Music Player - Report

Anirudh Saikrishnan

May 18, 2023

1 Introduction

In this report, we will discuss a Python program that allows users to play songs and shuffle the playlist. The program utilizes the Pygame library for audio playback and does not rely on the random library for shuffling songs.

2 Program Overview

The Python program consists of the following key components:

- The main function: This function serves as the entry point of the program. It initializes the Pygame mixer, sets the volume, and handles the user interaction.
- The Shuffle class: This class extends the list class and provides a custom shuffling algorithm using the numpy library. It takes a seed value to generate random numbers for shuffling the playlist.
- The play_song function: This function is responsible for playing a song using the Pygame mixer. It also includes the functionality to pause and resume the song based on user input.

3 Usage

To use the program, follow these steps:

1. Ensure that you have the Pygame library installed.
2. Create a directory called "Songs" and place your MP3 files in it.
3. Run the Python program.
4. Enter "start" to start the program and begin playing songs.
5. Once the program is running, you can enter the following commands:
    - "pause": Pause the currently playing song.
    - "play": Resume playback of the paused song.
    - "next": Skip to the next song in the playlist.
    - "prev": Go back to the previous song in the playlist.
    - "shuffle": Shuffle the playlist and start playing from the beginning.
    - "quit": Exit the program.

4 Code Listing

Below is the complete Python code for the program:

```python
import os
import numpy as np
import pygame

pygame.mixer.init()
pygame.mixer.music.set_volume(1)

class custom_random:
    def __init__(self, seed):
        self.seed = seed

    def generate(self, max_value):
        self.seed = (self.seed * 1103515245 + 12345) & 0x7FFFFFFF
        return self.seed % max_value

class Shuffle(list):
    def __init__(self, seed):
        self.random_generator = custom_random(seed)
        super().__init__()

    def do(self, l):
        song_order = []
        song_list = []

        while len(song_order) != 20:
            choice = self.random_generator.generate(20)
            if choice not in song_order:
                song_order.append(choice)
                song_list.append(l[choice])

        return song_list

def main():
    song = 0
    start_flag = False
    pauflag = False
    nextflag = False
    prevflag = False

    l = [song for song in os.listdir('Songs') if song.endswith('.mp3')]
    print(l)
    seed = int(input("Enter a seed value: "))
    s = Shuffle(seed)
    l = s.do(l)

    while True:
        if not start_flag:
            user_choice = input("Enter 'start' to enter, 'quit' to exit: ")
            if user_choice == "start":
                start_flag = True
                play = True
                curr_song = l[song]
                pygame.mixer.music.load(os.path.join('Songs', curr_song))
                pygame.mixer.music.play()
            elif user_choice == "quit":
                pygame.quit()
                break
            else:
                print("Invalid choice")

        print("Enter 'pause', 'play', 'quit', 'next', 'prev', or 'shuffle'")
        user_choice = input()

        if user_choice == "next":
```

```
            if pauflag:
                song += 1
                song %= 20
                nextflag = True
                curr_song = l[song]
                pygame.mixer.music.load(os.path.join('Songs', curr_song))
                print("Current song: ", curr_song)
                continue

            pygame.mixer.music.stop()
            song += 1
            song %= 20
            curr_song = l[song]
            pygame.mixer.music.load(os.path.join('Songs', curr_song))
            pygame.mixer.music.play()
            print("Current song: ", curr_song)

        elif user_choice == "prev":
            if pauflag:
                song -= 1
                song %= 20
                nextflag = True
                curr_song = l[song]
                pygame.mixer.music.load(os.path.join('Songs', curr_song))
                print("Current song: ", curr_song)
                continue

            pygame.mixer.music.stop()
            song -= 1
            song %= 20
            curr_song = l[song]
            pygame.mixer.music.load(os.path.join('Songs', curr_song))
            pygame.mixer.music.play()
            print("Current song: ", curr_song)

        elif user_choice == "quit":
            pygame.quit()
            break

        elif user_choice == "pause":
            pygame.mixer.music.pause()
            pauflag = True

        elif user_choice == "play":
            if pauflag and (nextflag or prevflag):
                pauflag = False
                nextflag = False
                prevflag = False
                pygame.mixer.music.play()
                continue

            pygame.mixer.music.unpause()
            pauflag = False

        elif user_choice == "shuffle":
            pygame.mixer.music.stop()
            seed = int(input("Enter a new seed value: "))
            s = Shuffle(seed)
            l = s.do(l)
            play = True
            curr_song = l[song]
            pygame.mixer.music.load(os.path.join('Songs', curr_song))
            pygame.mixer.music.play()
            pauflag = False
            nextflag = False
            prevflag = False
            print("Current song:", curr_song)

```

```python
        else:
            print("Invalid input")

if __name__ == '__main__':
    main()
```

5 Conclusion

In this report, we have discussed a Python program that enables users to play songs with shuffle functionality. The program utilizes the Pygame library for audio playback and incorporates a custom shuffling algorithm without relying on the random library. With this program, users can enjoy playing their favorite songs in a shuffled order and control the playback using simple commands.

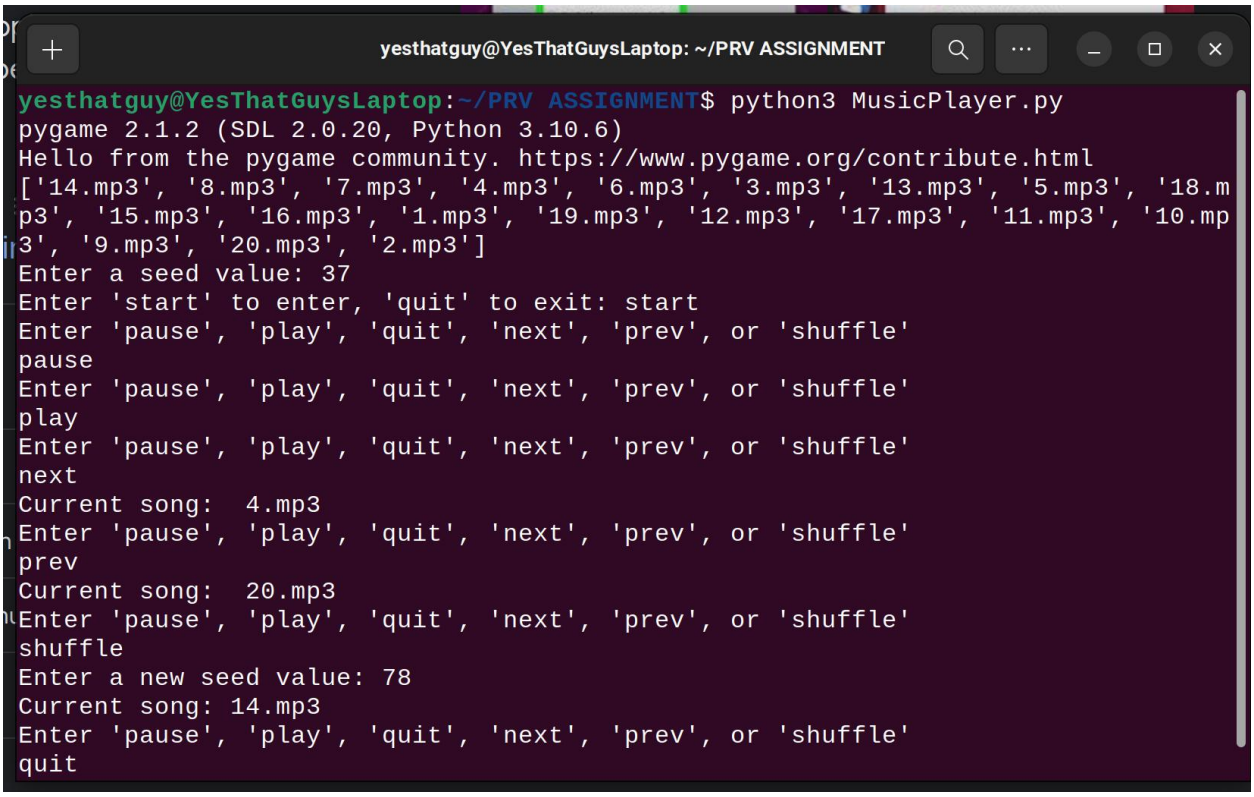

Figure 1: Music Player Functionality