

Randomized Music Player: A Python Project

Gholap Siddhesh Ashok
AI22BTECH11007

May 18, 2023

Abstract

This report presents a Python project that implements a randomized music player capable of shuffling and playing a given set of 20 songs. The project utilizes the pygame library for audio playback and incorporates user-friendly controls for song progression. The code ensures that all songs are played before repeating the process with a new random order, providing a dynamic and engaging music listening experience.

1 Introduction

The objective of this project is to develop a Python program that functions as a music player capable of randomizing and playing a collection of 20 songs. The project employs the pygame library for audio playback and incorporates essential technical aspects to ensure a smooth and reliable music player.

2 Methodology

The project follows a detailed methodology to achieve the desired functionality:

2.1 Library and Environment Setup

The pygame library, a popular Python module for multimedia applications, is used for audio playback. It must be installed and configured correctly to enable the program's functionality.

2.2 Song List Initialization

The Python code initializes a list of 20 songs, representing file paths to audio files. These songs should be stored in a format supported by the pygame.mixer.music module, which in this case was .mp3 format.

2.3 Shuffling Algorithm

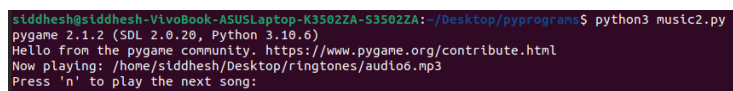
The `random.shuffle()` function from the `random` module is utilized to randomize the order of the songs. This ensures that each playback session presents a different sequence of songs. The algorithm assigns a random index to each song and rearranges them accordingly.

2.4 Audio Playback Control

The `pygame.mixer.music.load()` function loads each song into the mixer module, preparing it for playback. The `pygame.mixer.music.play()` function starts the playback of the current song.

2.5 User Interaction

During song playback, the program prompts the user for input, specifically the 'n' key. This input is captured using the `input()` function, allowing the user to control the progression to the next song. Upon receiving the 'n' input, `pygame.mixer.music.stop()` is called to stop the current song and proceed to the next one.

A terminal window with a dark background and light-colored text. The text shows the execution of a Python script named 'music2.py'. It displays the pygame version (2.1.2), SDL version (2.0.20), and Python version (3.10.6). It then prints a welcome message to the pygame community with a link to the website. The program then prints the path of the currently playing audio file and prompts the user to press 'n' to play the next song.

```
siddhesh@siddhesh-VlvoBook-ASUSLaptop-K3502ZA-53502ZA:~/Desktop/pyprograms$ python3 music2.py
pygame 2.1.2 (SDL 2.0.20, Python 3.10.6)
Hello from the pygame community. https://www.pygame.org/contribute.html
Now playing: /home/siddhesh/Desktop/ringtones/audio6.mp3
Press 'n' to play the next song:
```

Figure 1: asking `input(n)` for new song

2.6 Repeat Process

Once all the songs have been played, the program prompts the user to press Enter to play the songs again. This triggers the shuffling and playback process, generating a new random order for the songs.

3 Conclusion

The randomized music player project provides a simple Python program capable of shuffling and playing a set of given songs in a random order. The implementation utilizes the pygame library for audio playback and incorporates user interaction for controlling song progression. By ensuring that all songs are played before repeating the process with a new random order, the project offers a dynamic and engaging music listening experience. The program's technical considerations, such as library setup, shuffling algorithm, and audio playback control, contribute to its reliable and efficient performance.