Software Assignment Report

Saketh Ram Kumar Dondapati

May 18, 2023

1 Introduction

This report presents an implementation of a simple music player using the Pygame library. The program allows users to play, pause, skip songs, and shuffle the song list.

2 Code Explanation

The code consists of several components, each responsible for a specific functionality. Let's go through them step by step:

2.1 Initializing Pygame and Audio Settings

The Pygame library is initialized, along with the audio settings. The width and height of the game window are set, and the volume is configured.

2.2 Loading Assets

The necessary images for buttons and backgrounds are loaded using the `pygame.image.load` function and stored in variables for later use.

2.3 Button Class

A `Button` class is defined to represent clickable buttons in the user interface. It has attributes for position, image, and a method for drawing the button on the screen.

2.4 Shuffle Function

The `shuffle` function takes a list of songs as input and shuffles them randomly using the `numpy.random.choice` function. It returns the shuffled song list.

2.5 Main Function

The `main` function is the entry point of the program. It initializes variables, sets up the GUI elements, and enters the main event loop. The event loop handles user input and updates the display accordingly.

2.6 Event Handling

Within the event loop, various events are handled, such as button clicks, song changes, and program termination. The user's mouse position is obtained using the `pygame.mouse.get_pos` function, and button clicks are detected by checking if the mouse position falls within the button's hitbox.

3 Code Listing

Below is the complete Python code for the program:

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

width, height = 750, 600

pygame.font.init()

# audio settings:
volume = 90
pygame.mixer.init()
pygame.mixer.music.set_volume(volume / 100)

WIN = pygame.display.set_mode((width, height))

# loading assets
BG = pygame.transform.scale(pygame.image.load(os.path.join('assets'
    , 'BG.png')), (width, height))
shuffle_image = pygame.transform.scale(pygame.image.load(os.path.
    join('assets', 'shuffle button.png')), (50, 50))
play_image = pygame.transform.scale(pygame.image.load(os.path.join(
    'assets', 'play.png')), (75, 75))
pause_image = pygame.transform.scale(pygame.image.load(os.path.join
    ('assets', 'pause.png')), (75, 75))
start_image = pygame.transform.scale(pygame.image.load(os.path.join
    ('assets', 'start.png')), (50, 50))
next_image = pygame.transform.scale(pygame.image.load(os.path.join(
    'assets', 'next.png')), (50, 50))
prev_image = pygame.transform.scale(pygame.image.load(os.path.join(
    'assets', 'previous.png')), (50, 50))


class Button:
    def __init__(self, x, y, image):
        self.x = x
        self.y = y
        self.image = image
        self.list = list

    def draw(self, window):
        window.blit(self.image, (self.x, self.y))

    def hitbox(self, mouseposition):
        if self.x < mouseposition[0] < self.x + self.image.get_width():
            if self.y < mouseposition[1] < self.y + self.image.get_height():
                return True


def shuffle(l):
    song_order = []
    song_list = []
    while len(song_order) != 20:
        choice = np.random.choice(20, 1) + 1  # pick a number from 1 to 20 with equal probability(1/20)
        if choice not in song_order:
            song_order.append(choice[0])  # adding the song to the list
            song_list.append(l[choice[0] - 1])
    print(song_order)
    print(song_list)
    return song_list


def main():
    global width, height, BG

    run = True
    fps = 60
    clock = pygame.time.Clock()
    song = 0
    paused = True

    l = [song for song in os.listdir('assets/songs') if song.endswith('.mp3')]
    print(l)

    curr_song = l[0]

    # fonts
    smallfont = pygame.font.Font(os.path.join('assets', 'LemonMilk.otf'), 20)

    playbutton = Button(275, 450, play_image)
    pausebutton = Button(400, 450, pause_image)
    prevbutton = Button(175, 462.5, prev_image)
    nextbutton = Button(525, 462.5, next_image)
    shufflebutton = Button(350, 300, shuffle_image)

    pygame.mixer.music.load(os.path.join('assets/songs', l[0]))
    pygame.mixer.music.play()
    pygame.mixer.music.pause()

    while run:

        clock.tick(fps)

        # Draw all the elements in the GUI
        WIN.blit(BG, (0, 0))
        playbutton.draw(WIN)
        pausebutton.draw(WIN)
        prevbutton.draw(WIN)
        nextbutton.draw(WIN)
        shufflebutton.draw(WIN)

        curr_song_label = smallfont.render(f'Currently␣playing:␣{curr_song}', 1, 'black')
        next_song_label = smallfont.render(f'Next␣in␣queue:␣{l[(song+1)%20]}', 1, 'black')

        WIN.blit(next_song_label, (375 - next_song_label.get_width() / 2, 100))
        WIN.blit(curr_song_label, (375 - curr_song_label.get_width() / 2, 50))

        # defining
        DONE = pygame.USEREVENT + 1
        pygame.mixer.music.set_endevent(DONE)
        # get mouse location
        mx, my = pygame.mouse.get_pos()
        for event in pygame.event.get():

            if event.type == DONE:
                song += 1
                song %= 20
                curr_song = l[song]
                pygame.mixer.music.load(os.path.join('assets/songs', curr_song))
                pygame.mixer.music.play()

            if event.type == pygame.QUIT:
                run = False
                pygame.quit()

            if event.type == pygame.MOUSEBUTTONDOWN:
                if nextbutton.hitbox((mx, my)):
                    if paused:
                        song += 1
                        song %= 20
                        curr_song = l[song]
                        pygame.mixer.music.load(os.path.join('assets/songs', curr_song))
                        pygame.mixer.music.play()
                        pygame.mixer.music.pause()
                    else:
                        pygame.mixer.music.pause()
                        song += 1
                        song %= 20
                        curr_song = l[song]
                        pygame.mixer.music.load(os.path.join('
```

```python
                            assets/songs', curr_song))
                        pygame.mixer.music.play()

                elif prevbutton.hitbox((mx, my)):
                    if paused:  # To simply load a song, but not play it when they player is in pause condition
                        song -= 1
                        song %= 20
                        curr_song = l[song]
                        pygame.mixer.music.load(os.path.join('assets/songs', curr_song))
                        pygame.mixer.music.play()
                        pygame.mixer.music.pause()
                    else:
                        pygame.mixer.music.pause()
                        song -= 1
                        song %= 20
                        curr_song = l[song]
                        pygame.mixer.music.load(os.path.join('assets/songs', curr_song))
                        pygame.mixer.music.play()

                elif pausebutton.hitbox((mx, my)):
                    pygame.mixer.music.pause()
                    paused = True

                elif playbutton.hitbox((mx, my)):
                    if paused:
                        paused = False
                        pygame.mixer.music.unpause()

                elif shufflebutton.hitbox((mx, my)):
                    pygame.mixer.music.stop()
                    l = shuffle(l)
                    curr_song = l[0]
                    pygame.mixer.music.load((os.path.join('assets/songs', curr_song)))
                    pygame.mixer.music.play()

            pygame.display.update()


main()
```

4 Conclusion

The implemented music player provides basic functionalities for playing, pausing, skipping songs, and shuffling the playlist. It serves as a starting point for further enhancements and customization.