An Interactive System design in a Pushbutton-Input Auditory-Output Interface for Listening to An Audio Book

Programmer's Documentation

Author: Mamtaj Akter

October 27th, 2019

Introduction

This document discusses an interactive system design that helps the user to select a chapter of an audiobook and then continue to listen to that chapter. The program is written in Python 3 and uses the readchar, simpleaudio, multiprocessing, and time packages. To run the application, the user needs to install these packages first and then run the command "Python3 ks_main.py" in the terminal. ks_main.py is the main program that depends on several helper programs and all these program files need to be in the same directory. This program also requires some audio files that are stored in "wav_files_provided" and "project_wav_files" folders which need to be on the same directory where the program files remain.

System Flow

The program starts with an auditory help message "Select Book; Press <SPACE> to select, <J> to go backward, <K> to go forward, <L> for help, <;> to quit." The user can then navigate through the booklist. At present, we have only one audio book in the wav file directory. After selecting the book, the user can scroll through the chapter numbers and names using the forward key <K> and the backward key <J>. To select any chapter, the user needs to press select key <SPACE>. Users can ask for the help message at any point by pressing help key <L> and also can exit from the program by using the exit key <;>. Once the user selects the chapter 2, an auditory message "Continue to read. <book_title>" is played and the program starts playing the text contents of chapter two. However, since we only have the chapter two contents of the auditory book, the auditory message "<Chapter_number> not available, Press <;> and then press <j> to select another chapter" is played if user selects any other chapters. Essentially, <;> and <J> these two keys allow users to get the option to choose a chapter again. When the chapter 2 content is being played, in the middle of listening, the user can exit from the program by pressing the <;> key twice and then an auditory prompt "Exiting Program" gets played. Also, the user can go back to the previous menu: *Book Selection State* or to the *Chapter Selection State*, to select other books or other chapters respectively through sequentially pressing <;> and <J> keys.

System States and Auditory Help/Quit Systems

If the interactive system requires any additional pushbutton for inputs or to change any of the current pushbuttons, a programmer needs to edit the method named "keystroke_processor" in ks_main.py. There are four important local variables used in this method. First, *systemState* - a variable that tracks the current state of the system. This variable holds values from 0 to 2. When a user first enters into the program, the *systemState* variable gets initialized to 0. The initial state is essentially the *Book Selection State*. As the user selects a book, this variable becomes 1 and the *Chapter Selection State* gets activated. Finally, as soon as the user selects a chapter and the system starts reading the wav files of that chapter, the *systemState* gets incremented by 1, this state is

called *Continue Reading State*. *systemState* gets the reset value 0 again whenever it becomes more than 2, and thus the system goes back to the START state again to let users select another book. The next variable, *bookNumber*, basically holds only 0 as we have only one audio book in our audio directory. The third variable is *chapterNumber* – for each forward key pressed, this variable gets incremented by one until it reaches to its maximum number 6, and it gets its initial value 0 as soon as it gets a value that is more than 6. The value of *chapterNumber* gets decreased by one for each backward key is pressed. Likewise, the fourth local variable, *readItemNumbers*, gets an initial value 0. It increments by 1, each time the user scrolls forward and gets decremented by 1 as the user scrolls forward.

For the introductory auditory help message, the function "play_intro" is called from the "keystroke_processor" method. However, for the usual help message, the function named "play_help" is called and what messages will be announced, that depend on which state the system is currently in. When the system is in *Book Selection State*, the auditory help message is "Select Book; Press <J> to scroll backward; press <K> to scroll forward; Press <SPACE> to select; Press <;> to quit". The same help message precedes with "Select Chapter" instead of "Select Book" when the program is in the *Chapter Selection State*. During the *Continue Reading State*, the help auditory message is relatively smaller, "Press <J> to scroll backward; press <K> to scroll forward; Press <;> to quit".

At any states, user can quit from the program through pressing <;> key twice. This interactive design allowed user to have option if he or she really wants to quit or not. However, like the help message system, the quit message also varies according to the state the system is currently in. When the user is in *Book Selection State*, the auditory quit message is just "Press <;> again to quit". Pressing a second <;> key, user can exit from the program with an auditory announcement "Exiting Program". The same help message is played with the additional announcement "Press <J> to scroll to the previous menu" when the user is either in the *Chapter Selection State* or in the *Continue Reading State*. Therefore, the user gets the flexibility of going back to the previous state through pressing a <;> key and then a <J> key to select another book or another chapter.

Conclusion

In this document, I explained the actions that this interactive system can perform to accomplish the task – reading an auditory chapter of an audiobook and also have discussed what a programmer or user needs to know to modify the system. I also depicted a high-level discussion on the system flow, system states and also how the help messages and quit messages get changed as per the system states. In this pushbutton-input auditory-output system, all the outputs are played as audio.