Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

Домашнее задание по курсу

«Технологии мультимедиа»

Тема работы: «Создание плеера на Python для проигрывания звуковых файлов в формате MP3»

Выполнила: Попова Дарья, РТ5-61Б

Проверил: _____

12 апреля 2021 г.

ЗАЧТЕНО / НЕ ЗАЧТЕНО_____

(подпись)

Москва, 2021

Оглавление

Задание

Создать приложение, позволяющее выбирать из памяти компьютера аудиофайлы в формате .mp3, выводить их на экран (например, в виде списка-плейлиста) и проигрывать их. Также нужно предусмотреть следующий функционал:

- возможность ставить трек на паузу и снимать с паузы,
- останавливать трек,
- переключать на следующую или предыдущую композицию,
- выводить на экран общее время трека и время, прошедшее от начала проигрывания,
- менять текущее положение в треке с помощью ползунка,
- добавлять в список новые треки: по одному или несколько сразу,
- удалять треки: по одному или весь список целиком.

Средства реализации

Приложение реализовано на языке программирования Python с использованием модуля TKinter, позволяющего создавать графический пользовательский интерфейс, а также Pygame и Mutagen для работы с аудиофайлами.

Описание алгоритма

Если пользователь в меню выбрал «Добавить одну песню в плейлист», то необходимо дать ему возможность выбрать из файлового каталога один .mp3 файл, если на «Добавить несколько песен в плейлист», то допустить множественный выбор файлов.

Если пользователь в меню выбрал «Удалить одну песню из плейлиста», то удалить текущую выбранную песню, если на «Удалить все песни из плейлиста», то полностью очистить список композиций.

При выделении трека из списка и нажатии на кнопку со значком Play начать проигрывать выбранную композицию.

При нажатии на кнопку со значком «Пауза» приостановить проигрывание композиции, при повторном нажатии – возобновить.

При нажатии на кнопку со значком «Стоп» полностью остановить проигрывающуюся композицию, передвинуть ползунок в начало и сбросить счётчик секунд, прошедших с начала композиции.

При нажатии на кнопку со значком «Вперёд» начать проигрывать композицию, следующую за текущей в списке (причём если проигрывалась последняя композиция, то нужно перейти обратно в начало списка), передвинуть ползунок в начало и сбросить счётчик секунд, прошедших с начала композиции, а также начать новый отсчёт.

При нажатии на кнопку со значком «Назад» начать проигрывать композицию, находящуюся перед текущей в списке (причём если проигрывалась первая в списке композиция, то нужно перейти к последней), передвинуть ползунок в начало и сбросить счётчик секунд, прошедших с начала композиции, а также начать новый отсчёт.

При передвижении пользователем ползунка переключиться в соответствующую точку в композиции и обновить время от начала композиции, отображаемое в правом нижнем углу приложения.

Листинг программы

```
# импортируем модуль Pygame для работы с мультимедиа
# а также TKinter - стандартый графический пользовательский интерфейс для Питона
import pygame
from tkinter import *
import time
# импортируем модуль для сканирования аудио-файлов и определения их продолжительности
from mutagen.mp3 import MP3

# импортируем библиотеку Pillow для работы с изображениями формата .jpg и .png
from PIL import Image, ImageTk

# filedialog необходим для возможности выбора файла из файловой системы
```

```
компьютера
from tkinter import filedialog

# tkinter.ttk понадобится для работы со слайдерами
import tkinter.ttk as ttk

# создадим объект класса Tk
best_music_player = Tk()

# определим заголовок и размеры окна
best_music_player.title('Лучший в мире MP3-плеер')
best_music_player.configure(background='pink')
best_music_player.iconbitmap('c:/GUI/images/music-2-16.ico')
best_music_player.geometry("500x420")

# инициализируем миксер PyGame (предназначен для работы с аудиофайлами .mp3,
.wav и .ogg
pygame.mixer.init()


# определим функцию для добавления одной новой песни в плейлист
def add_song():
    song = filedialog.askopenfilename(initialdir='c:/GUI/audio',
                                      title='Выберите трек', filetypes=(("mp3
Files", "*.mp3"),))
    # уберём расширение файла и путь до файла из пользовательского вывода
    song = song.replace('C:/GUI/audio/', '')
    song = song.replace('.mp3', '')
    # добавим песню в listbox (плейлист)
    song_box.insert(END, song)


# определим функцию для добавления нескольких песен в плейлист
def add_many_songs():
    songs = filedialog.askopenfilenames(initialdir='c:/GUI/audio',
                                        title='Выберите трек',
filetypes=(("mp3 Files", "*.mp3"),))
    # пройдёмся по добавляемым песням и приведём их названия к удобному виду
    for song in songs:
        song = song.replace('C:/GUI/audio/', '')
        song = song.replace('.mp3', '')
        # и добавим песню в плейлист!
        song_box.insert(END, song)


# функция, удаляющая 1 выбранную песню
def remove_song():
    stop()
    song_box.delete(ANCHOR)
    pygame.mixer.music.stop()


# функция, удаляющая все песни из плейлиста
def remove_all_songs():
    stop()
    song_box.delete(0, END)
    pygame.mixer.music.stop()


# функция определяет, сколько времени прошло от момента начала проигрывания
песни
def play_time():
    if stopped:
```

```
            return
        # в целых секундах получим время от начала песни
        current_time = int(pygame.mixer.music.get_pos() / 1000)

        # с помощью модуля time переведём наши секунды в формат MM:CC
        converted_current_time = time.strftime('%M:%S',
time.gmtime(current_time))

        # определим текущую песню
        song = song_box.get(ACTIVE)
        song = f'C:/GUI/audio/{song}.mp3'

        global song_length

        # с помощью модуля  mutagen определим общую продолжительность песни
        song_mutagen = MP3(song)
        song_length = song_mutagen.info.length

        # конвертируем полученное в секундах время в формат MM:CC
        converted_total_time = time.strftime('%M:%S', time.gmtime(song_length))
        current_time += 1

        # каждый раз при вызове play_time() будем проверять, равно ли значение,
соответствующее ползунку, концу трека
        if int(my_slider.get()) == int(song_length):
            song_status_bar.config(
                text=f'Прошло:    {converted_total_time}    \n      Всего:
{converted_total_time}  ')
        # если песня приостановлена, то ползунок двигать не нужно
        elif paused:
            pass
        elif int(my_slider.get()) == current_time:
            # если значение ползунка равняется current_time, то пользователь не
сдвигал ползунок
            slider_position = int(song_length)
            my_slider.config(to=slider_position, value=current_time)
        else:
            # иначе ползунок сдвинут -> необходимо промотать трек до момента,
указанного ползунком
            slider_position = int(song_length)
            my_slider.config(to=slider_position, value=int(my_slider.get()))
            converted_current_time = time.strftime('%M:%S',
time.gmtime(int(my_slider.get())))
            song_status_bar.config(
                text=f'Прошло:    {converted_current_time}    \n      Всего:
{converted_total_time}  ')

            # удостоверимся,что статус обновляется каждую секунду (ф-ия
вызывается каждые 1000 мс)
            song_status_bar.after(1000, play_time)

            next_time = int(my_slider.get()) + 1
            my_slider.config(value=next_time)


# определим функцию для проигрывания выбранной песни
def play():
    global stopped
    stopped = False
    song = song_box.get(ACTIVE)
    song = f'C:/GUI/audio/{song}.mp3'
    pygame.mixer.music.load(song)
    pygame.mixer.music.play(loops=0)
```

```
    # вызовем ф-ию play_time и определим время, прошедшее с начала песни
    play_time()


global stopped
stopped = False


# функция останавливает проигрывание песни и убирает с неё выделение (выбор)
def stop():
    # переместим ползунок в начало
    my_slider.config(value=0)

    # очистим строку статуса с продолжительностью песни
    song_status_bar.config(text='')

    pygame.mixer.music.stop()
    song_box.select_clear(ACTIVE)

    # сделаем глобальный флаг
    global stopped
    stopped = True


# в глобальной переменной будет записано, поставлен ли трек на паузу или ещё
нет
global paused
paused = False


# функция даёт возможность поставить включённый трек на паузу
def pause(is_paused):
    global paused
    paused = is_paused

    if paused:
        # снять с паузы
        pygame.mixer.music.unpause()
        paused = False
    else:
        # поставить на паузу
        pygame.mixer.music.pause()
        paused = True


# функция включает следующую песню в плейлисте
def next_song():
    # определим номер (в составе кортежа) текущей выбранной песни
    next_one = song_box.curselection()
    # проверим, не является ли текущая песня последней в списке
    # если является, то переключимся на первую песню
    # если нет, то просто перейдём на следующий номер
    if next_one[0] == song_box.size() - 1:
        # придётся сделать список из кортежа (чтобы данные можно было
изменять)
        next_one = list(next_one)
        next_one[0] = 0
        next_one = tuple(next_one)
    else:
        next_one = next_one[0] + 1
    # по полученному номеру определяем песню из листбокса
    song = song_box.get(next_one)
```

```
    song = f'C:/GUI/audio/{song}.mp3'
    # переключим выбор-подсветку
    song_box.select_clear(0, END)
    song_box.activate(next_one)
    song_box.selection_set(next_one,last=None)
    # начнём проигрывание
    pygame.mixer.music.load(song)
    pygame.mixer.music.play(loops=0)

    # переместим ползунок в начало
    my_slider.config(value=0)

    # очистим строку статуса с продолжительностью песни
    song_status_bar.config(text='')

# функция включает предыдущую песню в плейлисте
def previous_song():
    previous_one = song_box.curselection()
    # проверим, не является ли текущая песня первой в списке
    # если является, то переключимся на последнюю песню из списка
    # если нет, то просто перейдём на предыдущий номер
    if previous_one[0] == 0:
        # придётся сделать список из кортежа (чтобы данные можно было
изменить)
        previous_one = list(previous_one)
        previous_one[0] = song_box.size() - 1
        previous_one = tuple(previous_one)
    else:
        previous_one = previous_one[0] - 1
    # по полученному номеру определяем песню из листбокса
    song = song_box.get(previous_one)
    song = f'C:/GUI/audio/{song}.mp3'
    # переключим выбор-подсветку
    song_box.select_clear(0, END)
    song_box.activate(previous_one)
    song_box.selection_set(previous_one, last=None)
    # начнём проигрывание
    pygame.mixer.music.load(song)
    pygame.mixer.music.play(loops=0)

    # переместим ползунок в начало
    my_slider.config(value=0)

    # очистим строку статуса с продолжительностью песни
    song_status_bar.config(text='')


# функция для ползунка
def slide(x):
    # slider_label.config(text=f'{int(my_slider.get())} of
{int(song_length)}')
    song = song_box.get(ACTIVE)
    song = f'C:/GUI/audio/{song}.mp3'
    pygame.mixer.music.load(song)
    pygame.mixer.music.play(loops=0, start=int(my_slider.get()))


# создадим список-плейлист
song_box = Listbox(best_music_player, bg='#DBF3FA', fg='#240935', width=60,
                   selectbackground='#cda4de', selectforeground='#1a153f')
song_box.pack(pady=20)

# определим картинки для кнопочек
```

```
play_btn_img = Image.open('c:/GUI/images/blue_play_button.jpeg.jpg')
resized = play_btn_img.resize((50, 50), Image.ANTIALIAS)
play_button_image = ImageTk.PhotoImage(resized)

stop_btn_img = Image.open('c:/GUI/images/pink_stop_button.jpeg.jpg')
resized = stop_btn_img.resize((50, 50), Image.ANTIALIAS)
stop_button_image = ImageTk.PhotoImage(resized)

pause_btn_img = Image.open('c:/GUI/images/purple_pause_button.jpg')
resized = pause_btn_img.resize((50, 50), Image.ANTIALIAS)
pause_button_image = ImageTk.PhotoImage(resized)

back_btn_img = Image.open('c:/GUI/images/green_back_button.jpg')
resized = back_btn_img.resize((50, 50), Image.ANTIALIAS)
back_button_image = ImageTk.PhotoImage(resized)

forward_btn_img = Image.open('c:/GUI/images/green_next_button.jpeg')
resized = forward_btn_img.resize((50, 50), Image.ANTIALIAS)
forward_button_image = ImageTk.PhotoImage(resized)

# сделаем рамочку
control_frame = Frame(best_music_player)
control_frame.pack()

# создадим кнопочки
back_button = Button(control_frame, image=back_button_image, borderwidth=0,
command=previous_song)
pause_button = Button(control_frame, image=pause_button_image, borderwidth=0,
command=lambda: pause(paused))
play_button = Button(control_frame, image=play_button_image, borderwidth=0,
command=play)
stop_button = Button(control_frame, image=stop_button_image, borderwidth=0,
command=stop)
forward_button = Button(control_frame, image=forward_button_image,
borderwidth=0, command=next_song)

# расположим кнопочки
back_button.grid(column=0, row=0)
pause_button.grid(column=1, row=0)
play_button.grid(column=2, row=0)
stop_button.grid(column=3, row=0)
forward_button.grid(column=4, row=0)

# создадим меню
my_menu = Menu(best_music_player)
best_music_player.config(menu=my_menu)

# в меню создадим кнопочку ("подменю") для добавления песен
add_song_menu = Menu(my_menu)
my_menu.add_cascade(label='Добавление треков', menu=add_song_menu)

# сделаем кнопочку для добавления одной песни
add_song_menu.add_command(label='Добавить одну песню в плейлист',
command=add_song)

# сделаем кнопочку для добавления нескольких песен
add_song_menu.add_command(label='Добавить несколько песен в плейлист',
command=add_many_songs)

# в меню создадим кнопочку ("подменю") для возможности удалять песни
remove_song_menu = Menu(my_menu)
my_menu.add_cascade(label='Удаление треков', menu=remove_song_menu)
```

```
# удаление одной песни
remove_song_menu.add_command(label='Удалить одну песню из плейлиста',
command=remove_song)

# удаление всех песен
remove_song_menu.add_command(label='Удалить все песни из плейлиста',
command=remove_all_songs)

# сделаем в правом нижнем углу штуку, которая будет нам показывать
продолжительность песни
# и сколько прошло от начала песни
song_status_bar = Label(best_music_player, text='', bd=1, relief=GROOVE,
anchor=E)
song_status_bar.pack(fill=X, side=BOTTOM, ipady=2)

# создадим ползунок
my_slider = ttk.Scale(best_music_player, cursor='heart', from_=0, to=100,
                      orient=HORIZONTAL, value=0, command=slide, length=360)
my_slider.pack(pady=40)

# создадим временную подпись для ползунка
# slider_label = Label(best_music_player, text='0')
# slider_label.pack(pady=10)

best_music_player.mainloop()
```

Результат выполнения