Music Player Assignment Report

Bendi Rutwik Chandra

May 18, 2023

1 Introduction

In this report, I described the code for a simple music player application. The code is written in Python using the Pygame and Tkinter libraries. The music player allows users to play a playlist of songs, shuffle the playlist, skip to the next song, and stop the music.

2 Code Explanation

2.1 Libraries

The necessary libraries are imported at the beginning of the code:

```
import os
import random
import pygame.mixer
import tkinter as tk
```

2.2 Variables and Functions

The code defines several variables and functions that will be used later:

- `songs_directory`: Specifies the path to the directory containing the songs.
- `get_song_names(directory)`: Retrieves the names of the songs in the specified directory.
- `fisher_yates_shuffle(arr)`: Implements the Fisher-Yates shuffle algorithm to shuffle the array.
- `play_song(song)`: Plays the specified song using Pygame's mixer module.
- `skip_next()`: Stops the current song and plays the next song in the playlist.
- `stop_music()`: Stops the music and closes the application window.

- `update_song_label()`: Updates the GUI label with the name of the currently playing song.
- `update_gui()`: Monitors the music player's state and updates the GUI accordingly.

2.3 GUI Creation

The code creates a GUI window using the Tkinter library:

```
window = tk.Tk()
window.title("Music Player")
```

2.4 Playlist Initialization

The code initializes the playlist by retrieving the names of the songs in the specified directory and shuffling them:

```
songs = get_song_names(songs_directory)
playlist = songs.copy()
fisher_yates_shuffle(playlist)
playlist_index = tk.IntVar()
playlist_index.set(0)
```

2.5 GUI Components

The code creates various GUI components, including a label to display the currently playing song, and buttons for skipping to the next song and stopping the music:

```
song_label = tk.Label(window, text="Playing song: ")
song_label.pack()

next_button = tk.Button(window, text="Next", command=skip_next)
next_button.pack(side=tk.LEFT)

stop_button = tk.Button(window, text="Stop", command=stop_music)
stop_button.pack(side=tk.LEFT)
```

2.6 Updating the GUI

The code defines a function to update the GUI, which is called periodically using the Tkinter `after` method:

```
def update_gui():
    # Check if the current song has finished playing
    if not pygame.mixer.music.get_busy():
```

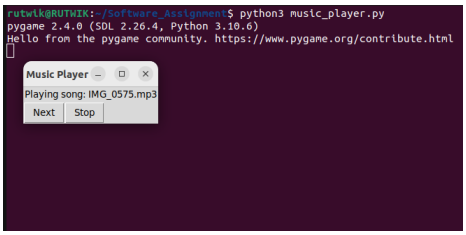

Figure 1: Image of GUI Window

```
        if len(played_songs) == len(playlist):
            played_songs.clear()  # Clear the played songs list
        playlist_index.set((playlist_index.get() + 1) % len(playlist))
        play_song(playlist[playlist_index.get()])
        update_song_label()
        played_songs.append(playlist[playlist_index.get()])
    window.after(100, update_gui)
```

2.7 Initialization and Event Loop

The code initializes the music player by playing the first song, updating the GUI label, and starting the GUI event loop:

```
played_songs = []
play_song(playlist[playlist_index.get()])
update_song_label()
window.after(100, update_gui)
window.mainloop()
```

3 Conclusion

In conclusion, the provided code implements a simple music player application using Python and the Pygame and Tkinter libraries. The code allows users to play a playlist of songs, shuffle the playlist, skip to the next song, and stop the music. By understanding the different components and their functionalities, one can further modify and enhance the music player application.