Large Applications Practicum

Group-3

Design Document

Vishnu Priya Jindal (B16041)

Ananya Shukla (B17007)

Aman Saxena (B17034)

Aaditya Arora (B17071)

October 2019

Evoice: The **Text-to-Speech** Synthesizer

Design Document

Revision History

Version	Date	Author(s)	Description
v1.0	10/16/19	Vishnu Priya Jindal, Aaditya Arora, Aman Saxena, Ananya Shukla	Initial version.
v1.5	10/22/19	Vishnu Priya Jindal, Aaditya Arora, Aman Saxena, Ananya Shukla	Pdf support added.
v2.0	11/05/19	Vishnu Priya Jindal, Aaditya Arora, Aman Saxena, Ananya Shukla	Final version.

Table of Contents

1 Introduction

Evoice is an offline GUI-based text-to-speech synthesizer for English and other languages, developed in order to help the visually impaired people, school going students - using computer generated voice which can read the text to the user.

1.1 Design Overview

Evoice is based on Festival TTS which is a C++ library that offers text to speech through several APIs. *Evoice* has created a Python2 based wrapper that calls the functions from Festival. Functions in Festival TTS are in C++. The GUI is built in Python2 and reads text either from the input field or from a file.

1.2 Intended Audience

This document is intended for software designers and students who have an interest in the field of software development. This may also be read by the users of this product.

1.3 References

[1] http://espeak.sourceforge.net/

[2] http://www.cstr.ed.ac.uk/projects/festival/

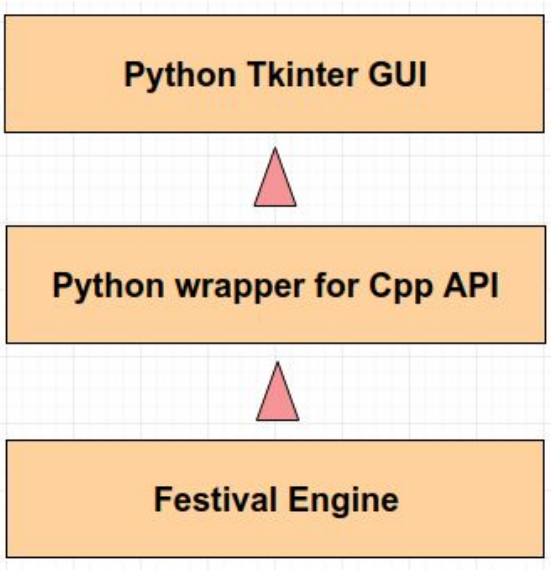
[3] https://en.wikipedia.org/wiki/Speech_synthesis

2 Detailed Design

The software is written in Python2 and uses the C++ APIs of Festival.

2.1 Architecture

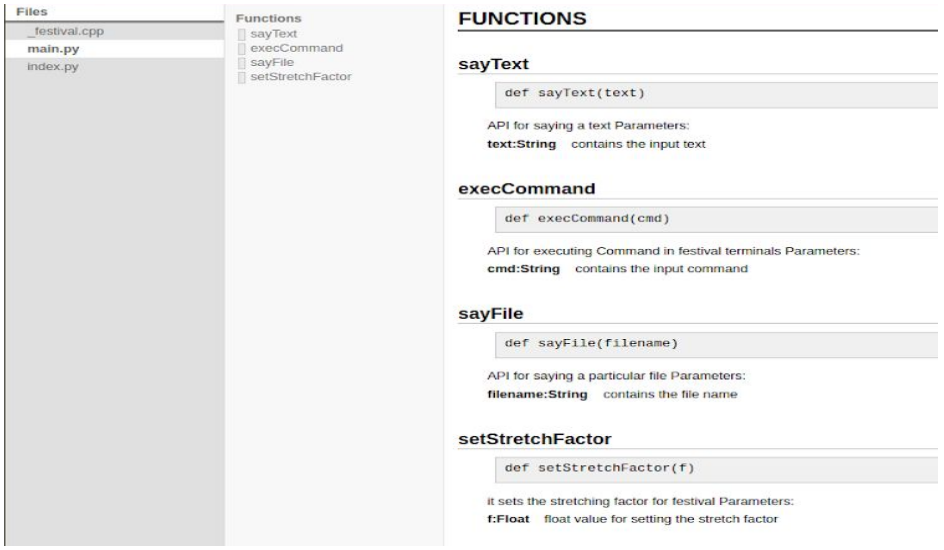
The product is built on the Festival TTS that provides C++ APIs to read the text in a computer generated voice. It creates Python wrappers to call the C++ functions provided by Festival TTS.



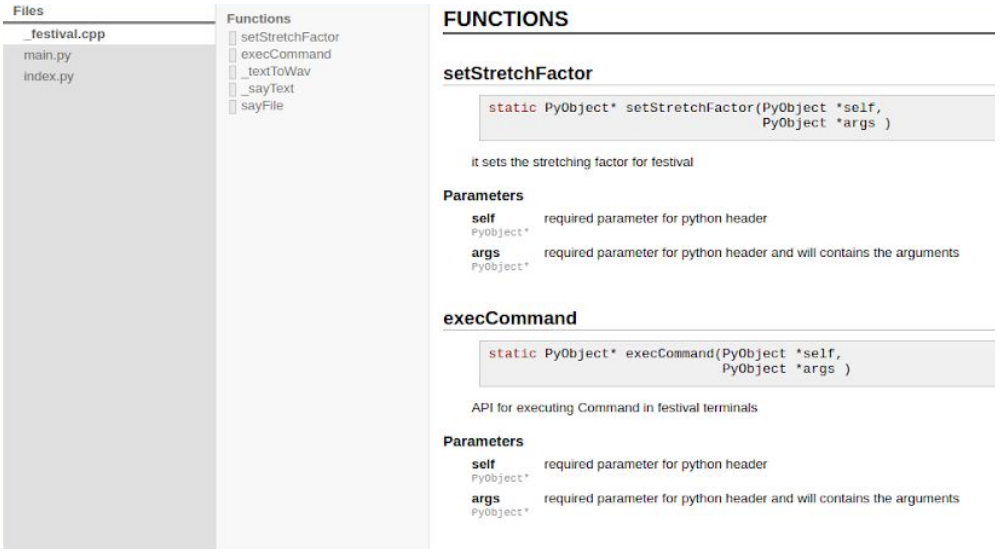
Components

Festival is used as-is, offering APIs for text to speech conversion.

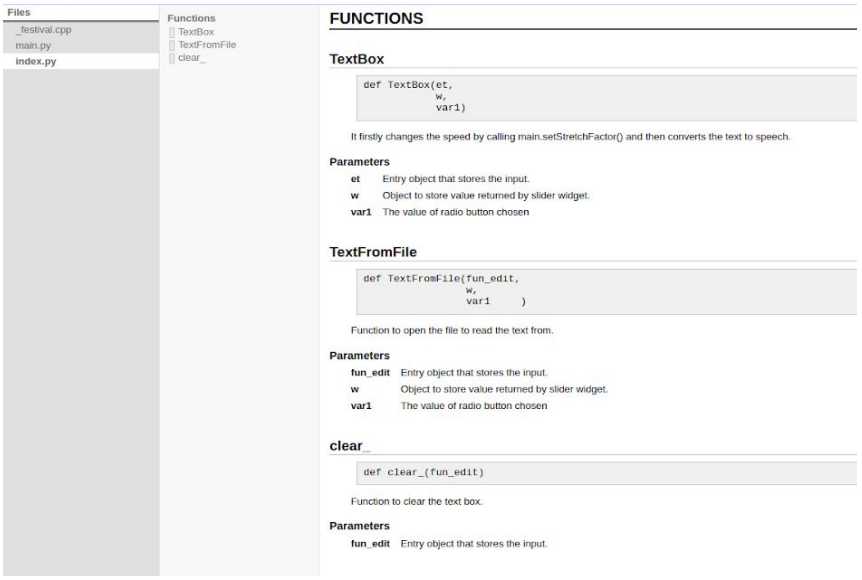
Main.py contains the wrappers in Python2 that are used to call the C++ functions of Festival.



_Festival.cpp Contains the C++ function which can be called from Python to call the built-in functions of festival API.



GUI built using Tkinter library that provides the text field for input, slider for the rate and an option to choose a text file.



Functionalities

1. We can convert text, from the dialogue box (GUI) as well as from any file in .TXT, . PDF format to speech.
2. We can change the speed of speaking.
3. Inter-line and inter-para pause feature is added.
4. We can highlight the text, which we want to be read out.

Interfaces

The Backend of the application is written in Python and is using Festival C++ API. The Frontend of the application is also written in Python and is connected to the backend by importing the functions defined in the backend.

GUI

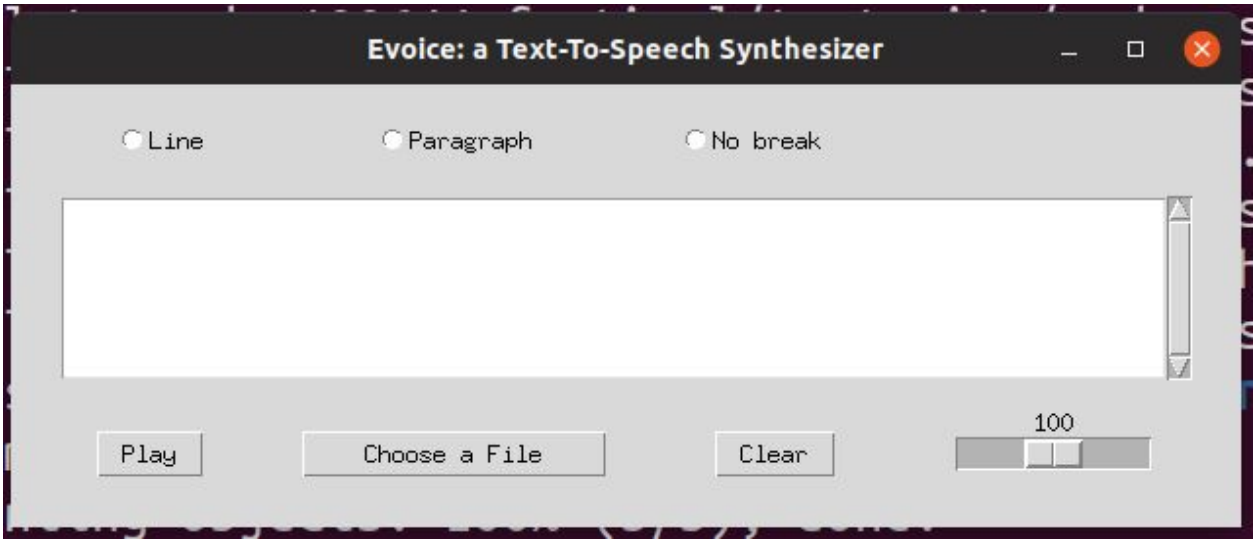


Fig 1. Basic GUI of the application. The three radio buttons - Line, Paragraph and No break are for the pause options. Scroll-text-space is to write the text that has to be spoken. 'Choose a File' button is to select file to be read out.' Clear' button is to remove the contents from text field. Scroller at bottom right is to adjust speed.

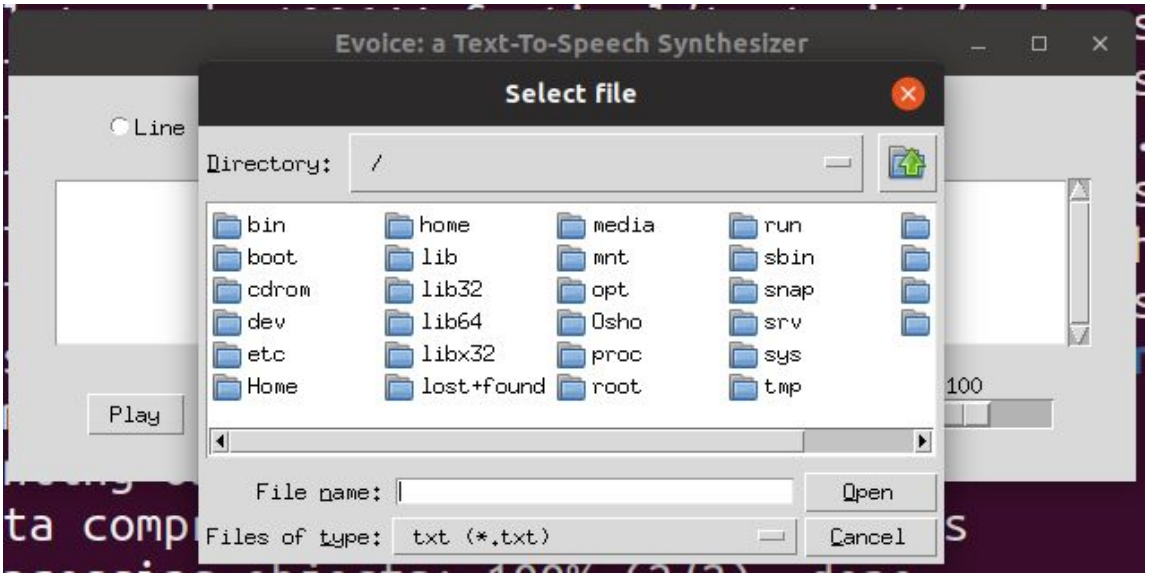


Fig 2. This GUI is to select a file to be read out when 'Choose a File' button is pressed.

2.2 Algorithms and Data Structures

Code snippets for our project are :

1. For reading from dialogue box/file :

```
ad = tkFileDialog.askopenfilename(initialdir = "/",title = "Select file",filetypes =
(("txt","*.txt"),("pdf","*.pdf"),("all files","*.*")))
 if(".pdf" in ad):
  with open(ad, "rb") as f:
   pdf = pdftotext.PDF(f)
   main.sayText("\n\n".join(pdf))
 else:
  with open(ad, 'r') as myfile:
   s = myfile.read()
   # print s
   if s == '\n':
    return 1
   if var1.get() == 2:
    s=s.split('\n')
    for i in range(len(s)):
     if not s[i].strip():
      continue
     main.sayText(s[i])
```

2. For speed change :

```
def TextBox(et,w,var1):
 x = 2 - (w.get()*1.0)/100
 if x <0.1:
        x = 0.1
 main.setStretchFactor(x)
```

3. Speaking Highlighted text :

```
 ad = str(et.get('1.0', END))
 if et.tag_ranges(SEL):
  ad = et.get(SEL_FIRST
,SEL_LAST)
```

4. Pause feature :

```
if var1.get() == 2:
  ad=ad.split('\n')
  for i in range(len(ad)):
    if not ad[i].strip():
      continue
    main.sayText(ad[i])
    result = tkMessageBox.askyesno("Python","Do you want to continue?")
    if not result:
      break
if var1.get() == 1  :
  ad=ad.split('.')
  for i in range(len(ad)):
    if not ad[i].strip():
      continue
    main.sayText(ad[i])
    result = tkMessageBox.askyesno("Python","Do you want to continue?")
    if not result:
      break
if var1.get() == 3 or var1.get() == 0:
  main.sayText(ad)
```

2.3 External Data

Files and Database

Evoice provides the users with an option to read the text files apart from the manual method. Also, we can highlight the text provided to be read out.

2.4 Performance

The performance of the product can be tested by running files of different sizes.

3 Future Improvement Scope

Some further future improvements that could be implemented could be :

1. Adding intra-line pause.
2. Extending highlight feature while reading through files too.
3. Adding ‘Repeat’ button to read out the last sentence.