Software Assignment-1 Report

AI1110: Probability and Random Variables

Talasu Sowmith (EE22BTECH11218)

AIM : To make a music player application which shuffles a given playlist and plays it in a random manner.

Libraries Required : numpy, random, tkinter, pygame

Description :

1) The imports numpy is used for array manipulation and random is used for shuffling the playlist. Tkinter is used for creating GUI. Pygame for handling music playback.
2) Choose music folder() opens a file dialog to choose a music folder and returns the selected folder path. ”get music files(folder path)” retrieves the music files (ending with ”.mp3”) from the specified folder path.
3) Next song() plays the next song in the playlist and updates the current song label. Quit player is used to quit the song and exit the music player. Play next song() plays the next song in the shuffled playlist.
4) Firstly we are taking the location of the folder, from which we are extracting all the mp3 files and creating a playlist.
5) Next we are shuffling the whole playlist such that the acquired order is random and creating a GUI.
6) We are giving some user controlled buttons like ”Next”, ”quit” which directs the user to next song and quit the music player respectively.
7) The songs will keep on playing until the user selects ”Quit”.

```python
import numpy as np
import random
import soundfile as sf
from tkinter import Tk, filedialog
import tkinter as tk
import pygame
import os


os.environ['SDL_VIDEODRIVER'] = 'x11'

def next_song():
    play_next_song()
    update_current_song_label()

def get_music_files(folder_path):
    music_files = []
    for file in os.listdir(folder_path):
        if file.endswith(".mp3"):
            music_files.append(os.path.join(folder_path, file))
    return music_files


def quit_player():
    pygame.mixer.music.stop()
    window.quit()

def play_next_song():
    global song_index
    song_index = (song_index + 1) % len(playlist)
    pygame.mixer.music.load(playlist[song_index])
    pygame.mixer.music.play()

def choose_music_folder():
    Tk().withdraw()
    folder_path = filedialog.askdirectory()
    return folder_path

def shuffle_playlist():
    global playlist, shuffled_playlist, song_index
    shuffled_playlist = playlist.copy()
    np.random.shuffle(shuffled_playlist)
    song_index = 0
    pygame.mixer.music.load(shuffled_playlist[song_index])
    pygame.mixer.music.play()

def update_current_song_label():
    current_song_label.config(text=f"playing : {os.path.basename(playlist[current_song_index])}")


pygame.init()
```

Fig. 7. code

```python

def update_current_song_label():
    current_song_label.config(text=f"playing : {os.path.basename(playlist[current_song_index])}")


pygame.init()


music_folder = choose_music_folder()


playlist = get_music_files(music_folder)


current_song_index = 0
shuffle_playlist()


window = tk.Tk()
window.title("Song Playlist")
window.geometry("400x200")


current_song_label = tk.Label(window, text="Current Song: ")
current_song_label.pack()


quit_button = tk.Button(window, text="Quit", command=quit_player)
quit_button.pack()

next_song_button = tk.Button(window, text="Next ", command=next_song)
next_song_button.pack()


update_current_song_label()


pygame.mixer.init()
pygame.mixer.music.load(playlist[current_song_index])
pygame.mixer.music.play()


window.mainloop()


pygame.quit()
```

Fig. 7. code

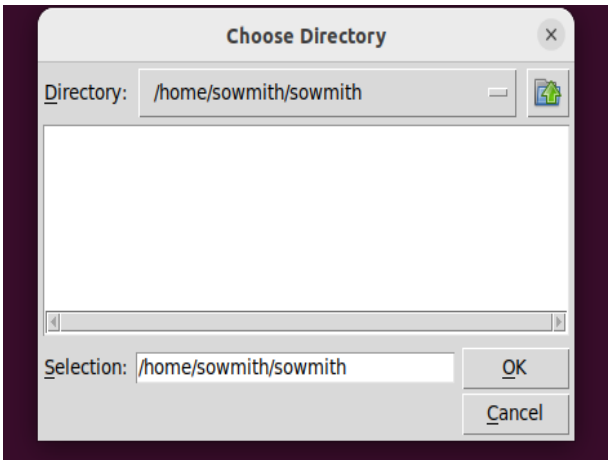

Fig. 7. this is how we select the folder of audio files

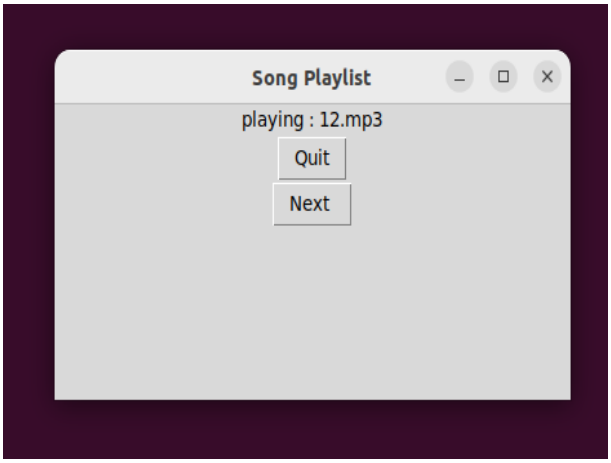

Fig. 7. these are the options provided to user