РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ГАЗА (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)ИМЕНИ И.М. ГУБКИНА

Кафедра информатики

Курсовая работа по дисциплине:

«Основы алгоритмизации и программирования»

на тему: **«Аудиоплеер»**

Выполнил: Ильичев Роман,
студент группы АС-22-05

Проверил: заведующий кафедрой информатики, к.т.н. Сидоров В.В.

Москва 2023

Оглавление

Введение

На сегодняшний день музыка - едва ли не самый популярный вид искусства. Она является важным компонентом в жизни многих людей. Сейчас музыка легкодоступная всем желающим. В настоящее время ни одно мероприятие, ни один праздник не проводятся без музыки, количество артистов растёт, проводится множество концертов. Выходя на улицу, можно заметить, что много тех, кто идёт в наушниках.

Но почему же музыка так популярна? Приведу несколько определений из интернета. Музыка – это один из видов искусства, который передает авторское настроение, ценности, мысли и чувства с помощью звуков. Или музыка – язык души, универсальный способ воздействовать на настроение, мышление и разум с помощью слухового восприятия. Можно сделать вывод, что музыка помогает нам «высказаться», справиться со своими духовными проблемами. Мелодия является способом оказать мощное воздействие на разум и настроение человека. Она поднимает дух. И так как музыка легкодоступна, то её популярность набирает обороты.

Я смело могу сказать, что музыка – это неотъемлемая часть моей жизни. Музыка – это хобби, которым я занимаюсь в свободное от учёбы время.

Выбирая тему курсовой работы, я прежде всего ориентировался на то, что я хочу, какая тема будет для меня интересной. Так как мне нравится музыка, то решил выбирать что-то связанное с этим видом искусства. Мне никогда не нравился десктопный аудиопроигрыватель на Windows. Именно поэтому мне в голову пришла мысль написать свой аудиопроигрыватель на языке Python, который будет простым в использовании, но полезным для меня.

Цели и задачи работы

Цель работы: написать на языке программирования Python аудиоплеер, который будет содержать в себе минимальные функции стандартного десктопного музыкального проигрывателя

Задачи работы:

1. Найти материал о создании музыкального плеера на языке программирования Python.
2. Отобрать материал, который является наиболее полезным для реализации моей цели.
3. Изучить материал о библиотеках языка программирования Python, которые будут задействованы в написании программы (GUI-библиотека tkinter, pygame, time, os, mutagen, TinyTag, pathlib) и их функциях.
4. Написать программу, реализующую основной функционал музыкального плеера.
5. Доработать интерфейс музыкального плеера.

Основная часть

Задействованные библиотеки

При просмотре многих материалов о создании музыкального плеера на просторах интернета я увидел, что графический интерфейс создаётся везде с помощью разных GUI-библиотек. В основном это конечно PyQt5 и Tkinter. Графический интерфейс в приложении — это, по моему мнению, чуть ли не главная составляющая. Несомненно, PyQt5 имеет намного обширней функционал, чем Tkinter. Но из-за загруженности в учёбе и сложности освоения PyQt5 я решил использовать Tkinter. Тем более, что я знаком с этой библиотекой.

Pygame – это важная библиотека в моём коде. Несмотря на то, что я использую всего около 10 функций этого модуля, эта библиотека «управляет» музыкой (загрузка музыки, воспроизведение, постановка на паузу, установление громкости, отслеживание события окончания проигрывания музыкального трека и др.)

Также я нашел некоторый минус библиотеки pygame. Мне требовалось найти длину трека для того, чтобы правильно согласовать процессы моего окна. Но pygame это делал долго - 4 секунды. Я нашел библиотеку mutagen. Она позволила мне узнать длину трека, причём сделать это очень быстро.

Модуль os предоставляет множество функций для работы с операционной системой, а самое главное с директориями. С помощью библиотеки os я создал системный файловый браузер.

Библиотека TinyTag помогает извлекать информацию о музыкальном файле. С помощью этого модуля я получаю название песни и артиста, исполняющего её.

Описание программы

Я решил писать программу, создав свой класс музыкального аудиоплеера.

```
1 usage
class MusicPlayer:
    def __init__(self):
        pygame.init()

        # add color
        self.color1 = '#E2CEE4'
        self.color2 = '#68456B'
        self.color3 = '#DFB3E4'
        self.color4= '#2D0731'
        self.color5 = '#413342'

```

Рисунок 1. Создание класса «MusicPlayer»

В функции __init__ (функция-конструктор, которая вызывается при создании объекта класса) я указываю основные цвета интерфейса, важные свойства и характеристики музыкального плеера, создаю окно, размещаю основные виджеты, привязываю сигналы виджетов к приёмникам сигналов (функции и методы класса), добавляю отслеживание события окончания музыки в pygame и, наконец, запускаю цикл событий приложения.

```
self.CurrentVolume = 0
self.CurrentSong = ""
self.CurrentTimeSong = tk.DoubleVar()
self.CurrentVolume = 70
self.CurrentNameSong = tk.StringVar()
self.CurrentArtistSong = tk.StringVar()
self.IntervalLineTime = 100
self.WidthButton = 45
self.HeightButton = 45
self.HeightFrameLineTime = 0.12 * self.HeightRoot
self.CountIterationAfter = -1
self.PlayingBool = False
self.ActiveMenu = False
self.ActiveBrowser = False
```

Рисунок 2. Важные свойства и характеристики музыкального плеера.

Дальше идут функции, которые «управляют» процессом. Сначала это

функции, которые вызываются для воспроизведения трека, остановки и снятия с паузы. Потом следуют пять слотов (приёмников сигналов), привязанных к кнопкам «Назад», «Вперёд», «Остановка/Воспроизведение», к событию «клик на песню в списке треков» и событию «окончание воспроизведения трека». Стоит отметить, что для контролирования процесса я использую много переменных типа bool. Например, переменная PlayingBool принимает истинное значение, если песня в данный момент играет, иначе ложное. Важная переменная Current Song типа str говорит о названии текущего аудиофайла в папке. Можно сказать, что принцип работы аудиоплеера – это использование глобальных перемен и привязка сигналов виджетов, к приёмникам сигналов (функциям).

```
def TurnOnMusic(self):
    self.LineTime.stop()
    self.CurrentTimeSong.set(value=0)
    pygame.mixer.music.unload()
    pygame.mixer.music.load(f'{self.Way}/{self.CurrentSong}')
    pygame.mixer.music.play()
    self.PlayingBool = True
    f = mutagen.File(f'{self.Way}/{self.CurrentSong}')
    self.LengthCurrentSong = f.info.length
    self.IntervalLineTime = int(self.LengthCurrentSong)
    self.LineTime.start(self.IntervalLineTime)
    self.CurrentSongInfo()
    if self.ActiveListSongs is True:
        self.ListSongs.focus_set()
```

Рисунок 3. Функция «TurnOnMusic» для воспроизведения трека

Потом идут функции, связанные с работой системного файлового браузера. Создаю файловый браузер я с помощью виджета ttk.Treview и словаря, ключи которого это id строк дерева, а значения – абсолютный путь до папки или файла. Затем можно увидеть функции, принимающие сигнал от линии времени, которая отображает текущее место песни, и линии громкости.

```
def ReleaseClickLineTime(self, event):
    if self.PlayingBool is True and self.CurrentSong != "":
        self.CurrentTimeSong.set(value=((self.Root.winfo_pointerx() - self.Root.winfo_rootx()) / self.WidthRoot) * 1000)
        pygame.mixer.music.play(loops=1, start=(self.CurrentTimeSong.get() / 1000) * self.LengthCurrentSong)
        self.LineTime.start(self.IntervalLineTime)
    elif self.CurrentSong != "":
        self.CurrentTimeSong.set(value=((self.Root.winfo_pointerx() - self.Root.winfo_rootx()) / self.WidthRoot) * 1000)
        pygame.mixer.music.play(loops=1, start=(self.CurrentTimeSong.get() / 1000) * self.LengthCurrentSong)
        pygame.mixer.music.pause()
```

Рисунок 4. Функция, вызывающаяся после клика по линии времени

Далее в коде прописаны функции, которые отвечают за поиск песни в текущей директории.

```
def SearchingProcess(self, event):
    if self.EntrySearching.get() != "Поиск песни":
        SearchingSong = self.EntrySearching.get()
        if len(SearchingSong) >= 1:
            if len(SearchingSong) >= self.LengthSearchingSong and len(self.AllSongsListSongs) != 0:
                self.LengthSearchingSong = len(SearchingSong)
                for song in self.AllSongsListSongs:
                    if SearchingSong not in song:
                        self.DeletedSongListSongs.append(song)
                        self.ListSongs.delete(self.ListSongs.get(0, 'end').index(song))
                self.AllSongsListSongs = []
                for song in self.ListSongs.get(0, 'end'):
                    self.AllSongsListSongs.append(song)

                if self.CurrentSong not in self.ListSongs.get(0, 'end'):
                    self.ActiveListSongs = False
```

Рисунок 5. Отрывок функции «SearchingProcess», отвечающей за процесс поиска

И наконец, пробегая по коду, можно увидеть функции, написанные для остальных кнопок и функцию, которая принимает сигнал при изменении конфигурации окна.

```python
def ClickVolumeButton(self, event):
    if self.ActiveNotVolume is False:
        self.VolumeButton.delete('all')
        self.VolumeButton.create_image(22.5, 22.5, anchor='center', image=self.PhotoNotVolume)
        self.LabelImageVolume.config(image=self.PhotoNotVolume)
        self.CurrentVolume = self.LineVolume['value']
        self.LineVolume['value'] = 0
        pygame.mixer.music.set_volume(0)
        self.ActiveNotVolume = True
    else:
        self.VolumeButton.delete('all')
        self.VolumeButton.create_image(22.5, 22.5, anchor='center', image=self.PhotoVolume)
        self.LabelImageVolume.config(image=self.PhotoNotVolume)
        self.LineVolume['value'] = self.CurrentVolume
        pygame.mixer.music.set_volume(self.CurrentVolume/100)
        self.LabelImageVolume.config(image=self.PhotoVolume)
        self.ActiveNotVolume = False
```

Рисунок 6. Функция, принимающая сигнал "клик по кнопке громкости".

Дополнительные утилиты

В создании аудиоплеера мне помогали ещё два приложения: ColorMania, Inkscape.

Рисунок 7. ColorMania

ColorMania – это программа, которая предоставляет пользователям упрощенный доступ к практически неограниченному количеству цветов, которые можно встроить в конкретный дизайн.

Рисунок 8. Inkscape

Inkscape – векторный редактор, позволяющий преобразовывать векторную графику в растровую и наоборот. Это приложение я использовал для рисования значков.

Заключение

Таким образом, мне удалось реализовать аудиоплеер на языке программирования Python. В ходе работы мною были приобретены новые знания в области разработки графического интерфейса с помощью GUI-библиотеки Tkinter, а также я открыл для себя библиотеки, позволяющие работать с аудиофайлами.

Хочу отметить, что мой аудиоплеер обладает минимальным набором тех функций, которые обеспечивают нормальную работу музыкального проигрывателя. В будущем этот аудиоплеер можно дорабатывать, добавляя новые опции в приложение. Например, можно в меню проигрывателя добавить настройку текущей темы приложения. А также впоследствии можно оптимизировать код этого плеера.

Список использованных литературных источников

1. https://www.youtube.com/watch?v=CnKk_67ytCA&t=3226s
2. https://docs.python.org/3/library/tkinter.ttk.html
3. https://pyga.me/docs/ref/music.html
4. https://lavrynenko.com/python-mutagen-primer-raboty
5. https://www.opensourceagenda.com/projects/tinytag
6. https://pythonworld.ru/moduli/modul-os.html?ysclid=lhdvnoshbp465527276