Flite:
a small run-time synthesizer

`http://cmuflite.org`

Alan W Black (awb@cs.cmu.edu)
Language Technologies Institute
Carnegie Mellon University

Flite – Festival-lite

- ☐ Small, fast, portable run-time synthesizer
- ☐ Festival "compatible"
- ☐ **not** a replacement, but companion
- ☐ C for size and portability
- ☐ Voices get "compiled" into static structures:
  – offering same quality as festival voice
- ☐ Targets:
  – less than 4 meg (flite+lex+diphone voice)
  – options to be half (and half again)
  – time to speak: less than 0.250 seconds (on ipaq)
  – thread safe
  – no loading (copying) of data, all const
  – no Scheme, no interpreter, no gc

Flite: motivation

- Festival is too slow, to big, and not portable:
  - "OK, but does it scale ..."
  - synthesis advocacy allows no excuses
- Speech technology has to be ubiquitous
- "But its just an engineering problem ..":
  - "we'll put a team of 20 programmers on it ..."
  - need to understand task be a good programmer
  - helps to have built previous systems
- Research questions:
  - trade-off between size/quality/speed
  - what's **really** necessary in a synthesis engine
  - can components be simplified

Target audiences

- Embedded systems: PDA:
  - local rendering of speech on PDA
  - digital radios, books
  - toys and games
- Server systems:
  - multi-channel telephone dialog systems
- But these are "just" commercial uses:
  - No, not exclusively
  - these are also for the speech/dialog researcher
  - when a component can be easily added it will be

Flite components

- Flite library: with CST "C Speech Tools":
  – core code (shared between languages/voices)
- Lang:
  – language models: text analysis prosody etc
  – lexicon: words plus lts,
  – voice: waveforms/mcep/lpc, index (from festvox/)

Each goes to distinct lib:
libflite.a
libflite_usenglish.a
libflite_cmulex.a
libflite_cmu_us_kal.a

Some general points

- static consts where possible:
  - cart trees, waveforms, etc converted C structures
  - scheme code used to do convertion
- dynamically allocated vals linking into utt:
  - therefore no gc required
  - except utterance deletion
- avoid non-const globals

Non-global current voices

- Festival: uses notion of "current voice":
  - set as global
  - used by each utterance
  - (other globals too)
- Flite: link voice with each utterance:
  - voice is global (const)
  - voice linked to synthesis functions
  - (e.g. appropriate text analysis, F0 model functions
  - and appropriate models, duration cart etc)

CST: “C Speech Tools” I

- audio/:
  - output, client/server, (no i yet, o only)
- utils/
  - alloc, endian, error, socket, strings
  - vals (int,flt,str, plus user defined)
  - features
  - tokenstream
- regex/ (Henry Spencer’s regex lib)
  - festival compatible
- speech/
  - waves, tracks
- stats/
  - cart interpreter, viterbi decoder
- hrg/
  - items, relations, utts, paths, feature functions

CST II: flite

- lexicon/
  - lexicon index support
  - lts rule interpreter
- synth/
  - public flite functions
  - generic synth functions (voice parameterisable)
  - voice structure, phonesets
- wavesynth/
  - diphone (lpc resynthesis)
  - prosody modification
  - clunits support (ldom/unit selection), to be completed

lang/usenglish/

US English specific models

- text analysis:
  - number/symbol expansion
  - pretty basic at the moment
- prosody:
  - autoconverted CART trees from Festival voices
- phrasing:
  - hand written cart and converted
- US English feature functions

```
(set! phrase_cart_tree
'((R:Token.n.name is 0)
  ((R:Token.parent.punc is ",")
   ((BB))
   ((R:Token.parent.punc is ".")
    ((BB))
    ((NB))))
  ((n.name is 0)
   ((BB))
   ((NB)))))
```

```
static const cst_cart_node us_phrasing_cart_nodes[ ] = {
{ 0, CST_CART_OP_IS, CTNODE_NO_0000, &val_0000},
{ 1, CST_CART_OP_IS, CTNODE_NO_0001, &val_0001},
{ 255, CST_CART_OP_NONE, 0, &val_0002 },
{ 1, CST_CART_OP_IS, CTNODE_NO_0003, &val_0003},
{ 255, CST_CART_OP_NONE, 0, &val_0002 },
{ 255, CST_CART_OP_NONE, 0, &val_0004 },
{ 2, CST_CART_OP_IS, CTNODE_NO_0006, &val_0000},
{ 255, CST_CART_OP_NONE, 0, &val_0002 },
{ 255, CST_CART_OP_NONE, 0, &val_0004 },
{ 255, CST_CART_OP_NONE, 0, 0}};

DEF_STATIC_CONST_VAL_STRING(val_0000,"0");
DEF_STATIC_CONST_VAL_STRING(val_0001,",");
DEF_STATIC_CONST_VAL_STRING(val_0002,"BB");
DEF_STATIC_CONST_VAL_STRING(val_0003,".");
DEF_STATIC_CONST_VAL_STRING(val_0004,"NB");

static const char * const us_phrasing_feat_table[] = {
"R:Token.n.name",
"R:Token.parent.punc",
"n.name",
NULL };

const cst_cart us_phrasing_cart = {
  us_phrasing_cart_nodes,
  us_phrasing_feat_table
};
```

lang/cmulex/

- Pronunciations:
  - fsm word to pronuciation or “use lts”
  - minimised packed lts decision graphs (Black&Lenzo 98)
- lexicon:
  - addenda (simple list)
  - word list
  - lts rules
  - syllabifier
- comment:
  - the word list/phone list is well over 2.9M for cmulex
  - can we remove infrequent words

cmu_us_kal/

Generated from a festvox/ voice

- cmu_us_kal_idx.c
  - unit (diphone) index
  - compressed pitch periods (coeffs, plus residual)
  - autoconverted
- cmu_us_kal.c
  - voice definition
  - what gets used for what

Current sizes

- no leaks
- thread safe (after short initialization of voice)
- With current 8KHz diphone voice:
  - flite code 50K
  - usenglish 25K
  - cmulex 2.9M
  - unitdb 2.1M
- Runtime RAM requirements:
  - about 4 x utt duration * sample rate
  - less than 1M for 2 chapters of "Alice in Wonderland"
- Speed:
  - On 500MHz PIII, about 60 times faster then real time
  - 20 mins of speech takes 20 seconds to synth
  - thus about 10MHz per port
- Festival (tuned):
  - 2.5M code, 20M run-time, 10 times fast than real time

Improvement: Streaming synthesis

- ☐ Flite (and festival) synthesis
  - – utterance by utterance
  - – process full utt, and make full waveform
- ☐ Streaming synthesis:
  - – don't build full waveform
  - – play/send as its built
  - – must do (all ?) text, phrase, prosodics first
  - – but waveform regeneration can be streamed
- ☐ Flite advantages would be
  - – time to first noise, less than %50
  - – reduce mem footprint to 0.25 (at least)

Improvement: Unit database compression

- LPC resynthesis:
  - Requires LPC coefficients (pitch synchronous)
  - Residual (small dynamic range)
- Optimise size vs quality:
  - number of coefficients
  - quantizing coefficients and/or residual
  - (could use spike excited for very small footprint)
- Unit selection:
  - re-use units:
  - e.g. all diphones into stop can be shared

Target footprints

- Very small (will be poorer quality):
  - all less than 1M
  - compressed minimal lexicon
  - streaming synthesis
  - spike excited LPC (quantised ?)
- Small:
  - 2.5M
  - telephone quality speech
  - lex, etc as with Festival
- Medium:
  - 4M
  - 16KHz speech

Todo...

- Support clunits/ldom
  - DB will always be larger than diphone
  - build process from FestVox format
  - should be fast and much smaller
- Text analysis:
  - Should really be using the NSW models
- Fixed point version
- Documentation
- Some key demostration projects:
  - iPAQ, itsy etc

First Release

- Intended release end of Feb '01
  - under BSD-style licence like Sphinx/Festival
- Release will include:
  - core flite library
  - plus basic USEnglish
  - pruned lex based on cmulex
  - 8 KHz diphone voice
- probable size:
  - at about the 4M level
  - (i.e. about twice the size I'd like it to be)