
flite (voice KAL)	1.252 M	886.8 K
flite (voice KAL16)	1.787 M	1.247 M
flite (voice SLT)	19.80 M	12.93 M
eSpeak (default voice)	456.5 K	456.5 K

B. Conclusions

Flite's memory usage seems to increase with the length of the generated text due to the fact that the system keeps track of the relations between the phonemes in the entire text, not just the current sentence/segment. Rewriting this would not be trivial but should be doable, if it proves to be a problem. However, diphone voices have a small footprint already, so I don't think it will be an issue. We can also keep the memory footprint low by splitting the text into sentences and passing each sentence to the engine.

Clearly, clustergen voices are unsuitable for any synthesis longer than a few seconds, so I think my main focus should be on getting diphone voices working fast enough.

Further, it is worth noting that flite causes a considerable amount of memory fragmentation. I believe this is due to the fact that the basic data structure is 8-bytes long and it gets created/deleted a lot. Choosing a good malloc implementation will likely remedy (if not completely solve) this issue. It is important that said implementation aligns addresses on an even boundary since flite uses a union of a pointer and an int to determine the type of the aforementioned struct (the int is odd if it's set by flite, even if it's actually a pointer; for a clarification, see **`include/cst_val.h`** in flite's source).

Various experiments (not documented here) have shown eSpeak to have a constant memory footprint, regardless of text length but depending on the selected voice. This is clearly preferable but the results are sub-par.

V. Appendix

A. timeapp.py

```
import timeit , os, sys
def average(values):
    return sum(values, 0.0) / len(values)

runs = 50
app = " ".join(sys.argv[1:])
stmt = "os.system(\"%s\")" % (app)
print "Command: \"%s\"" % (app,)
timer = timeit.Timer(stmt, "import os")
times = timer.repeat(runs, 1)
print "Times from %i runs: min=%f, average=%f, max=%f" % (runs,
min(times), average(times), max(times))
```

Usage: `python timeapp.py <app> [<arg1>] [<arg2>] ...`

B. Test No.1 “Alice”

This is a 405-word excerpt from *Alice's Adventures in Wonderland* by Lewis Carroll.

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do. Once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, "and what is the use of a book," thought Alice, "without pictures or conversations?" So she was considering in her own mind (as well as she could, for the day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.

There was nothing so very remarkable in that, nor did Alice think it so very much out of the way to hear the Rabbit say to itself, "Oh dear! Oh dear! I shall be too late!" But when the Rabbit actually took a watch out of its waistcoat-pocket and looked at it and then hurried on, Alice started to her feet, for it flashed across her mind that she had never before seen a rabbit with either a waistcoat-pocket, or a watch to take out of it, and, burning with curiosity, she ran across the field after it and was just in time to see it pop down a large rabbit-hole, under the hedge. In another moment, down went Alice after it!

The rabbit-hole went straight on like a tunnel for some way and then dipped suddenly down, so suddenly that Alice had not a moment to think about stopping herself before she found herself falling down what seemed to be a very deep well.

Either the well was very deep, or she fell very slowly, for she had plenty of time, as she went down, to look about her. First, she tried to make out what she was coming to, but it was too dark to see anything; then she looked at the sides of the well and noticed that they were filled with cupboards and book-shelves; here and there she saw maps and pictures hung upon pegs. She took down a jar from one of the shelves as she passed. It was labeled "ORANGE MARMALADE," but, to her great disappointment, it was empty; she did not like to drop the jar, so managed to put it into one of the cupboards as she fell past it.

C. Test No.2 “Alice/4”

This is a 112-word excerpt from *Alice's Adventures in Wonderland* by Lewis Carroll.

Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do. Once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, "and what is the use of a book," thought Alice, "without pictures or conversations?" So she was considering in her own mind (as well as she could, for the day made her feel very sleepy and stupid), whether the pleasure of making a daisy-chain would be worth the trouble of getting up and picking the daisies, when suddenly a White Rabbit with pink eyes ran close by her.

VI. Samples

A. All samples use the Alice test file from above.

B. flite – KAL (diphone)

http://dl.dropbox.com/u/137119/ttssamples/alice_flite_kal.flac

C. flite – KAL16 (diphone)

http://dl.dropbox.com/u/137119/ttssamples/alice_flite_kal16.flac

D. flite – SLT (clustergen)

http://dl.dropbox.com/u/137119/ttssamples/alice_flite_slt.flac

E. flite – AWB (clustergen, Scottish dialect, not used in the tests)

http://dl.dropbox.com/u/137119/ttssamples/alice_flite_awb.flac

F. eSpeak – default voice

http://dl.dropbox.com/u/137119/ttssamples/alice_espeak.flac