Objectives

- understand basic processes in speech synthesis
- understand relative complexity of implementing solutions to problems
- become familiar with Festival's architecture and know what it can and can't do.

After the course, you will

- be able to make Festival speak what you want
- be able to influence the way it does it
- be able to adapt it for your applications
- be able to explain how the system works
- be able to build simple voices within the system

Text to speech

Four major topics in speech synthesis

- Architecture
  – objects and processes required
- Text processing
  – from text to tokens to utterances to words.
- Linguistic/prosodic processing
  – lexicons, phrasing, intonation, duration
- Waveform synthesis
  – diphone, unit selection, concatenation and modification

Course outline

- March:
  - history, basic Festival use
  - TTS, Utterance structure, processes
  - Text analysis, Lexicons, LTS
  - Prosody: phrasing, intonation, duration and power
  - Prosody modelling techniques
- April:
  - Large projects:
  - Waveform synthesis: diphones, unit selection
  - Building a new voices in new languages
  - Limited domain synthesis
- May:
  - Project time
  - Concept to speech, Voice Transformation
  - Speech Synthesis evaluation and diagnosis

Course evaluation

- (approx) weekly small exercises:
  - best 4 contribute to grade
- 1 "large" project:
  - set beginning of April
  - e.g. build a new voice
  - Requires (appropriate) writeup.
- No exam

Important weblinks

- Course notes, slides, exercises
  - `http://www.cs.cmu.edu/~awb/11752.html`
- Building Voices in Festival
  - `http://www.festvox.org`

History: Speech Synthesis

audio examples from
`http://www.festvox.org/history/klatt.html`
From Dennis Klatt's 1987 JASA 82, 737-793 article.
(Sampled history, this isn't eveything).

- **1930s**:
  - 1936: UK Telephone Company's speaking clock, used optical storage, with phrases, words and part-word concatenation
  - (1) 1939: Bell Lab's Homer Dudley developed VODER mechanical (organ like) device that can "play" speech.
  - Dudley's VOCODER, devised by decomposing speech signal into source/filter model.

History: Speech Synthesis

- **1940s, 50s**:
  - Analogue based models
  - (4) OVE formant synthesis (Gunnar Fant) 1953
  - (11) The DAVO articulatory synthesizer developed by George Rosen at M.I.T., 1958.
- **1960s**:
  - Digital models and text-to-speech, prosodic rules
  - (17) Elegant rule program for British English by John Holmes, Ignatius Mattingly, and John Shearne, 1964.
  - (19) Rules to control a low-dimensionality articulatory model, by Cecil Coker, 1968.
  - (20) First prosodic synthesis by rule, by Ignatius Mattingly, 1968.

History: Speech Synthesis

- **1970s**:
  - TTS, products, and diphones
  - (24) The first full text-to-speech system, done in Japan by Noriko Umeda et al., 1968.
  - (21) Sentence-level phonology incorporated in rules by Dennis Klatt, 1976.
  - (22) Concatenation of linear-prediction diphones, by Joe Olive, 1977.
  - (28) The inexpensive Votrax Type-n-Talk system, by Richard Gagnon, 1978.
  - (30) The M.I.T. MITalk system by Jonathan Allen, Sheri Hunnicut, and Dennis Klatt, 1979.

History: Speech Synthesis

- **1980s**:
  - concatenative speech, larger systems
  - (34) The AT&T Bell Laboratories text-to-speech system, 1985.
  - (35) Several of the DECtalk voices.
  - (36) DECtalk speaking at about 300 words/minute.

History: Speech Synthesis

- **1990s**:
  - products, multi-lingual, unit selection
  - Japanese NUU-talk system, Sagisaka, Iwahashi, ATR, 1992
  - General unit-selection CHATR 1994
    (diphones/unit selection example from Festival 1996)
  - Free software and everyone joins in, MBROLA 1995, Festival 1996.
  - Unit selection in AT&T's NextGen system 1999.

But ...

- Commercial systems still use recorded prompts
- and word concatenation (cf. 1936)
- Because it sounds better.

Festival: a generic speech synthesis system

- multi-lingual text to speech
  - synthesis for language systems
    - synthesis development environment

Festival Speech Synthesis System

- Open Source speech synthesizer system:
  - designed for development *and* runtime use
- Used in many commercial and academic systems:
  - from large (AT&T) to small (various startups)
  - distributed with RedHat 8.x
  - hundreds of thousands users
- Multi-lingual
  - no built-in language
  - designed to allow addition of new languages
- Additional tools for rapid voice development:
  - guidelines and documentation
  - autoaligner for labelling recorded speech
  - statistical learning tools
  - scripts for building models

Festival as software

`http://festvox.org/festival/`

- General system for multi-lingual TTS
- C/C++ code with Scheme scripting language
- General replaceable modules:
  - lexicons, LTS, duration, intonation, phrasing, POS tagging, tokenizing, diphone/unit selection, signal processing
- General tools:
  - intonation analysis (F0, Tilt), signal processing, CART building, n-gram, SCFG, WFST, OLS
- No fixed theories
- New languages without new C++ code
- Multiplatform (Unix/Windows NT)
- Full sources in distribution
- Free software

The CMU FestVox project

`http://festvox.org`

- Festival is an engine, how do you make voices
- Festvox: building synthetic voices:
  - Tools, scripts documentation
  - Discussion and examples for building voices
  - Example voice databases
  - Step by step walktrhoughs of processes
- Support for English and other languages
- Support for different waveform synthesis methods:
  - diphone
  - unit selection
  - limited domain
- Other support:
  - text analysers
  - prosodic modelling
  - lexicon building

The CMU Flite project

`http://cmuflite.org` **FLITE** a fast, small portable run-time synthesis engine

- C based (no loadable files)
- Basic FestVox voices compiled into C/data
- Thread-safe
- Suitable for embedded systems and multi-client servers
  - Ipaq, Linux, WinCE, etc
- Scalable:
  - quality/size/speed trade offs
  - frequency based lexicon pruning
  - quality/speed db compression
- Sizes:
  - less than 4Meg footprint (code+data+runtime RAM)
  - less that 0.025 secs time to speak (streaming synthesis)

Synthesis Tools

- I want my computer to talk:
  - Festival Speech Synthesis System
- I want my computer to talk in my voice:
  - FestVox Project
- I want it to be fast and efficient:
  - Flite

Using Festival

- How to get Festival to talk
- Scheme (Festival's scripting language)
- Basic Festival commands
- Exercises

Getting it to talk

- Say a file
  - `festival --tts file.txt`
- From Emacs
  - say region, say buffer
- Command line interpreter
  - `festival> (SayText "Hello")`

Scheme – Festival's scripting language

- Why:
  - too many options
  - need flexibility
  - easy to add functionality

- Why Scheme:
  - very simple
  - very powerful
  - well established
  - authors are familiar with it

Bluffer's guide to Scheme

Scheme is a dialect of Lisp

- **expressions** are
  - atoms or
  - lists

  a bcd "hello world" 12.3
  (a b c) (a (1 2) seven)

- Interpter evaluates expressions
  - atoms evaluate as variables
  - lists as function calls

```
festival_version
3.14
(+ 2 3)
```

Bluffer's guide to Scheme

- Setting variables
  - `(set!  a 3.14)`
- defining functions
  - `(define (timestwo n) (* 2 n))`

```
(timestwo a)
6.28
```

Scheme: lists

```
festival> (set!  alist '(apples pears bananas))
(apples pears bananas)
festival> (car alist)
apples
festival> (cdr alist)
(pears bananas)
festival> (set!  blist (cons 'oranges alist))
(oranges apples pears bananas)
festival> (append alist blist)
(apples pears bananas oranges apples pears
bananas)
festival> (length alist)
3
festival> (length (append alist blist))
7
```

Scheme: speech

Make an utterance of type text

```
festival> (set! utt1 (Utterance Text "Hello"))
<UTT 982345>
```

Synthesize an utterance

```
festival> (utt.synth utt1)
<UTT 982345>
```

Play waveform

```
festival> (utt.play utt1)
<UTT 982345>
```

Do all together

```
festival> (SayText "This is an example")
<UTT 983277>
```

```
;; In a file
(define (SpeechPlus a b)
   (SayText
    (format nil
     "%d plus %d equals %d"
     a b (+ a b))))
```

Loading files

```
festival> (load "file.scm")
t
festival> (SpeechPlus 2 4)
<UTT 839727>
```

```
(define (sp_time hour minute)
(cond
 ((< hour 12)
  (SayText
   (format nil
    "Its %d %d in the morning"
    hour minute)))
 ((< hour 18)
  (SayText
   (format nil
    "Its %d %d in the afternoon"
    (- hour 12) minute)))
 (t
  (SayText
   (format nil
    "Its %d %d in the evening"
    (- hour 12) minute)))))
```

Getting help

- Online manual
  `"http://festvox.org/docs/manual-1.4.3/''`
- alt-h on current symbol short help
- alt-s to speak help
- alt-m goto manual page
- Use TAB key for completion

Lexicons and Lexical entries

Festival *will* make mistakes in pronunciation.
You can explicitly give pronunciations for words.

- Each language/dialect has its own lexicon
- You can lookup words with
  `(lex.lookup WORD PartOfSpeech)`
- You can add entries to the current lexicon
  `(lex.add.entry NEWENTRY)`
- And entry consists of
  (WORD POS (SYL0 SYL1 ...))
- a Syllable is
  ((PHONE0 PHONE1 ...) STRESS)
- `'("cepstra" n (((k eh p) 1) ((s t r aa) 0))))`

Exercises

by noon March 10th to `antoine@cs.cmu.edu`

1. * Make Festival say your name, and then everyones name in the class. Add explicit pronunciations to the lexicon.
2. Use Festival to say selected pieces of text. Find ten things Festival fails on.
3. How long does it take for Festival to say *Alice's Adventures in Wonderland*?

Hints

1. to test pronunciations, use
   `(SayText "My name is John Smith.")`
   to set new entries in the lexicon use
   ```
   (lex.add.entry
   '("edinburgh" n (((eh d) 1) ((ah n) 0)
   ((b ax ) 0) ((r ow) 0))))
   ```
   To find out phone names use
   `(lex.lookup "word" nil)`
   on similarly pronounced words
2. See Festival manual for instructions for installation
   `"http://festvox/docs/manual-1.4.3/"`
3. use
   `$SPPPDIR/src/festtut/examples/books/alice29.txt`
   an *estimate* of the time is fine.