Project #1
Explore interactive system design in a pushbutton-input auditory-output interface

Addendum (new ones at the top):

10-27-2019

The due date was initially written as "9AM, Wednesday, October 28, 2019". This was corrected to "Monday" as was specified in class when the project was handed out.

10-23-2019

1. Please download and use the latest "P1_Starter_Code_v3.zip", which is now available on Canvas. The directory structure and filenames in the "wav_files_provided" directory have changed. The hard-coded directory names and file names used for loading the provided sound files have changed in "ks_GLOBAL.py". No other provided python files should have changed.

2. The "Text file filename specification" structure for the text content files has been changed from
   NNN_PP_CC.wav
to
   NNN_CC_PP.wav
It is important that you change your code accordingly.

3. Please study the additional sound files that have been provided in the "wav_files_provided" directory. They may be useful to you.

4. A PDF of Chapter 2 of Norman (1988) has been provided in the file "Norman (1988) Design-of Chapter 2.pdf" so that you can see how the text is laid out visually.

Due dates:

- 9AM, Monday, October 21, 2019: A preliminary version of all deliverables must be submitted, with an expansion to the sample code such that the user can step through all the chapter titles of a book, select chapter 2, and start reading that chapter.
- 9AM, Monday, October 28, 2019: Complete project due.

Overview

In this project, you will design and build the interaction for a pushbutton-input and auditory-output interface for listening to an audio book. You can think of the user interface as the software that would be used for a device such as that shown in Figure 1, in which there is is an auditory display but no visual display, and in which there are five buttons for input, though your systems will use the five "home" keys on the keyboard (space, J, K, L, and ;).

Figure 1. An imaginary handheld eyes-free 5-button-input auditory-output device.
(Image adapted from http://www.crsltd.com Pyka 5 and http://www.aliexpress.com 842294332)

The point of the project

This project serves to illustrate the interactive nature of human-computer interfaces, specifically the need to consider task hierarchy, procedural knowledge, interaction and information design, and the constraints of human information processing, all of which interact to contribute to the usability of a human-computer interface.

A detailed task analysis is required.

The project includes converting a primarily visual and physical task to a primarily auditory and pushbutton task, which will require you to (a) break down the task of reading a book into subtasks at an appropriate level of granularity, (b) analyze what is the visual information that is needed when reading, and (c) figure out how to provide that information through pushbutton-input auditory-outputs.

Possible user tasks to support

Open a book, resume reading at a bookmark, "see" where you are in a book (such as how far through the book, what chapter, what page number), read the chapter headings, skim a book just reading the headings and topic sentences, put a bookmark where you are done reading, close the book and pick a different book to read.

The specifications of the program

You will create a user interface in which a person uses five keys on a computer's physical keyboard to navigate through auditory menus that permit the user to navigate through the book. Figure 2 illustrates how one such interaction might start.

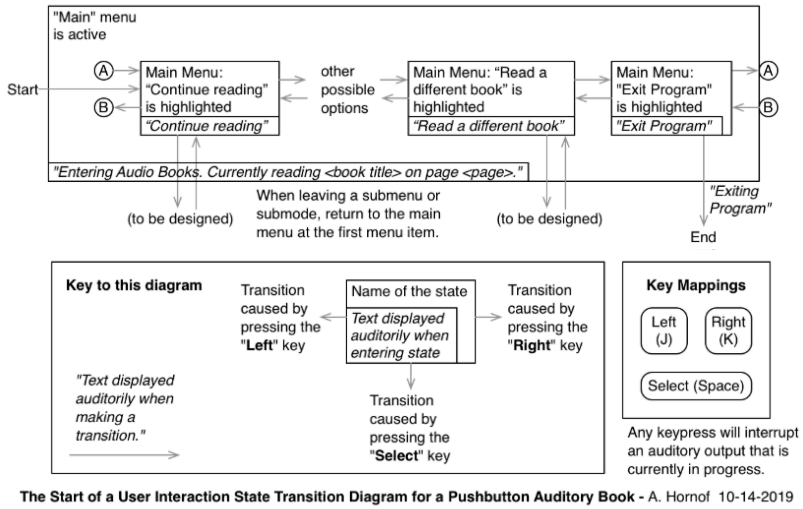

Figure 2. An excerpt from an interaction design for an auditory audiobook interface.

Some technical specifications of the program

The program must be written in Python 3, must use simpleaudio for sound, and must use the Python "readchar" package for single-keystroke input. No python libraries may be used except for multiprocessing, simpleaudio, readchar, and time.

Starter code

P1 starter code is available on Canvas to show how Python3, simpleaudio, and readchar can work together to create a pushbutton auditory menu. Seven source code files are available to you: ks_GLOBAL.py, ks_load.py, ks_log.py, ks_main.py, ks_o.py, ks_play.py, ks_stop.py. Only modify **ks_main.py** and **ks_o.py**. These are the only source code files that will be used from the code that you submit. All of the other files will be used from the starter code. Hence, you should keep the other five files exactly as they are, without modifying the code.

Text file filename specification

The wav file filenames are all nine characters long and encoded as follows:

NNN_PP_CC.wav

NNN is the sequence position. It is a sequence of integers such that if all of the soundfiles are played in order, from 000 to 634 (if there are 635 sound files), the chapter will have been played in its entirety, in exactly the order in which it was recorded.

PP is the page number of the printed book on which this sound file starts.

CC is a two-character string code describing the type of text, using the following codes:

CN for chapter name
SH for section heading
TS for topic sentence
RP for rest of paragraph

Sound files

The starter code has a directory structure filled with soundfiles that your program can use. Keep this directory structure, and the file-names, exactly as they are, and **do not resubmit the soundfiles with your projects**. You can create new soundfiles, but put them into a different directory, and submit that directory and those soundfiles.

If you create your own sound files

If you create your own sound files: Make sure that the sound quality is very good. That is, the sounds should be of a similar volume and clarity as the soundfiles provided. The voices should maintain a serious tone. And, as with the sound files provided to you, trim off the leading silence before the voice starts, and the trailing silence after the voice ends. You should be able to use Audacity or ocenaudio to normalize volumes, trim silence, and save as .wav files with file parameters, such as sample rate, that match the soundfiles provided. Try to get the the best sound that you can by trying different recording techniques and listening to and comparing the results using good-quality speakers or headphones. You may find, for example, that the microphone on your smartphone is superior to that on your laptop.

Coding requirements

Use only forward-slashes in the file pathnames in your code. Your code should be well-documented. Attribution should be provided at the top of each source code file for any source code that is copied or derived from other sources. For example, if you copy or derive some code from docs.python.org, you should write something like "# Code derived from docs.python.org" at the top of the source code file in which the copied or derived code appears.

No Additional Python Packages

Your program should not require any packages to be installed other than multiprocessing, simpleaudio, readchar, and time.

Documentation requirements

Along with your source code, you should submit:
(a) A complete description of the user interaction that is supported by the system. This should include all of the functionality provided by the system. All modes and states, and all transitions from one state to another, should be included in the documentation. The state transition diagram shown in Figure 2 would be a fine notation to use to describe the interactive component of your system's user interface, but note that Figure 2 would need to be expanded to include the additional functionality required. It should be easy and clear for a person reading this document to understand how the system works from the user's perspective. A hand-drawn and well-scanned (not sloppily photographed) diagram would be fine.
(b) Programmers documentation. This should include a high-level description of how your system works and any major or important data structures (such as classes and class hierarchies). This should be roughly 400 words in length and might include a diagram or two.

Working with Python in Room 100 Deschutes

- If you are going to work in Room 100 Deschutes, you need acquire a USB flash drive to use for the project, and take it with you every time you go to 100 Deschutes to work on the project. Work from that USB drive rather than from your CIS network drive. The USB drive will be 10 to 100 times faster. Remember to eject the USB flash drive when you are done with each work session, and be sure to back up your work that is on the USB drive onto some other machine. For example, small files might be copied onto your CIS account.
- You may need to use pyvenv in order to install and use the simpleaudio and readchar packages in Room 100. You might first check to see whether the packages have been installed by simply trying to run the sample code and looking for errors pertaining to these packages.

Installing simpleaudio and readchar

It should be straightforward for you to install simpleaudio and readchar using pip3. (Note that to install into Python3, you might need to use the command "pip3" rather than "pip".)

Programming Resources

There are many excellent Python resources that are available to you. Here are some general programming suggestions (PDF).

Consider using the PyCharm IDE

- I use the PyCharm Community Edition integrated development environment (IDE). I tried several IDEs and have found this one to be the one of the easiest to use. I find the interface for IDLE to be very difficult to use. Here are my ongoing Notes on Using Pycharm (PDF) which might be useful to you. These notes can serve as an example for the kinds of notes you should perhaps take as you are learning how to use a programming language or IDE. (I have a similar notes file for Python, for C++, for general computer usage, and for other topics.)
- If you try to do this assignment using PyCharm, note the "If you need a real TTY terminal within PyCharm" entry in the Notes on Using Pycharm (PDF).

Submit your project for grading on Canvas

You should create a single compressed .zip file that includes the Python files and soundfiles that you created, as well as your two documentation files in PDF format. Please do not submit the"sound.py" file or the "wav_files_provided" directory. Your compressed file should maintain the directory structure that your source code used to access the sound files. (Of course, please use relative references for all of your directory references.) Name the files as follows: Your main program file should be "ks_main.py" such that the program will run by unpacking your .tar or .zip file, cd-ing to that directory, and typing "python3 ks_main.py" at the command line. Your user interaction description should be "user_interaction.pdf". Your programmer's documentation should be "prog_doc.pdf".

Grading criteria

Does the system work? Does the system support all of the the tasks and subtasks that are needed to read a book? Is the system easy to use? Are the usability requirements satisfied? Are the description of the user interaction and the programmer's documentation both well-written, useful, and clear? If you created sound files of your own, are they of high quality? Is the source code well-written and well-commented? Is the source code written in a modular object-oriented manner that lends itself to easy and flexible redesign of the menus (such as by avoiding long sequences of nested loops)?