Негосударственное учреждение дополнительного образования

**«НОВОСИБИРСКАЯ АКАДЕМИЯ ДИЗАЙНА И ПРОГРАММИРОВАНИЯ»**

**Тема «Создание аудиоплеера с графическим интерфейсом на языке Python»**

Пояснительная записка к дипломному проекту

|  |  |  |
| --- | --- | --- |
| Студент |  | Полещук Евгений Алексеевич |
|  |  |  |
| Руководитель дипломного проекта |  | Литвина Софья Сергеевна |
|  |  |  |

2017

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 3

1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ 5

1.1 Технические требования 5

1.2 Основания для разработки 5

1.3 Назначения для разработки 5

2. ТЕОРИТИЧЕСКОЕ ОПИСАНИЕ 6

2.1 Что такое Python 6

2.2 Преимущества языка 6

2.3 О модулях «Tkinter» и «Pygame» 7

3. ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ 8

3.1 Подготовка к написанию программы 8

3.2 Написание основных функций программы 9

3.3 Программирование интерфейса аудиоплеера 12

3.4 Общий вид программы 15

ВЫВОД 16

СПИСОК ЛИТЕРАТУРЫ 17

ПРИЛОЖЕНИЕ - 1 18

ПРИЛОЖЕНИЕ - 2 диск

**ВВЕДЕНИЕ**

Я решил создать этот проект, так как хотел научится программированию программного обеспечения и его последующему оформлению. Также я рассчитываю, что этот проект поможет мне получить опыт в использовании различных особенностей языка, а также позволит попрактиковаться в написании функций, позволяющих работать с директориями и аудио файлами. Также я получу общее представление о том, как строятся программы такого рода, научусь программировать интерфейс. Язык Python был выбран в качестве используемого мной, из-за лёгкости в изучении, простым синтаксисом, и из-за своей универсальности. Python — активно развивающийся язык программирования, новые версии (с добавлением/изменением языковых свойств) выходят примерно раз в два с половиной года. Я считаю, что он подходит для работы на нём как начинающих программистов, так и для опытных программистов. Именно поэтому, мой проект сделан на нём. Из числа преимуществ этого языка, можно выделить:

* Скорость выполнения программ, написанных на Python очень высока. Это связанно с тем, что основные библиотеки Python   
  написаны на C++ и выполнение задач занимает меньше времени, чем на других языках высокого уровня.
* В связи с этим вы можете писать свои собственные модули для Python на C или C++
* В стандартных библиотеках Python вы можете найти средства для работы с электронной почтой, протоколами Интернета, FTP, HTTP, базами данных, и пр.
* Скрипты, написанные при помощи Python выполняются на большинстве современных ОС. Такая переносимость обеспечивает Python применение в самых различных областях.
* Python подходит для любых решений в области программирования,

будь то офисные программы, вэб-приложения, GUI-приложения и т.д.

* Над разработкой Python трудились тысячи энтузиастов со всего мира. Поддержкой современных технологий в стандартных библиотеках мы можем быть обязаны именно тому, что Python был открыт для всех желающих.

Учитывая это, можно смело сказать, что Python абсолютно подходит для написания такого рода проекта.

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

* 1. **Технические требования**

Минимальные требования для функционирования продукта:

* Компьютер;
* Установленный Python 3.x.x и его интерпретатор;
* Графическая операционная система;
* Предустановленная аудиосистема.
  1. **Основание для разработки**

Основания для проведения разработки: приказ о выполнении дипломного проекта от 31 мая 2017 года.

Исполнитель: студент Полещук Евгений Алексеевич.

* 1. **Назначения разработки**

Научиться разработке графического интерфейса.

Отточить навыки работы с аудио файлами.

Изучить системные библиотеки и библиотеку pygame.

**ТЕОРЕТИЧЕСКОЕ ОПИСАНИЕ**

* 1. **Что такое Python**

Интерпретируемый объектно-ориентированный язык программирования высокого уровня с динамической типизацией, автоматическим управлением памятью и удобными высокоуровневыми структурами данных, такими как словари (хэш-таблицы), списки, кортежи. Поддерживает классы, модули (которые могут быть объединены в пакеты), обработку исключений, а также многопоточные вычисления. Питон обладает простым и выразительным синтаксисом. Язык поддерживает несколько парадигм программирования: структурное, объектно-ориентированное, функциональное и аспектно-ориентированное.

**2.2 Философия языка**

Разработчики языка Python придерживаются определённой философии программирования, называемой «TheZenofPython» («Дзен Питона», или «Дзен Пайтона»). Её текст выдаётся интерпретатором Python по команде import this (работает один раз за сессию). Автором этой философии считается Тим Петерс (TimPeters).

**2.3 О модулях Tkinter и Pygame**

**Tkinter** – это кроссплатформенная библиотека для разработки графического интерфейса на языке Python (**начиная с Python 3.0 переименована в tkinter**). Tkinter расшифровывается как Tk interface, и является интерфейсом к Tcl/Tk. Tkinter входит в стандартный дистрибутив Python. Ткинтер является широко используемым модулем для создания интерфейса в Python. Благодаря этому, наличие обильного количество документации позволяет использовать все его преимущества по максимуму, но без особых сложностей.

**Pygame** — набор модулей (библиотек) языка программирования Python, предназначенный для написания компьютерных игр и мультимедиа-приложений. Pygame базируется на мультимедийной библиотеке SDL.

Изначально Pygame был написан Питом Шиннерсом (Pete Shinners). Начиная, примерно с 2004/2005 года поддерживается и развивается сообществом свободного программного обеспечения.

**ТЕХНОЛОГИЧЕСКИЙ РАЗДЕЛ**

**3.1. Подготовка к написанию программы**

Открываем среду разработки IDLE. Создаём новый файл путём нажатия на клавиатуре сочетания клавиш «Ctrl+N» либо нажатием на «New File» во вкладке «File»(Рисунок 1).

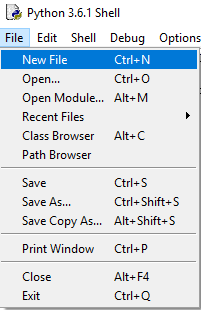


Рисунок 1

Для удобства,файл можно сразу сохранить в выбранную папку «Ctrl+Shift+S» на клавиатуре.

Теперь импортируем необходимые нам модули и инициазизируем миксер

путем написания следующих команд (Рисунок 2).

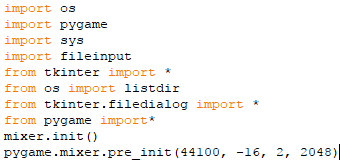


Рисунок 2

Здесь мы импортируем модули Tkinter,Pygame,Fileinput и os, которые будем задействовать позднеее в программе.

**3.2 Написание основных функций программы**

В этой части программы мы будем прописывать все необходимые функции для её функционирования.

Сначала задаем функции воспроизведения и остановки аудиофайла. (Рисунок 3)

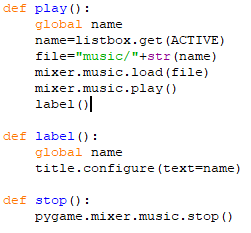
****

Рисунок 3

Функции задаются с помощью команды def «название функции»(\_):

Внутри я прописал, откуда функция должна считывать необходимый аудиофайл, и как его правильно читать.

В данном случае используется listbox(список),его мы добавим немного позже,когда будем программировать интерфейс. В конце функции пропишем вызов ещё одной функции «label», она ответчает за вывод на экран названия воспроизводимого трека. def label использует глобальную переменную name, которую мы задали в def play, и конфигурирует свой виджет(Label) вставляя в него название трека, которое хранится в name. Сам виджет мы также добавим после написания всех основных функций.

Далее пишем несколько ещё несколько важных функций.

Включение следующего трека в плейлисте,импорт аудиофайла из любой директории,и автоматическое считывание аудофайлов из папки «music»(Рисунок 4)

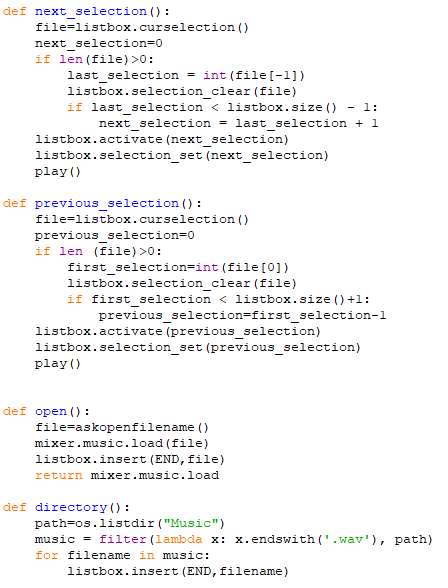


Рисунок 4

Фунцкии next\_selection и previous\_selection отвечают за включение следующего и предыдущего трека соответственно, по нажатию кнопки,их мы также запрограмируем позднее. def open позволяет нам выбирать директорию на компьюетере и таким образом импортировать любой трек в наш список. Это мы делаем при помощи модуля tkinter.filedialog, который как раз таки позволяет рабоать с файлами на компютере. askopenfilename(\_) открывает дополнительное окно, в котором мы выбираем нужный нам трек.mixer.music.load сразу загружает выбранный аудиофайл в миксер, тоесть загруженный файл будет проигрыватся автоматически при запуске фунции play. Здесь также указан listbox, .insertозначает что после выбора аудиофайла, трек должен быть помещён в наш список, для последующего его проигрывания. returnнеобходим для правильной работы mixer.music.load, без него, другие функции не смогли бы определить что трек загружен, так как переменная в функции осталась бы локальной.

defdirectory() позволяет нам считывать файлы из заданной папки в самом конце кода, мы пропишем open(\_) что бы функция запускалась автоматически, при запуске программы. Переменная pathьудет хранить в себе директорию, которая будет открываться при запуске функции. Я сделал это при помощи команды listdirмодуля os, который был импортирован ранее. В переменной musicмы фильтруем файлы, для того что бы в случае если в папке с музыкой находились бы другие файлы(например папки или абложки к альбомам) , программа не выводила их в наш список. Вывод всех необходимых файлов осуществляется цикл, с помощью уже знакомой нам команды insert. В скобочках я прописал, куда нужно помещать каждый последующий файл, в данном случае в конец списка(END). Также указал что конкретно должно выводится в listbox, а именно имя файла (filename).

**3.3 Программирование интерфейса при помощи библиотеки Tkinter**

После того, как основные функции готовы к использованию, следует привязать их непосредственно к интерфейсу нашего аудиоплеера. Здесь нам и потребуется библиотека «Tkinter» которую мы импортировали ранее. Для начала пропишем несколько простых строк(Рисунок 5).

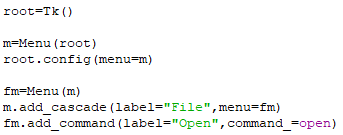


Рисунок 5

Здесь мы создаём главное окно программы: root=Tk(\_). Это стандартное начало для любого GUIнаписанного на «Tkinter». Затем мы создаём меню которое будет располагаться на верхней панели (m = Menu), в скобках прописываем root, тем самым указывая где будет располагаться меню. Для того что бы меню отображалось, конфигурируем главное окно указывая в скобках главное меню (то есть переменную m), которое и будет распологаться сверху. На него помещаем каскад с названием File(m.add\_cascade) в скобках указываем куда его поместить, а также текст на нём. Затем прописываем что будет появляться в каскаде, и какую функцию будет выполнять этот Label(fm.add\_command).



После того, как все приготовления завершены, можно приступать к написанию основного интерфейса. Он будет состоять из списка треков, нескольких кнопок, шкалы громкости, и меню (которое мы уже создали).

Сначала поместим несколько виджетов - кнопок для удобного использования наших функций (Рисунок 6).

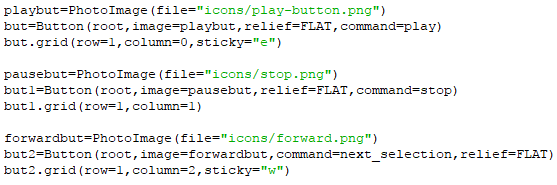


Рисунок 6

В этом нам поможет виджет Button. В скобках прописываем где он будет располагаться, какая иконка будет отображаться на месте виджета (их мы подгружаем заранее при помощи команды PhotoImage, внутри неё указываем путь до иконки. В данном случае изображение лежит в директории ниже, в папке icons.). В конце указываем функцию, которую он будет запускать(command).

Сами виджеты мы размещаем с помощью grid.

Этот упаковщик представляет собой таблицу с ячейками, в которые помещаются виджеты. В скобках мы указываем в какую строку(row) и столбец(column) мы хотим разместить наш виджет. Также можно указать сторону, к которой будет «прилипать» (sticky) виджет. (Рисунок 7)



Рисунок 7

Теперь, мы должны поместить в нашу программу список, в котором будут отображаться загруженные аудио файлы, и из которого мы сможем выбирать любой из них. Делается это при помощи виджета «Listbox» (Рисунок 8).



Рисунок 8

В параметрах Listbox’а мы указываем высоту окна, ширину, и количество файлов которые можем выбрать, в данном случае SINGLE.

Далее создадим шкалу громкости. Для этого используем виджет scale, в скобках укажем направление (orient,в данном случае для громкости больше подходит горизонтальное) , точки максимума и минимума и минимальное расстояние на которое можно будет двигать шкалу.showvalue установим на FALSE, этот параметр отвечает за отображение значения шкалы, но в нашем случае эту функцию будет выполнять отдельный Label. (Рисунок 9)



Рисунок 9

Сама по себе шкала ничего не делает, поэтому далее пропишем функцию, с которой она будет взаимодействовать (Рисунок 10).

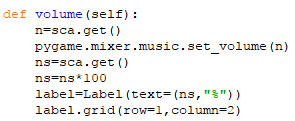


Рисунок 10

Эта функция считывает значение шкалы через переменную n, с помощью .get(\_), и изменяет громкость воспроизведения аудио файла в соответствии с ним(set\_volumeв миксере). Также, над самой шкалой добавляем «Label». Этот виджет выводит некий текст, мы же поместим его прямо в функцию, затем что бы он отображал текущую громкость: 

**3.4 Структура и общий вид программы**

Таким образом, наша программа разбита на три основных части:

1. Импорт необходимых для функционирования программы модулей.
2. Функций для различного рода манипулирования аудио файлами.
3. Интерфейс для удобной работы с созданными выше функциями.

Все это мы можем наблюдать на Рисунке 12.

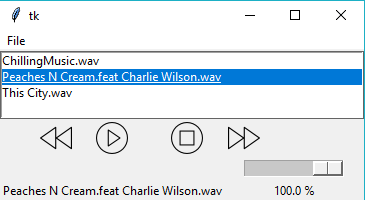


Рисунок 12

**Вывод**

Проведение исследования в этой области дало мне массу комплексных знаний, которые я буду использовать в своей дальнейшей работе. Благодаря этой работе у меня было много практики в данном направлении, а именно:

* Работа с аудио файлами.
* Работа с графическим интерфейсом.
* Работа с несколькими основополагающими библиотеками написанных для языка Python.
* Цельное понимание того, как примерно устроены программы такого типа.

**5.1 Список Литературы и источников**

1. Pygame,Документация по Pygame,2017,https://www.pygame.org/docs/
2. Python,Документация по Tkinter, https://docs.python.org/3/library/tk.html
3. Flaticon, Иконки для программ, 2017, http://www.flaticon.com/free-icons/music-player\_150435
4. The New Boston, Видеотуториалы по Python, https://thenewboston.com/videos.php?cat=99&video=20149
5. Лаборатория юного линуксоида,Учебник по Tkinter, http://younglinux.info/tkinter.php
6. Stackoverflow,Форум для программистов, 2017, https://stackoverflow.com/questions/43678493/pygame-is-not-responding-after-playing-an-audio-file

**Приложение**

import os

import pygame

import sys

import fileinput

from tkinter import \*

from os import listdir

from tkinter.filedialog import \*

from pygame import\*

mixer.init()

pygame.mixer.pre\_init(44100, -16, 2, 2048)

def play():

name=listbox.get(ACTIVE)

file="music/"+str(name)

mixer.music.load(file)

mixer.music.play()

def stop():

pygame.mixer.music.stop()

def next\_selection():

file=listbox.curselection()

next\_selection=0

if len(file)>0:

last\_selection = int(file[-1])

listbox.selection\_clear(file)

if last\_selection < listbox.size() - 1:

next\_selection = last\_selection + 1

listbox.activate(next\_selection)

listbox.selection\_set(next\_selection)

play()

def open():

file=askopenfilename()

mixer.music.load(file)

listbox.insert(END,file)

return mixer.music.load

def directory():

path=os.listdir("Music")

music = filter(lambda x: x.endswith('.wav'), path)

for filename in music:

listbox.insert(END,filename)

def volume(self):

n=sca.get()

pygame.mixer.music.set\_volume(n)

ns=sca.get()

ns=ns\*100

label=Label(text=(ns,"%"))

label.grid(row=1,column=2)

root=Tk()

m=Menu(root)

root.config(menu=m)

fm=Menu(m)

m.add\_cascade(label="File",menu=fm)

fm.add\_command(label="Open",command\_=open)

playbut=PhotoImage(file="icons/play-button.png")

but=Button(root,image=playbut,relief=FLAT,command=play)

but.grid(row=1,column=0,sticky="e")

pausebut=PhotoImage(file="icons/stop.png")

but1=Button(root,image=pausebut,relief=FLAT,command=stop)

but1.grid(row=1,column=1)

forwardbut=PhotoImage(file="icons/forward.png")

but2=Button(root,image=forwardbut,command=next\_selection,relief=FLAT)

but2.grid(row=1,column=2,sticky="w")

sca=Scale(root,orient=HORIZONTAL,length=100,from\_=0,to=1,resolution=0.1, command=volume, showvalue=FALSE)

sca.grid(row=2,column=2,sticky="e")

sca.set(1)

listbox=Listbox(root,height=4,width=60,selectmode=SINGLE)

listbox.grid(row=0,column=0,columnspan=3)

directory()

root.mainloop()

pygame.quit()