TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN

PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

Phát triển phần mềm Play Audio - Video

GVHD: Từ Lãng Phiêu
Sinh Viên: Nguyễn Bá Lợi - 3120560056
Trần Công Hùng - 3120410197
Nguyễn Hoài Kha - 3120410228

TP. HỒ CHÍ MINH, THÁNG 5/2024

Mục lục

1 Giới thiệu đề tài

Trong thời đại số ngày nay, việc làm việc với âm thanh và video đã trở nên phổ biến và quan trọng hơn bao giờ hết. Đặc biệt, việc phát audio và video không chỉ dành cho giải trí mà còn được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau như giáo dục, truyền thông và công nghệ thông tin. Trong bối cảnh này, nghiên cứu về các phần mềm được thiết kế từ ngôn ngữ Python để phát audio và video không chỉ mang lại sự linh hoạt và hiệu suất mà còn mở ra nhiều cơ hội cho phát triển ứng dụng đa phương tiện..

Ứng dụng sẽ được phát triển với giao diện đồ họa người dùng (GUI) thân thiện, cho phép người dùng dễ dàng thêm danh sách bài hát mà họ muốn nghe, hoặc có thể xoá những bài hát mà họ khuông muốn có trong danh sách. Ngoài ra người dùng có thể play video mà họ muốn phát bằng cách mở thư mục riêng có trên phần mềm. Bên cạnh đó ứng dụng còn có thể play với nhiều định dạng khác nhau như .mp3, .mp4, .flac, .wav, .aiff, .ogg, .mov, .avi. Ngoài ra người dùng còn có thể điều chỉnh âm lượng trên ứng dụng.

2 Cơ sở lý thuyết

2.1 Ngôn ngữ Python

2.1.1 Giới thiệu ngôn ngữ Python

Python là ngôn ngữ lập trình cấp cao, hướng đối tượng, diễn giải với ngữ nghĩa động được phát triển bởi Guido van Rossum. Ban đầu nó được phát hành vào năm 1991. Được thiết kế để dễ sử dụng cũng như vui nhộn, cái tên "Python" là sự gợi nhớ đến nhóm hài kịch người Anh Monty Python. Python nổi tiếng là ngôn ngữ thân thiện với người mới bắt đầu, thay thế Java trở thành ngôn ngữ giới thiệu được sử dụng rộng rãi nhất vì nó xử lý phần lớn sự phức tạp cho người dùng, cho phép người mới bắt đầu tập trung vào việc nắm bắt đầy đủ các khái niệm lập trình hơn là các chi tiết nhỏ.

Python được sử dụng để phát triển web phía máy chủ, phát triển phần mềm, toán học và viết kịch bản hệ thống, đồng thời phổ biến cho Phát triển ứng dụng nhanh và là ngôn ngữ viết kịch bản hoặc kết dính để liên kết các thành phần hiện có vì cấu trúc dữ liệu tích hợp, cấp cao của nó, gõ động và ràng buộc động. Chi phí bảo trì chương trình được giảm bớt bằng Python do cú pháp dễ học và nhấn mạnh vào khả năng đọc. Ngoài ra, sự hỗ trợ của các mô-đun và gói của Python tạo điều kiện thuận lợi cho các chương trình mô-đun và tái sử dụng mã. Python là ngôn ngữ cộng đồng nguồn mở nên rất nhiều lập trình viên độc lập đang liên tục xây dựng thư viện và chức năng cho nó.

2.1.2 Ứng dụng của Python

Dữ liệu và trực quan hóa dữ liệu. Vì tương đối dễ học, Python đã được nhiều người không phải là lập trình viên như kế toán và nhà khoa học áp dụng cho nhiều công việc hàng ngày, chẳng hạn như tổ chức tài chính.

- Phân tích dữ liệu và học máy: Python đã trở thành một công cụ không thể thiếu trong lĩnh vực khoa học dữ liệu. Với sức mạnh

của các thư viện như NumPy, Pandas, và Matplotlib, Python cho phép các nhà phân tích dữ liệu thực hiện các phép tính thống kê phức tạp, tạo ra các biểu đồ trực quan, và xây dựng các mô hình học máy. Các thư viện như TensorFlow và Keras cung cấp cho lập trình viên các công cụ mạnh mẽ để phát triển và huấn luyện mô hình học máy, từ nhận diện hình ảnh đến dự đoán chuỗi thời gian. Python không chỉ là một ngôn ngữ lập trình, mà còn là một hệ sinh thái mạnh mẽ hỗ trợ cho việc nghiên cứu và ứng dụng trong lĩnh vực dữ liệu..

- Phát triển web: Python thường được sử dụng để phát triển back-end của trang web hoặc ứng dụng—những phần mà người dùng không nhìn thấy. Vai trò của Python trong phát triển web có thể bao gồm gửi dữ liệu đến và đi từ máy chủ, xử lý dữ liệu và giao tiếp với cơ sở dữ liệu, định tuyến URL và đảm bảo tính bảo mật. Python cung cấp một số khuôn khổ để phát triển web. Những cái thường được sử dụng bao gồm Django và Flask.Một số công việc phát triển web sử dụng Python bao gồm kỹ sư phụ trợ, nhà phát triển Python, kỹ sư phần mềm và kỹ sư DevOps.
- Tự động hoá và phát triển phần mềm: Python thường được sử dụng để phát triển back-end của trang web hoặc ứng dụng—những phần mà người dùng không nhìn thấy. Vai trò của Python trong phát triển web có thể bao gồm gửi dữ liệu đến và đi từ máy chủ, xử lý dữ liệu và giao tiếp với cơ sở dữ liệu, định tuyến URL và đảm bảo tính bảo mật. Python cung cấp một số khuôn khổ để phát triển web. Những cái thường được sử dụng bao gồm Django và Flask.Một số công việc phát triển web sử dụng Python bao gồm kỹ sư phụ trợ, nhà phát triển Python, kỹ sư phần mềm và kỹ sư DevOps.
- Tự động hoá và phát triển phần mềm: Python không chỉ là một ngôn ngữ lập trình mạnh mẽ, mà còn là một công cụ tuyệt vời cho việc tự động hóa các nhiệm vụ lặp đi lặp lại. Việc viết script trong Python giúp tự động hóa các quy trình như kiểm tra lỗi,

chuyển đổi tệp, và loại bỏ các bản sao dữ liệu. Ngay cả những người mới bắt đầu cũng có thể sử dụng Python để tự động hóa các tác vụ đơn giản trên máy tính, như đổi tên tệp, tìm kiếm và tải xuống nội dung trực tuyến, hoặc gửi email theo lịch trình. Trong phát triển phần mềm, Python cũng hỗ trợ các tác vụ như kiểm soát bản dựng, theo dõi lỗi, và kiểm thử sản phẩm mới. Điều này giúp các nhà phát triển phần mềm tự động hóa các quy trình kiểm thử và đảm bảo chất lượng sản phẩm.

2.1.3 Đặc tính của Python

Python đang trở nên phổ biến trong cộng đồng lập trình nhờ có các đặc tính sau:

- Ngôn ngữ thông dịch: Python được xử lý trong thời gian chạy bởi Trình thông dịch Python.
- Ngôn ngữ hướng đối tượng: Nó hỗ trợ các tính năng và kỹ thuật lập trình hướng đối tượng.
- Ngôn ngữ lập trình tương tác: Người dùng có thể tương tác trực tiếp với trình thông dịch python để viết chương trình.
- Ngôn ngữ dễ học: Python rất dễ học, đặc biệt là cho người mới bắt đầu.
- Cú pháp đơn giản: Việc hình thành cú pháp Python rất đơn giản và dễ hiểu, điều này cũng làm cho nó trở nên phổ biến.
- Dễ đọc: Mã nguồn Python được xác định rõ ràng và có thể nhìn thấy bằng mắt.
- Di động: Mã Python có thể chạy trên nhiều nền tảng phần cứng có cùng giao diện.
- Có thể mở rộng: Người dùng có thể thêm các mô-đun cấp thấp vào trình thông dịch Python.
- Có thể cải tiến: Python cung cấp một cấu trúc cải tiến để hỗ trợ các chương trình lớn sau đó là shell-script.

2.2 Các thư viện sử dụng

2.2.1 Thư viện Pygame

Pygame là một thư viện của ngôn ngữ lập trình Python và là một tập hợp các mô-đun Python được thiết kế riêng để lập trình trò chơi. Pygame được viết bởi Pete Shinners thay thế cho chương trình PySDL sau khi quá trình phát triển dự án này bị đình trệ. Chính thức phát hành từ năm 2000, Pygame được phát hành theo phần mềm miễn phí GNU Lesser General Public License.

Pygame có thể chạy trên nhiều nền tảng và hệ điều hành khác nhau. Với thư viện pygame trong Python, các nhà phát triển có thể sử dụng công cụ và chức năng mở rộng để tạo ra các trò chơi nhập vai ấn tượng. Bởi vậy, Pygame đang ngày càng phổ biến với nhà phát triển vì tính đơn giản, linh hoạt, dễ sử dụng.

Các tính năng chính của Pygame bao gồm:

- Pygame sử dụng Simple DirectMedia Layer (SDL), một thư viện phát triển đa nền tảng cho phép các nhà phát triển có thể truy cập vào phần cứng máy tính như đồ họa, âm thanh và thiết bị đầu vào.
- Xây dựng các trò chơi trên nhiều nền tảng khác nhau như Windows, Mac, Linux thậm chí là cả các thiết bị di động
- Nhà phát triển có thể quản lý tất cả các yếu tố trong quá trình phát triển trò chơi. Đó có thể là các chức năng như xuất đồ họa, xử lý sự kiện, hoạt ảnh, hiệu ứng âm thanh và phát lại nhạc.
- API trực quan và dễ hiểu, hỗ trợ người mới sử dụng hay cả những nhà phát triển có kinh nghiệm đều có thể truy cập được.
- Tính đa phương tiện giúp nhà phát triển có thể ứng dụng để xử lí hình ảnh hay video, mô phỏng, công cụ giáo dục....

Bạn có thể cài đặt Pygame thông qua pip, trình quản lý gói cho Python. Lệnh cài đặt thông thường là: pip install pygame.

2.2.2 Thư viện Tkinter

Tkinter là một thư viện trong ngôn ngữ lập trình Python được sử dụng để tạo giao diện đồ họa người dùng (GUI). "Tkinter" là viết tắt của "Tk interface,", một toolkit đồ họa cung cấp các công cụ để phát triển giao diện người dùng.

Tkinter là một phần của thư viện tiêu chuẩn của Python và đã được tích hợp sẵn trong hầu hết các cài đặt Python. Điều này giúp cho Tkinter trở thành một lựa chọn phổ biến cho việc phát triển ứng dụng với giao diện đồ họa đơn giản trong Python.

Một số đặc điểm của Tkinter bao gồm khả năng tạo các thành phần giao diện như cửa sổ, nút, ô văn bản, và các widget khác để tương tác với người dùng. Tkinter cung cấp cả các sự kiện và phương thức để xử lý tương tác người dùng và thay đổi trạng thái của ứng dụng.

Tkinter được cài đặt mặc định với Python, vì vậy không cần phải cài đặt bổ sung..

2.2.3 OS

Thư viện os trong Python cung cấp các chức năng để tương tác với hệ điều hành, bao gồm truy cập vào các tệp và thư mục, thực thi các lệnh hệ thống, và nhiều hơn nữa.

Các tính năng chính của thư viện OS bao gồm:

- Kiểm tra sự tồn tại của tệp.
- Tạo thư mục mới.
- Thao tác với thư mục:
  - Lấy thư mục hiện tại.
  - Di chuyển thư mục.
  - Xoá thư mục.
- Thao tác với đường dẫn.

– Lấy đường dẫn tuyệt đối.
– Kết hợp đường dẫn.

2.2.4 Mutagen

Thư viện Mutagen là một công cụ mạnh mẽ trong Python cho việc xử lý metadata của các tệp âm thanh.

Các tính năng chính của Mutagen bao gồm:

- Xử lý Metadata: Mutagen cho phép bạn đọc và chỉnh sửa các thông tin metadata trong các tệp âm thanh như tên bài hát, nghệ sĩ, album, năm phát hành, thể loại, và nhiều hơn nữa.
- Đa định dạng hỗ trợ: Ngoài MP3, Mutagen cũng hỗ trợ các định dạng âm thanh phổ biến khác như OGG, FLAC, WAV, AAC, và nhiều định dạng khác nữa.
- Đọc thông tin từ tệp MP3: Trong trường hợp của bạn, khi sử dụng MP3, bạn có thể đọc thông tin từ các tệp MP3 như độ dài của bài hát, bitrate, tần số lấy mẫu, và các thông tin metadata khác.
- Mutagen được thiết kế để sử dụng dễ dàng và cung cấp một API đơn giản và rõ ràng để truy cập và sửa đổi các thông tin trong tệp âm thanh.

Bạn có thể cài đặt Mutagen thông qua pip, trình quản lý gói cho Python. Lệnh cài đặt thông thường là: pip install mutagen

2.2.5 Time

Thư viện time là một công cụ mạnh mẽ cho việc làm việc với thời gian trong Python, cho phép bạn đo đếm thời gian, thực hiện các tính toán thời gian, và kiểm soát luồng của chương trình dựa trên thời gian.

2.2.6 Threading

Thư viện threading trong Python cung cấp một cách tiện lợi để làm việc với các luồng (threads) trong chương trình.

Các tính năng của của Threading bao gồm:

- Tạo luồng: Bằng cách sử dụng threading. Thread, mình có thể tạo một luồng mới cho một hoạt động cụ thể. Điều này cho phép mình thực thi các tác vụ đồng thời trong chương trình của mình.
- Quản lý luồng: Mình có thể khởi động một luồng bằng cách gọi phương thức start() trên một đối tượng luồng. Để chờ một luồng kết thúc thực thi, mình có thể gọi phương thức join(). Điều này cho phép chương trình chính chờ đợi cho đến khi luồng đã hoàn thành trước khi tiếp tục thực hiện các tác vụ tiếp theo.
- Đồng bộ hóa và tương tác giữa các luồng: Thư viện threading cung cấp các cơ chế để đồng bộ hóa truy cập đến dữ liệu chia sẻ giữa các luồng. Bằng cách sử dụng các cấu trúc dữ liệu như Lock, Semaphore, và Event, bạn có thể đảm bảo an toàn khi truy cập vào dữ liệu từ nhiều luồng. Bạn cũng có thể sử dụng các cấu trúc dữ liệu như Queue để truyền dữ liệu giữa các luồng một cách an toàn.
- Xử lý đa luồng trong ứng dụng: Threading cho phép bạn tận dụng lợi ích của xử lý đa luồng trong các ứng dụng Python, giúp tăng hiệu suất và phản hồi của ứng dụng.
- Lập lịch và điều khiển luồng: Bằng cách sử dụng các module như threading.Timer hoặc threading.Event, bạn có thể lập lịch hoặc điều khiển việc thực thi của các luồng trong thời gian cụ thể.

2.2.7 OpenCV

OpenCV là một thư viện mã nguồn mở hàng đầu được sử dụng rộng rãi cho xử lý hình ảnh và thị giác máy tính. Được phát triển bởi Intel vào năm 1999, OpenCV hiện đang được duy trì bởi một cộng

đồng rộng lớn và được sử dụng trong nhiều ứng dụng từ nhận dạng khuôn mặt đến xe tự hành.

Các tính năng chính:

- Xử lý hình ảnh: OpenCV cung cấp các công cụ mạnh mẽ cho việc xử lý hình ảnh như làm mờ, làm nét, biến đổi hình học, phát hiện cạnh, và nhiều hơn nữa.
- Nhận dạng đối tượng: Thư viện này hỗ trợ việc nhận dạng và phát hiện các đối tượng trong hình ảnh, bao gồm khuôn mặt, đối tượng di động, vật thể, vv.
- Thị giác máy tính: OpenCV cho phép bạn thực hiện các tác vụ liên quan đến thị giác máy tính như việc theo dõi chuyển động, phát hiện và theo dõi vật thể, và phân tích hành vi.
- Xử lý video: Thư viện này cung cấp các công cụ để xử lý video như cắt, ghép, chỉnh sửa và phân tích video.

Module cv2:

- Trong Python, OpenCV được thể hiện qua module cv2, là giao diện Python cho thư viện OpenCV C++.
- Module cv2 cung cấp các hàm và lớp để thực hiện các tác vụ xử lý hình ảnh và video, từ đọc/ghi hình ảnh và video đến xử lý và phân tích chúng.

Để cài đặt thư viện OpenCV trong Python, bạn có thể sử dụng pip, công cụ quản lý gói chuẩn của Python. Đây là cách cài đặt OpenCV: pip install opencv-python

3 Xây dựng phần mềm

3.1 Xây dựng PlayerModel

3.1.1 Import thư viện

```
from pygame import mixer
import time
```

Đầu tiên, chúng ta import các thư viện cần thiết: 'pygame', 'time'. Thêm mô-đun mixer từ thư viện pygame cho phép thực hiện các việc đa phương tiên, chẳng hạn như là phát lại âm thanh. Tiếp theo thêm thư viện Time để cho phép kiểm soát thời gian trong chương trình.

3.1.2 Tạo lớp playlist và playmodel

```
class audioPlayerModel:...

class playList:...
```

Đầu tiên chúng ta tạo ra lớp phụ playlist đóng vai trò quản lý danh sách phát. Nó chứa các phương thức để chuyển đến bài hát tiếp theo hoặc trước đó trong danh sách phát.

Sau đó chúng ta tạo ra một lớp audioPlayerModel đây là một lớp quan trọng được thiết kế theo mô hình MVC (Model-View-Controller). Mô hình này chứa các phương thức và thuộc tính để điều khiển và quản lý âm nhạc được phát trong ứng dụng.

3.1.3 Lớp playList

```
def __init__(self, list,view,model):
    self.list = list
    self.index = 0
    self.current_song=None
    self.view=view
    self.model=model
```

Đoạn code này khởi tạo phương thức của lớp playList, nhận các tham số là list, view, và model. Các tham số này là danh sách phát, giao diện người dùng và mô hình của ứng dụng.

```
def next(self):
    try:
        time.sleep(1)
        self.view.model.stop=False
        result = self.list[self.index]
        self.index += 1
        self.current_song = self.list[self.index]
        mixer.music.load(self.current_song)
        mixer.music.play()
        self.view.show_details(self.current_song)

    except IndexError:
        self.index = 0
        result = self.list[self.index]
        self.current_song = self.list[self.index]
        mixer.music.load(self.current_song)
        mixer.music.play()
        self.view.show_details(self.current_song)
    return result
```

Phương thức này cung cấp cơ chế để chuyển đến bài hát tiếp theo trong danh sách phát, xử lý các trường hợp khi đã đến cuối danh sách phát và quay lại phát lại từ đầu nếu cần.

Bắt đầu với một khối lệnh try-except để xử lý ngoại lệ. Dòng time.sleep(1) này tạm ngưng thực thi cho đến khi đã trôi qua 1 giây. Điều này có thể được sử dụng để tạo ra một khoảng thời gian chờ giữa các bài hát. Bắt đầu đặt cờ stop trong mô hình của giao diện người dùng thành False. Điều này có thể được sử dụng để thông báo rằng nhạc đang được phát và không nên dừng lại. Sau đó gán bài hát hiện tại trong danh sách phát vào biến result. self.index += 1 tăng chỉ số của bài hát hiện tại để chuyển đến bài hát tiếp theo trong

danh sách phát. Và gán bài hát tiếp theo trong danh sách phát vào biến `self.current_song`. Sau đó dùng thư viện mixer tải bài hát tiếp theo trong danh sách phát để chuẩn bị cho việc phát lại. Dùng hàm play() để bắt đầu phát bài hát tiếp theo, cuối cùng hiển thị thông tin chi tiết về bài hát đang được phát lại trên giao diện người dùng. Tương tự như khối lệnh except khi xảy ra lỗi thì ứng dụng sẽ bắt đầu thực hiện phát lại bài hát đầu tiên trong danh sách. Cuối cùng return result trả về bài hát hiện tại để bắt đầu phát lại.

```
def prev(self):
    time.sleep(1)
    self.view.model.stop = False
    self.index -= 1
    if self.index < 0:
        self.index = len(self.list)-1
        self.current_song = self.list[self.index]
        mixer.music.load(self.current_song)
        mixer.music.play()
        self.view.show_details(self.current_song)
    else:
        self.current_song = self.list[self.index]
        mixer.music.load(self.current_song)
        mixer.music.play()
        self.view.show_details(self.current_song)
        return self.list[self.index]
```

Phương thức này cung cấp cơ chế để chuyển đến bài hát trước đó trong danh sách phát, xử lý trường hợp khi đã ở đầu danh sách phát và quay lại phát lại từ cuối danh sách phát nếu cần.

Thực hiện việc chuyển giao giữa 2 bài hát là 1 giây và đặt cờ hiệu là False. Sau đó giảm chỉ số index - 1 của bài hát hiện tại để chuyển đến bài hát trước trong danh sách phát. Kiểm tra xem chỉ số của bài hát hiện tại đã nhỏ hơn 0 hay chưa. Nếu chỉ số của bài hát hiện tại nhỏ hơn 0, nghĩa là đã đến đầu danh sách phát, ta sẽ đặt chỉ số của bài hát hiện tại là chỉ số của bài hát cuối cùng trong danh sách phát. Sau đó gán bài hát trong danh sách phát vào biến `self.current_song`. Và cuối cùng là tải dữ liệu, chạy bài hát và hiển thị thông tin lên giao diện người dùng. Trường hợp nếu chỉ số của bài hát hiện tại không nhỏ hơn 0. Thì gán bài hát trước đó trong danh sách phát vào biến `self.current_song` và thực hiện load bài hát, chạy bài hát và hiển thị thông tin lên giao diện người dùng.

```
def __iter__(self):
    return self
```

Phương thức này cho phép lớp playList có thể lặp qua các phần tử trong danh sách phát.

3.1.4 Lớp audioPlayerModel

```
def __init__(self, view):

  self.play=False
  self.paused=False
  self.stop=False
  self.play_list = []
  self.playListIterator = playList(self.play_list,view,self)
  self.view = view
  self.muted = False
  self.current_song = None
  mixer.init()
```

Phương thức khởi tạo này thiết lập các thuộc tính ban đầu của đối tượng audioPlayerModel, bao gồm cài đặt các cờ điều khiển, khởi tạo danh sách phát và các thành phần liên quan đến việc phát lại âm nhạc.

Đặt biến play thành False, đây là một cờ để xác định xem âm nhạc có đang được phát hay không.

Đặt biến paused thành False, đây là một cờ để xác định xem âm nhạc có đang được tạm dừng hay không.

Đặt biến stop thành False, đây là một cờ để xác định xem âm nhạc có đã dừng hay không.

Khởi tạo danh sách phát rỗng, danh sách này sẽ chứa các bài hát được thêm vào để phát lại.

`self.playListIterator = playList(self.play_list,view,self)`: Khởi tạo một đối tượng của lớp playList, được sử dụng để quản lý việc lặp lại danh sách phát.

Đặt biến muted thành False, đây là một cờ để xác định xem âm

thanh có đang bị tắt hay không.

Khởi tạo biến `current_song` thành None, biến này sẽ chứa thông tin về bài hát hiện đang được phát.

Khởi tạo mô-đun mixer của pygame để sẵn sàng cho việc phát lại âm thanh.

```
def play_Music(self):
    self.stop=False
    if self.paused:
        #mixer.music.unpause()
        mixer.music.play(start=int(self.view.my_slider.get()))
        self.paused=False
        self.play=True
    else:
        try:
            if self.play is False:
                time.sleep(1)
                selected_song = self.view.list_song.curselection()
                current_selected = int(selected_song[0])
                self.current_song = self.play_list[current_selected]
                mixer.music.load(self.current_song)
                mixer.music.play(self.view.my_slider.get())
                self.play = True
                self.view.show_details(self.current_song)
        except:
         return None
```

Phương thức playMusic này cho phép người dùng phát lại âm nhạc từ vị trí đã tạm dừng hoặc phát lại một bài hát mới từ danh sách phát.

self.stop=False: Đặt biến stop thành False, cho biết rằng âm nhạc chưa được dừng lại.

Kiểm tra nếu âm nhạc đã được tạm dừng.

- mixer.music.play(start=int(self.view.my_slider.get())): Phát lại âm nhạc từ vị trí được xác định bởi thanh trượt trên giao diện người dùng. Điều này cho phép người dùng tiếp tục phát lại âm nhạc từ vị trí đã tạm dừng trước đó.
- self.paused=False: Đặt biến paused thành False, cho biết rằng âm nhạc không còn ở trạng thái tạm dừng nữa.
- Và cuối cùng set play=True

Nếu âm nhạc không được tạm dừng trước đó, chúng ta sẽ thực hiện việc phát lại một bài hát mới.

- selected_song = self.view.list_song.curselection(): Lấy vị trí của bài hát được chọn trong danh sách phát trên giao diện người dùng.
- current_selected = int(selected_song[0]): Chuyển đổi vị trí được chọn thành một số nguyên.
- self.current_song = self.play_list[current_selected]: Gán bài hát được chọn từ danh sách phát vào biến current_song.
- Tải bài hát được chọn và chuẩn bị cho việc bắt đầu phát lại. Hiển thị thông tin chi tiết lên giao diện.

Xử lý ngoại lệ nếu có bất kỳ lỗi nào xảy ra trong quá trình phát lại âm nhạc. return None: Trả về None nếu có lỗi xảy ra trong quá trình phát lại âm nhạc.

```
def stop_Music(self):
    #self.pasued = False
    self.play = False
    self.stop=True
    mixer.music.stop()
```

Phương thức này cung cấp một cách an toàn để dừng phát âm nhạc và cập nhật trạng thái của các biến điều khiển trong lớp audioPlayerModel.

- self.play = False: Đặt biến play thành False, cho biết rằng không có bài hát nào đang được phát hiện tại.
- self.stop=True: Đặt biến stop thành True, cho biết rằng âm nhạc đã được dừng lại.
- mixer.music.stop(): Dừng phát lại âm nhạc bằng cách gọi phương thức stop() từ mô-đun mixer của pygame.

```
def del_song(self,selected_song):
    self.stop_Music()
    self.play_list.pop(selected_song)
```

Phương thức này kết hợp việc dừng phát âm nhạc và xóa bài hát khỏi danh sách phát một cách an toàn, giúp đảm bảo rằng việc xóa bài hát không gây ra vấn đề gì đối với hoạt động của ứng dụng.

- self.stop_Music(): Gọi phương thức stop_Music() để dừng phát lại âm nhạc. Điều này đảm bảo rằng âm nhạc sẽ dừng lại trước khi bài hát được xóa khỏi danh sách phát.
- self.play_list.pop(selected_song): Xóa bài hát được chỉ định khỏi danh sách phát. Phương thức pop() loại bỏ phần tử tại vị trí chỉ định (selected_song) khỏi danh sách và trả về giá trị của phần tử đó.

```
def add_to_playlist(self,filename):
 index = len(self.play_list)
 self.play_list.insert(index, filename)
 self.current_song = self.play_list[0]
```

Phương thức này cho phép thêm một bài hát vào danh sách phát và cập nhật bài hát hiện đang được chọn để phát lại nếu không có bài hát nào khác đang phát.

- index = len(self.play_list): Tính độ dài hiện tại của danh sách phát để xác định vị trí mới của bài hát sẽ được thêm vào.
- self.play_list.insert(index, filename): Chèn bài hát được chỉ định (filename) vào danh sách phát tại vị trí cuối cùng của danh sách.
- self.current_song = self.play_list[0]: Gán bài hát đầu tiên trong danh sách phát vào biến current_song, đảm bảo rằng bài hát mới thêm vào sẽ được chọn để phát lại nếu không có bài hát nào khác đang được phát.

```
def mute_Music(self):
    if self.muted:
        mixer.music.set_volume(0.7)
        self.view.scale.set(70)
        self.muted = False
    else:
        mixer.music.set_volume(0)
        self.view.scale.set(0)
        self.muted = True
```

Phương thức này cho phép người dùng tắt hoặc bật âm thanh của âm nhạc và cập nhật trạng thái hiển thị trên giao diện người dùng tương ứng.

- Kiểm tra biến mute nếu ở trạng thái true thì âm thanh sẽ bị tắt bằng cách set_volume = 0.
- Còn nếu kiểm tra biến mute ở trạng thái false thì âm thanh sẽ được điều chỉnh mặc định ở mức 70%.

```
def set_volume(self,val):
    if self.muted is True:
        return
    volume =int(val)/100
    mixer.music.set_volume(volume)
```

Phương thức này cho phép cài đặt âm lượng của âm nhạc dựa trên giá trị được chỉ định và kiểm tra xem âm thanh có đang bị tắt hay không trước khi thực hiện thay đổi.

- if self.muted is True:: Kiểm tra xem âm thanh có đang bị tắt hay không.
- Nếu âm thanh đã bị tắt (self.muted==True), phương thức sẽ kết thúc mà không làm gì cả, do không cần thiết phải thay đổi âm lượng khi âm thanh đã được tắt.
- Nếu âm thanh không bị tắt tạo biến volume=int(val)/100: Chuyển đổi giá trị của val từ định dạng phần trăm sang giá trị thực của âm lượng (từ 0 đến 1).
- mixer.music.set_volume(volume): Đặt âm lượng của âm nhạc thành giá trị volume đã tính toán.

```
def pause_Music(self):
    self.paused=True
    self.play=False
    mixer.music.pause()
```

Phương thức này cho phép tạm dừng âm nhạc và cập nhật trạng thái của các biến điều khiển trong lớp audioPlayerModel tương ứng.

- self.paused=True: Đặt biến paused thành True, cho biết rằng âm nhạc đang ở trạng thái tạm dừng.
- self.play=False: Đặt biến play thành False, cho biết rằng không có bài hát nào đang được phát lại.
- mixer.music.pause(): Tạm dừng phát lại âm nhạc bằng cách gọi phương thức pause() từ mô-đun mixer của pygame.

```
def previous_Music(self):
 self.stop = True
 self.playListIterator.prev()
 self.view.my_slider.config(value=0)
```

Phương thức này cung cấp khả năng chuyển đến bài hát trước đó trong danh sách phát và cập nhật trạng thái của thanh trượt trên giao diện người dùng.

- self.stop = True: Đặt biến stop thành True, cho biết rằng việc chuyển đến bài hát trước đó đang được thực hiện.
- self.playListIterator.prev(): Gọi phương thức prev() từ đối tượng playListIterator để di chuyển đến bài hát trước đó trong danh sách phát.
- self.view.my_slider.config(value=0): Thiết lập giá trị của thanh trượt trên giao diện người dùng về 0, đảm bảo rằng thanh trượt sẽ được đặt lại khi chuyển đến bài hát trước đó.

```
def next_selection(self):
    selection_indices = self.list_song.curselection()

    # default next selection is the beginning
    next_selection = 0

    # make sure at least one item is selected
    if len(selection_indices) > 0:
        # Get the last selection, remember they are strings for
        # so convert to int
        last_selection = int(selection_indices[-1])
        next_selection = last_selection + 1
    if int(selection_indices[-1]) == self.list_song.size() - 1:
        last_selection = int(selection_indices[-1])
        next_selection = 0

    self.list_song.selection_clear(last_selection)
    self.list_song.activate(next_selection)
    self.list_song.selection_set(next_selection)
```

Phương thức này cung cấp khả năng chuyển đến bài hát tiếp theo trong danh sách phát và cập nhật trạng thái của thanh trượt trên giao diện người dùng.

- Đặt biến stop thành True cho biết việc dừng bài hát và cbi cho việc chuyển đến bài hát tiếp theo.
- Gọi phương thức next() từ đối tượng playListIterator để di chuyển đến bài hát tiếp theo trong danh sách phát.
- Thiết lập giá trị của thanh trượt trên giao diện người dùng về 0, đảm bảo rằng thanh trượt sẽ được đặt lại khi chuyển đến bài hát tiếp theo.

```
def slider_Music(self):
    if self.paused==False:
        mixer.music.play(start=int(self.view.my_slider.get()))
```

Phương thức này cho phép điều chỉnh vị trí phát lại của âm nhạc dựa trên vị trí hiện tại của thanh trượt trên giao diện người dùng và đảm bảo rằng âm nhạc sẽ tiếp tục phát lại từ vị trí mới được chọn.

- if self.paused==False: Kiểm tra xem âm nhạc có đang trong trạng thái không bị tạm dừng hay không.
- mixer.music.play(start=int(self.view.my_slider.get())): Nếu âm nhạc không bị tạm dừng, phương thức sẽ phát lại âm nhạc từ vị trí được xác định bởi giá trị hiện tại của thanh trượt trên giao diện người dùng. Điều này cho phép người dùng chuyển đến một vị trí cụ thể trong bài hát bằng cách di chuyển thanh trượt.

```
def write_List_To_File(self, lines):
    list = []
    list = lines
    while '\n' in list: list.remove('\n')
    with open("MP3MusicList.txt", "w", encoding='UTF-8') as playlist_File:
        if(len(list) == 0):
            playlist_File.close()
        else:
            playlist_File.write('\n'.join(list))
            playlist_File.close()
```

Phương thức này cho phép ghi danh sách các bài hát vào một tệp văn bản với mã hóa UTF-8, loại bỏ các ký tự xuống dòng không mong muốn, và đảm bảo rằng tệp văn bản được đóng sau khi hoàn tất việc ghi.

- Bắt đầu khởi tạo một danh sách tróng list=[].

- Gán danh sách đầu vào (lines) cho danh sách mới được khởi tạo. Điều này sẽ sao chép tất cả các phần tử từ danh sách đầu vào sang danh sách mới.
- while ” in list: list.remove(”): Loại bỏ các ký tự xuống dòng (”) khỏi danh sách mới. Điều này đảm bảo rằng danh sách sẽ không chứa bất kỳ ký tự xuống dòng nào.
- with open("MP3MusicList.txt", "w", encoding=’UTF-8’) as playlist_File: Mở tệp văn bản có tên "MP3MusicList.txt" để ghi dữ liệu. Tệp này sẽ được mở trong chế độ ghi ("w"), có mã hóa UTF-8.
- Kiểm tra xem danh sách có rỗng hay không. Nếu rỗng thì hoàn tất việc ghi file.
- Nếu danh sách không rỗng thì thực hiện các việc sau: playlist_File.write(”.join(li Ghi nội dung của danh sách vào tệp văn bản, mỗi phần tử trên một dòng. Phương thức join() được sử dụng để nối các phần tử trong danh sách với ký tự xuống dòng (”) giữa chúng. Sau đó hoàn tất việc ghi file và kết thúc file.

3.2 Xây dựng PlayerView

3.2.1 Import thư viện

```
from tkinter import *
import tkinter.messagebox
import os
from tkinter import filedialog
from audioPlayerModel import audioPlayerModel
from pygame import mixer
from mutagen.mp3 import MP3
import time
import threading
import tkinter.ttk as ttk

from tkinter import filedialog
import cv2
```

Đầu tiên, chúng ta import các thư viện cần thiết như: 'tkinter', 'os', 'pygame.mixer', 'mutagen.mp3', 'time', 'threading', 'cv2' và trình xử lý logic từ model audioPlayerModel. Thư viện tkinter được sử dụng để tạo giao diện người dùng đồ họa trong Python, os cho phép tương tác với hệ thống tập tin, pygame.mixer được sử dụng để phát âm thanh, mutagen.mp3 được sử dụng để độc thông tin từ các tệp MP3, time được xử dụng để quản lý thời gian, threading cho phép thực các tác vụ đồng thời, cv2 thư viện OpenCV cho phép làm việc với video ,và audioPlayerModel sử dụng logic để quản lý việc phát lại âm nhạc và tương tác với người dùng thông qua giao diện người dùng.

3.2.2 Khởi tạo lớp audioPlayerView

```
class audioPlayerView:
    def __init__(self):…

    def select_media(self):…

    def play_video(self, video_url):…
    #this function open the browse file to choose one file
    def browse_file(self):…


    def load_Playlist_From_File(self):…


    def update_Play_Music(self):…


    def update_Delet_Song(self):…

    def update_Mute_Music(self):…

    def update_Stop_Music(self):…


   #show current time of song and show on the screen
    def show_details(self, play_song):…


     #start to count the time in different thread.
    def start_count(self,total_length):…


    def update_Pause_Music(self):…

    def update_Next_Music(self):…

    def update_Previous_Music(self):…



    def slide_Music(self,x):…

    def show_current_time(self,time):…


     ##show totel legnth of song
    def show_total_time(self,time):…

    def next_selection(self):…

    def perv_selection(self):…


    def update_set_Volume(self):…

   # function that start when we press to exit from t
    def on_closing(self):…
```

- __init__(): Khởi tạo giao diện người dùng, các thành phần UI như nút, thanh trượt, hộp danh sách và hộp thoại.
- Các phương thức update_*_Music(): Được gọi khi người dùng tương tác với các nút như Play, Pause, Stop, Next, Previous để điều khiển phát lại âm nhạc.
- Các phương thức browse_file(), load_Playlist_From_File(): Được sử dụng để tải danh sách bài hát từ một tệp văn bản và cho phép người dùng chọn bài hát từ hệ thống tệp.
- Các phương thức show_current_time(), show_total_time(): Được sử dụng để hiển thị thời gian hiện tại và tổng thời gian của bài hát đang được phát lại.
- Phương thức start_count(): Sử dụng để tính thời gian hiện tại của bài hát và cập nhật thanh trượt thời gian của bài hát.
- Phương thức on_closing(): Được gọi khi ứng dụng đóng, để lưu danh sách bài hát hiện tại vào một tệp văn bản trước khi thoát.
- Phương thức select_media() và play_video(): Cho phép người dùng chọn và phát video bằng cách sử dụng thư viện OpenCV.

3.2.3 Các phương thức được khởi tạo

```
def __init__(self):
    #init Tkitner
    self.total_length=0
    self.current_time=0
    self.lines = []
    self.root = Tk()
    self.root.title("PlayMusic_Nhom9")
    self.root.iconbitmap(r'play_button_JpT_icon.ico')

    menubar = Menu(self.root)  # create menubar
    self.root.config(menu=menubar)

    # create status massage
    statusbar = Label(self.root, text="Welcom to my player", relief=SUNKEN)
    statusbar.pack(side=BOTTOM, fill=X)
    self.statusbar=statusbar

    submenu = Menu(menubar)

    #left frame
    left_frame = Frame(self.root)
    left_frame.pack(side=LEFT, padx=30)

    #right frame
    right_frame = Frame(self.root)
    right_frame.pack(side=RIGHT)

    #middle frame
    middle_frame = Frame(right_frame)
    middle_frame.pack(padx=10, pady=10)


    top_frame = Frame(right_frame)
    top_frame.pack()

    middle_frame = Frame(right_frame)
    middle_frame.pack(padx=10, pady=10)
```

```
    #buttom frame
    bottomframe = Frame(right_frame)
    bottomframe.pack()


    self.lenght_lable = Label(top_frame, text="Totel length :00:00 ")
    self.lenght_lable.pack()

    self.curren_Time_lable = Label(top_frame, text="current Time  :00:00 ")
    self.curren_Time_lable.pack()  #

    list_song = Listbox(left_frame)
    list_song.pack()
    self.list_song = list_song
    self.model = audioPlayerModel(self)
    self.load_Playlist_From_File()

    #images
    self.play_photo = PhotoImage(file='play-button.png')
    self.volume_photo = PhotoImage(file='volume.png')
    self.mute_photo = PhotoImage(file='mute.png')
    self.stop_photo = PhotoImage(file='stop.png')
    self.pause_photo = PhotoImage(file='pause.png')
    self.next_photo = PhotoImage(file='next.png')
    self.previous_photo = PhotoImage(file='previous.png')


    self.add_btn = Button(left_frame, text="Thêm bài hát",command=self.browse_file)
    self.add_btn.pack(side=LEFT, padx=10)

    self.del_btn = Button(left_frame, text="Xoá bài hát",command=self.update_Delet_Song)
    self.del_btn.pack(side=LEFT)

    self.play_video_btn = Button(left_frame, text="Mở Video", command=self.select_media)
    self.play_video_btn.pack(side=LEFT, padx=10)
```

```
self.btn_mute = Button(bottomframe, image=self.volume_photo, command=self.update_Mute_Music)
self.btn_mute.grid(row=0, column=1)

self.btn_play = Button(middle_frame, image=self.play_photo,command=self.update_Play_Music)
self.btn_play.grid(row=0, column=1, padx=10)

self.btn_stop = Button(middle_frame, image=self.stop_photo, command=self.update_Stop_Music)
self.btn_stop.grid(row=0, column=2, padx=10)

self.btn_pause = Button(middle_frame, image=self.mute_photo, command=self.update_Pause_Music)

self.btn_next = Button(middle_frame, image=self.next_photo, command=self.update_Next_Music)
self.btn_next.grid(row=0, column=3)

self.btn_previous = Button(middle_frame, image=self.previous_photo, command=self.update_Previous_Music)
self.btn_previous.grid(row=0, column=0, )

self.scale = Scale(bottomframe, from_=0, to=100, orien=HORIZONTAL,command=self.model.set_volume)  # scal
self.scale.set(50)  # set the initial valume to be 50
mixer.music.set_volume(0.5)
self.scale.grid(row=0, column=2, pady=15, padx=30)

#create slider of the song time
self.my_slider=ttk.Scale(top_frame, from_=0, to_=100, orient=HORIZONTAL, value=0,
command=self.slide_Music, length=250)
self.my_slider.pack(side=BOTTOM)

self.root.protocol("WM_DELETE_WINDOW", self.on_closing)
self.root.mainloop()
```

Phương thức __init__() này cung cấp cấu trúc cơ bản cho giao diện người dùng của ứng dụng và kết nối các hành động của người dùng với các chức năng xử lý của ứng dụng.

- Khởi tạo cửa sổ giao diện: Một cửa sổ Tkinter được tạo ra với tiêu đề "PlayMusic_Nhom9" và biểu tượng (icon) được thiết lập bằng hình ảnh play_button_JpT_icon.ico. Một thanh trạng thái (status bar) được tạo để hiển thị các thông điệp "Welcom to my player" và được đặt ở phía dưới cùng của cửa sổ.
- Tạo khung và các thành phần giao diện: Các khung (frames) được tạo ra để sắp xếp các thành phần của giao diện người dùng. Các thành phần như nhãn (label), danh sách (listbox), và các nút (button) được thêm vào các khung tương ứng.
- Tải danh sách bài hát từ tệp và khởi tạo các hình ảnh: Danh sách bài hát được tải từ tệp MP3MusicList.txt và hiển thị trong hộp danh sách. Các hình ảnh (biểu tượng) được tải từ các tệp hình ảnh để sử dụng cho các nút như Play, Pause, Stop, và các nút khác.
- Gắn các hàm xử lý sự kiện với các nút: Để thực hiện các hành động như thêm bài hát, xóa bài hát, mở video, điều khiển phát lại âm nhạc, vv.
- Tạo thanh trượt và thực hiện sự kiện đóng cửa sổ: Thanh trượt được tạo để điều chỉnh âm lượng và thời gian của bài hát. Sự

kiện đóng cửa sổ được gắn với phương thức on_closing() để lưu danh sách bài hát vào tệp trước khi ứng dụng đóng.

```
def browse_file(self):
    global filePath
    filePath = filedialog.askopenfilename()
    if(filePath != ''):
      filename = os.path.basename(filePath)
      index = self.list_song.size()
      self.list_song.insert(index, filename)
      self.list_song.pack()
      self.model.add_to_playlist(filePath)
      self.lines.insert(index, filePath)
      lastIndex = len(self.lines)
      if lastIndex > 0:
        self.list_song.selection_clear(0)
        self.list_song.activate(0)
        self.list_song.selection_set(0)
```

Phương thức browse_file(self) được sử dụng để mở hộp thoại chọn tệp và thêm tệp được chọn vào danh sách bài hát của ứng dụng.

- Đầu tiên dùng filedialog.askopenfilename(): Để mở hộp thoại chọn tệp và trả về đường dẫn của tệp đã chọn.
- Sau đó kiểm tra xem đã chọn tệp hay chưa. Nếu đã chọn tệp thì lấy tên tệp từ đường dẫn đã được chọn, sau đó lấy số lượng mục hiện có trong danh sách bài hát, bắt đầu thêm tên tệp vào danh sách bài hát ở vị trí cuối cùng. Sau đó cập nhật và hiển thị danh sách bài hát, dùng model.add_to_playlist để lưu đường dẫn của tệp vào danh sách phát trong mô hình, kế tiếp là thêm đường dẫn của tệp vào danh sách đường dẫn trong giao diện người dùng. Cuối cùng là lấy độ dài của danh sách đường dẫn.
- Nếu như độ dài của danh sách đường dẫn có ít nhất 1 phần tử thì bỏ chọn tất cả các mục trong danh sách sau đó kích hoạt và chọn mục đầu tiên trong danh sách phát, .

```
def load_Playlist_From_File(self):
    try:
        with open("MP3MusicList.txt", "r", encoding='UTF-8') as playlist_File:
            list = playlist_File.readlines()
            for element in list:
                if not element.isspace():
                    self.lines.append(element)
            for filePath in self.lines:
                index = self.list_song.size()
                f = os.path.basename(filePath.rstrip())
                self.list_song.insert(index, f)
                self.list_song.pack()
                self.model.add_to_playlist(filePath.rstrip())  # need to check
                lastIndex = len(self.lines)
                if lastIndex > 0:
                    self.list_song.selection_clear(0)
                    self.list_song.activate(0)
                    self.list_song.selection_set(0)
    except:
        print("file not found")
        playlist_File = open("MP3MusicList.txt", "w+", encoding='UTF-8')
```

Phương thức load_Playlist_From_File(self) được sử dụng để đọc danh sách các bài hát từ tệp "MP3MusicList.txt" và hiển thị chúng trong giao diện người dùng. Thêm đường dẫn của các bài hát vào danh sách phát trong mô hình người dùng. Hiển thị tên của các bài hát trong danh sách bên trong giao diện người dùng. Khi người dùng khởi chạy ứng dụng, phương thức này được gọi để tải danh sách các bài hát đã lưu trước đó từ tệp "MP3MusicList.txt" vào giao diện người dùng.

- Đầu tiên chúng ta mở tệp MP3MusicList.txt để đọc nội dung bằng cách sử dụng câu lệnh with nhằm đảm bảo khi không còn sử dụng thì tệp sẽ được tự động đống.
- Dùng hàm readlines() để đọc nội dung của tệp và lưu vào danh sách list, trong đó mỗi dòng của tệp là một phần tử trong danh sách.
- Dùng hàm for để duyệt qua từng phần tử của danh sách list và bắt đầu kiểm tra từng phần tử đó có phải là khoảng trắng hết hay không. Nếu là khoảng trắng hết thì nó không được thêm vào danh sách.
- Tiếp theo duyệt qua từng phần tử trong danh sách bài hát, và lấy tên đường dẫn lưu vào biến f, sau đó thêm tên tệp vào danh sách phát của người dùng và thêm đường dẫn vào danh sách phát của mô hình người dùng.

```
def update_Delet_Song(self):
    selected_song = self.list_song.curselection()
    self.list_song.delete(selected_song)
    selected_song = int(selected_song[0])
    self.lines.pop(selected_song) #delete the selected song
    self.model.del_song(selected_song)
    if (selected_song > 0):
        self.list_song.activate(selected_song - 1)
        self.list_song.selection_set(selected_song - 1)
    if (selected_song == 0 and len(self.lines) != 0):
        self.list_song.activate(selected_song)
        self.list_song.selection_set(selected_song)
    self.btn_play.configure(image=self.play_photo)
    self.my_slider.config(value=0)
    self.show_current_time(0)
    self.show_total_time(0)
```

- Phương thức này được gọi khi người dùng muốn xóa một bài hát khỏi danh sách phát.
- Đầu tiên, nó lấy ra bài hát được chọn bằng cách sử dụng phương thức curselection() của đối tượng Listbox list_song.
- Sau đó, nó xóa bài hát được chọn từ danh sách bằng cách sử dụng phương thức delete() của Listbox.
- Tiếp theo, nó loại bỏ bài hát đã chọn khỏi danh sách bằng cách sử dụng phương thức pop() trên danh sách lines.
- Sau khi xóa, nó gọi phương thức del_song() của đối tượng model để xóa bài hát khỏi danh sách phát trong model.
- Nếu bài hát được xóa không phải là bài hát cuối cùng trong danh sách, nó chọn bài hát tiếp theo trong danh sách bằng cách thay đổi chỉ mục được kích hoạt và chỉ mục được chọn trong Listbox.
- Nếu bài hát được xóa là bài hát đầu tiên trong danh sách và danh sách không trống, nó cũng chọn bài hát tiếp theo trong danh sách.

- Cuối cùng, nó cập nhật hình ảnh của nút play sang hình ảnh ban đầu (play_photo), thiết lập giá trị của thanh trượt của slider về 0, và cập nhật hiển thị thời gian hiện tại và thời gian tổng cộng của bài hát sang giá trị mặc định là 0.

```
def update_Mute_Music(self):
    self.model.mute_Music()
    if(self.model.muted):
     self.btn_mute.configure(image=self.mute_photo)
    else:
     self.btn_mute.configure(image=self.volume_photo)
```

- Phương thức này được gọi khi người dùng muốn tắt hoặc bật âm thanh.
- Đầu tiên, nó gọi phương thức mute_Music() của đối tượng model để tắt hoặc bật âm thanh.
- Nếu âm thanh đã được tắt, nó cập nhật hình ảnh của nút mute sang hình ảnh mute (mute_photo). Nếu âm thanh được bật, nó cập nhật hình ảnh của nút mute sang hình ảnh volume (volume_photo). Điều này giúp người dùng biết trạng thái hiện tại của âm thanh.

```
def update_Stop_Music(self):
    self.model.stop_Music()
    self.statusbar['text'] = "Music Stop"
    self.btn_play.configure(image=self.play_photo)
    self.my_slider.config(value=0)
    self.show_current_time(0)
```

- Phương thức này được gọi khi người dùng muốn tắt bài hát.
- Đầu tiên, nó gọi phương thức stop_Music() của đối tượng model để dừng nhạc.

- Sau đó, nó cập nhật văn bản của thanh trạng thái (statusbar) để thông báo rằng âm nhạc đã dừng.
- Tiếp theo, nút phát nhạc (btn_play) được cấu hình để hiển thị hình ảnh của nút phát (play_photo), để người dùng biết rằng âm nhạc có thể được phát lại từ đầu.
- Thanhs trượt thời gian của bài hát (my_slider) được cài đặt lại về giá trị ban đầu là 0.
- Cuối cùng, thời gian hiện tại của bài hát (current_time) được cập nhật và hiển thị trên giao diện người dùng bằng cách gọi phương thức show_current_time(0) với tham số là 0, tức là thời gian bắt đầu.

```
def update_Play_Music(self):
    if(self.model.play):
        self.update_Pause_Music()
        return

    self.model.play_Music()
    if(self.model.current_song!=None):
        self.statusbar['text'] = "playing music  " + os.path.basename(self.model.current_song)
    else:
        tkinter.messagebox.showerror("Eror", "Not found music to play !")
    if(self.model.play):
        self.btn_play.configure(image=self.pause_photo)
```

- Phương thức này kiểm tra xem có đang phát nhạc không bằng cách kiểm tra giá trị của biến play trong đối tượng model. Nếu đang phát, nó gọi phương thức update_Pause_Music để tạm dừng nhạc và trả về ngay sau đó.
- Nếu không đang phát nhạc, nó gọi phương thức play_Music của đối tượng model để bắt đầu phát nhạc. Nếu bài hát hiện tại không tồn tại, nó hiển thị một hộp thoại thông báo lỗi. Nếu bắt đầu phát nhạc thành công, nó cập nhật thanh trạng thái để hiển thị tên của bài hát đang phát và thay đổi hình ảnh của nút play sang hình ảnh tạm dừng (pause_photo).

```
def show_details(self, play_song):
    file_data = os.path.splitext(play_song)
    if file_data[1] == ".mp3":
        audio = MP3(play_song)
        self.total_length = audio.info.length
    else:
        a = mixer.Sound(play_song)
        self.total_length = a.get_length()

    self.show_total_time(self.total_length)
    self.statusbar['text'] = "playing music  " + os.path.basename(play_song)
    t1 = threading.Thread(target=self.start_count, args=(self.total_length,))
    t1.start()
```

- Phương thức này được sử dụng để hiển thị chi tiết về bài hát đang được phát, bao gồm độ dài tổng cộng của bài hát và tên bài hát trên thanh trạng thái.
- Đầu tiên, phương thức này nhận đầu vào là đường dẫn của bài hát đang được phát (play_song).
- Tiếp theo, nó kiểm tra phần mở rộng của tệp âm thanh (play_song). Nếu nó là một tệp MP3, nó sẽ sử dụng thư viện Mutagen để lấy thông tin về độ dài của bài hát bằng cách tạo một đối tượng MP3 và truy cập thuộc tính info.length. Nếu không, nó sẽ sử dụng thư viện Pygame để lấy độ dài của bài hát bằng cách tạo một đối tượng mixer.Sound và gọi phương thức get_length().
- Sau đó, nó gọi phương thức show_total_time để hiển thị độ dài tổng cộng của bài hát trên giao diện người dùng.
- Tiếp theo, nó cập nhật văn bản của thanh trạng thái để hiển thị tên của bài hát đang được phát
- Cuối cùng, nó tạo một luồng mới (threading.Thread) để bắt đầu đếm thời gian của bài hát bằng cách gọi phương thức start_count với tham số là độ dài tổng cộng của bài hát.

```
def start_count(self,total_length):
    # mixer.music.get.busy=return fal se when we press stop music
    self.current_time=0
    while self.current_time<=total_length and mixer.music.get_busy():
        if self.model.paused:
            continue

        if self.model.stop:
            return

        else:
         self.show_current_time(self.current_time)
         if(self.my_slider.get()==int(self.total_length)):
            pass

         if(self.model.paused):
            pass

         elif(self.my_slider.get()==int(self.current_time)):
           slider_position=int(self.total_length)
           self.my_slider.config(to=slider_position,value=int(self.current_time))
         else:
            slider_position = int(self.total_length)
            self.my_slider.config(to=slider_position, value=int(self.my_slider.get()))
            next_time=int(self.my_slider.get()+1)
            self.show_current_time(next_time)
            self.my_slider.config(value=next_time)

         time.sleep(1)
         self.current_time+=1
```

- Phương thức này được sử dụng để bắt đầu đếm thời gian của bài hát đang phát và cập nhật thanh trượt thời gian của bài hát trên giao diện người dùng.
- Đầu tiên, nó khởi tạo biến self.current_time để đếm thời gian hiện tại của bài hát.
- Tiếp theo, nó sử dụng một vòng lặp while để tiếp tục đếm thời gian cho đến khi thời gian hiện tại vượt quá độ dài tổng cộng của bài hát hoặc khi âm nhạc dừng lại.
- Trong vòng lặp, nếu âm nhạc đang tạm dừng (self.model.paused là True), vòng lặp sẽ tiếp tục mà không làm gì cả.
- Nếu âm nhạc được dừng (self.model.stop là True), vòng lặp sẽ kết thúc.

- Nếu không, nó cập nhật thời gian hiện tại của bài hát trên giao diện người dùng bằng cách gọi phương thức show_current_time.
- Nếu thanh trượt thời gian của bài hát đã đạt đến cuối cùng, không làm gì cả.
- Nếu không, nếu thanh trượt thời gian hiện tại của bài hát trên giao diện người dùng bằng với thời gian hiện tại (self.my_slider.get() là int(self.current_time)), thanh trượt sẽ được cập nhật để di chuyển đến cuối cùng của bài hát.
- Nếu không, nó sẽ cập nhật vị trí của thanh trượt để di chuyển theo thời gian hiện tại của bài hát.
- Cuối cùng, sau mỗi giây, nó tăng thời gian hiện tại lên 1 giây và chờ một giây trước khi tiếp tục vòng lặp.

```
def update_Pause_Music(self):

    self.model.pause_Music()
    self.btn_play.configure(image=self.play_photo)
    self.statusbar['text'] = "Music pause"
```

- Phương thức này được sử dụng để tạm dừng hoặc tiếp tục phát nhạc khi người dùng nhấn vào nút phát/pause trên giao diện người dùng.
- Đầu tiên, nó gọi phương thức pause_Music từ đối tượng self.model, điều này sẽ tạm dừng hoặc tiếp tục phát nhạc, tùy thuộc vào trạng thái hiện tại của âm nhạc.
- Sau đó, nó cập nhật hình ảnh của nút phát/pause để hiển thị hình ảnh phù hợp. Nếu âm nhạc được tạm dừng, nút sẽ chuyển sang hình ảnh phát; nếu âm nhạc đang phát, nút sẽ chuyển sang hình ảnh tạm dừng.
- Cuối cùng, nó cập nhật thanh trạng thái (statusbar) trên giao diện người dùng để hiển thị thông báo "Music pause".

```
def update_Next_Music(self):
    if self.model.current_song != None and len(self.lines) > 0:
        self.model.next_Music()
        self.next_selection()

    else:
        tkinter.messagebox.showerror("Eror", "The List is Empty!")
```

- Phương thức này được sử dụng để chuyển đến bài hát kế tiếp trong danh sách phát khi người dùng nhấn vào nút "Next".
- Đầu tiên, nó kiểm tra xem có bài hát đang được phát và danh sách phát có ít nhất một bài hát không. Nếu cả hai điều kiện này đều đúng, nó tiến hành chuyển đến bài hát tiếp theo bằng cách gọi phương thức next_Music từ đối tượng self.model.
- Sau đó, nó gọi phương thức next_selection để cập nhật việc chọn bài hát tiếp theo trong danh sách phát trên giao diện người dùng.
- Nếu không có bài hát nào đang được phát hoặc danh sách phát trống, nó hiển thị một thông báo lỗi sử dụng tkinter.messagebox.showerror.

```
def update_Pause_Music(self):

    self.model.pause_Music()
    self.btn_play.configure(image=self.play_photo)
    self.statusbar['text'] = "Music pause"
```

- Phương thức này được sử dụng để chuyển đến bài hát trước đó trong danh sách phát khi người dùng nhấn vào nút "Previous".
- Đầu tiên, nó kiểm tra xem có bài hát đang được phát và danh sách phát có ít nhất một bài hát không. Nếu cả hai điều kiện này đều đúng, nó tiến hành chuyển đến bài hát trước đó bằng cách gọi phương thức previous_Music từ đối tượng self.model.
- Sau đó, nó gọi phương thức perv_selection để cập nhật việc chọn bài hát trước đó trong danh sách phát trên giao diện người dùng.

- Nếu không có bài hát nào đang được phát hoặc danh sách phát trống, nó hiển thị một thông báo lỗi sử dụng tkinter.messagebox.showerror.

```
def slide_Music(self,x):
    self.model.slider_Music()
```

- Phương thức slide_Music trong lớp audioPlayerView được gọi khi trượt thanh trượt của bài hát trên thanh trượt thời gian. Khi người dùng thay đổi vị trí của thanh trượt, giá trị mới của thanh trượt được chuyển đến phương thức slider_Music trong đối tượng self.model, để điều chỉnh thời gian phát của bài hát tương ứng. Điều này giúp người dùng có thể tua nhanh chóng đến một phần của bài hát mà họ muốn nghe.

```
def show_current_time(self,time):
    mint, sec = divmod(time, 60)
    mint = round(mint)
    sec = round(sec)
    time_formet = '{:02d}:{:02d}'.format(mint, sec)  # make the format text
    self.curren_Time_lable['text'] = "Current time  " + ' - ' + time_formet
```

- Phương thức show_current_time trong lớp audioPlayerView nhận đầu vào là time, đại diện cho thời gian hiện tại của bài hát được phát, tính bằng giây. Phương thức này chuyển đổi thời gian từ định dạng giây thành định dạng phút và giây, sau đó cập nhật nhãn hiển thị thời gian hiện tại trên giao diện người dùng. Định dạng hiển thị của thời gian là hai chữ số cho phút và giây, được hiển thị trong nhãn curren_Time_lable.

```
def show_total_time(self,time):
    mint, sec = divmod(time, 60)
    mint = round(mint)
    sec = round(sec)
    time_formet = '{:02d}:{:02d}'.format(mint, sec)  # make the format
    self.lenght_lable['text'] = "Total length  " + ' - ' + time_formet
```

- Phương thức show_total_time trong lớp audioPlayerView nhận đầu vào là time, đại diện cho tổng thời lượng của bài hát được phát, tính bằng giây. Phương thức này chuyển đổi thời gian từ định dạng giây thành định dạng phút và giây, sau đó cập nhật nhãn hiển thị tổng thời lượng trên giao diện người dùng. Định dạng hiển thị của thời lượng là hai chữ số cho phút và giây, được hiển thị trong nhãn lenght_lable.

```
def next_selection(self):
    selection_indices = self.list_song.curselection()

    # default next selection is the beginning
    next_selection = 0

    # make sure at least one item is selected
    if len(selection_indices) > 0:
        # Get the last selection, remember they are strings for
        # so convert to int
        last_selection = int(selection_indices[-1])
        next_selection = last_selection + 1
    if int(selection_indices[-1]) == self.list_song.size() - 1:
        last_selection = int(selection_indices[-1])
        next_selection = 0

    self.list_song.selection_clear(last_selection)
    self.list_song.activate(next_selection)
    self.list_song.selection_set(next_selection)
```

- Phương thức next_selection trong lớp audioPlayerView được sử dụng để chọn bài hát tiếp theo trong danh sách phát khi người dùng nhấn nút "Next". Phương thức này thực hiện các bước sau:
- Lấy chỉ số của các bài hát đang được chọn trong danh sách.
- Nếu có ít nhất một mục được chọn, phương thức tính toán chỉ số của bài hát tiếp theo dựa trên bài hát cuối cùng được chọn.
- Nếu bài hát cuối cùng được chọn là bài hát cuối cùng trong danh sách, phương thức chọn bài hát đầu tiên làm bài hát tiếp theo.

- Dọn dẹp lựa chọn trước đó, kích hoạt bài hát tiếp theo và đặt lựa chọn cho bài hát đó.

```
def pause_Music(self):
    self.paused=True
    self.play=False
    mixer.music.pause()
```

- Phương thức perv_selection trong lớp audioPlayerView thực hiện chức năng chọn bài hát trước đó trong danh sách phát khi người dùng nhấn nút "Previous". Phương thức này thực hiện các bước sau:
- Lấy chỉ số của các bài hát đang được chọn trong danh sách.
- Xác định chỉ số của bài hát trước đó dựa trên bài hát cuối cùng được chọn.
- Nếu bài hát cuối cùng được chọn là bài hát đầu tiên trong danh sách, phương thức chọn bài hát cuối cùng làm bài hát trước đó.
- Dọn dẹp lựa chọn trước đó, kích hoạt bài hát trước đó và đặt lựa chọn cho bài hát đó.

```
def update_set_Volume(self):
    self.btn_mute.configure(image=self.volume_photo)
```

- Phương thức update_set_Volume trong lớp audioPlayerView thực hiện chức năng update lại hình ảnh của nút tắt âm thanh khi người dùng giảm âm lượng xuống mức 0.

```
def select_media(self):
    global selected_media_path
    filetypes = [
        ("MP4 files", "*.mp4"),
        ("AVI files", "*.avi"),
        ("MOV files", "*.mov")
    ]

    selected_media_path = filedialog.askopenfilename(filetypes=filetypes)
    if selected_media_path:
        if selected_media_path.endswith((".mp4", ".avi", ".mov")):
            self.play_video(selected_media_path)
```

- Phương thức select_media trong lớp audioPlayerView thực hiện nhiệm vụ cho phép người dùng chọn một tệp phương tiện (video) từ hộp thoại mở tệp và sau đó chạy video nếu tệp đó hợp lệ.
- Khởi tạo biến toàn cục selected_media_path: Biến này được sử dụng để lưu đường dẫn đến tệp phương tiện được chọn bởi người dùng. Việc sử dụng biến toàn cục cho phép hàm này và các phương thức khác có thể truy cập đến đường dẫn tệp được chọn.
- Định dạng các loại tệp: Trước hết, hàm xác định loại các tệp mà người dùng có thể chọn bằng cách định nghĩa danh sách các loại tệp và các mẫu tìm kiếm tương ứng (ví dụ: MP4, AVI, MOV).
- Hiển thị hộp thoại mở tệp: Hàm askopenfilename từ thư viện filedialog được sử dụng để hiển thị hộp thoại mở tệp cho người dùng. Hộp thoại này sẽ hiển thị các tệp phương tiện dựa trên các loại tệp đã được định nghĩa trước đó.
- Xác định loại tệp được chọn: Sau khi người dùng chọn một tệp, hàm kiểm tra xem tệp đó có phải là một tệp video hợp lệ hay không bằng cách kiểm tra phần mở rộng của tên tệp.
- Chạy video nếu tệp hợp lệ: Nếu tệp được chọn là một tệp video hợp lệ (có phần mở rộng là .mp4, .avi, hoặc .mov), hàm sẽ gọi phương thức play_video với đường dẫn của tệp video làm đối số.

```python
def play_video(self, video_url):
    cap = cv2.VideoCapture(video_url)
    while True:
        ret, frame = cap.read()
        if not ret:
            break
        cv2.imshow('Video Player', frame)
        if cv2.waitKey(25) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
```

- Phương thức select_media trong lớp audioPlayerView thực hiện nhiệm vụ phát video từ một tệp video có đường dẫn được chỉ định.
- Mở tệp video: Hàm sử dụng phương thức cv2.VideoCapture từ thư viện OpenCV để mở tệp video. Đường dẫn của tệp video được truyền vào qua tham số video_url.
- Vòng lặp phát video: Hàm bắt đầu một vòng lặp vô hạn để đọc từng khung hình từ video một cách tuần tự. Trong mỗi lần lặp, hàm sử dụng phương thức cap.read() để đọc một khung hình mới từ video.
- Hiển thị khung hình: Sau khi một khung hình mới được đọc từ video, hàm sử dụng phương thức cv2.imshow để hiển thị khung hình đó trên một cửa sổ mới có tên là "Video Player".
- Thoát khỏi vòng lặp: Vòng lặp sẽ tiếp tục cho đến khi hết video hoặc người dùng nhấn phím 'q' trên bàn phím. Khi điều kiện này xảy ra, vòng lặp sẽ dừng và các cửa sổ hiển thị video sẽ đóng.
- Giải phóng tài nguyên: Sau khi kết thúc vòng lặp, hàm giải phóng tài nguyên bằng cách gọi phương thức cap.release() để đóng tệp

video và cv2.destroyAllWindows() để đóng tất cả các cửa sổ hiển thị video.

```
def on_closing(self):
    self.model.stop_Music()
    self.model.write_List_To_File(self.lines)
    self.root.destroy()
```

- Phương thức on_closing trong lớp audioPlayerView là một hàm được gọi khi người dùng đóng ứng dụng. Chức năng chính của nó là dừng phát nhạc, sau đó lưu danh sách những bài hát hiện tại vào tệp tin và cuối cùng là đóng ứng dụng.

4 Cài đặt vày chạy ứng dụng

Đối với nhà phát triển:

Trước khi chạy ứng dụng, đảm bảo bạn đã cài đặt các phụ thuộc sau:

- Python 3.x
- Tkinter
- Pygame
- OS
- Mutagen
- Time
- Threading
- OpenCV

Bạn có thể cài đặt các thư viện phụ thuộc bằng pip:

pip3 install pygame
pip3 install mutagen

pip3 install opencv-python-headless

Cách sử dụng:

Clone the Repository:https://github.com/hung573/PlayAudio$_PlayVideo$

git clone https://github.com/hung573/PlayAudio_PlayVideo

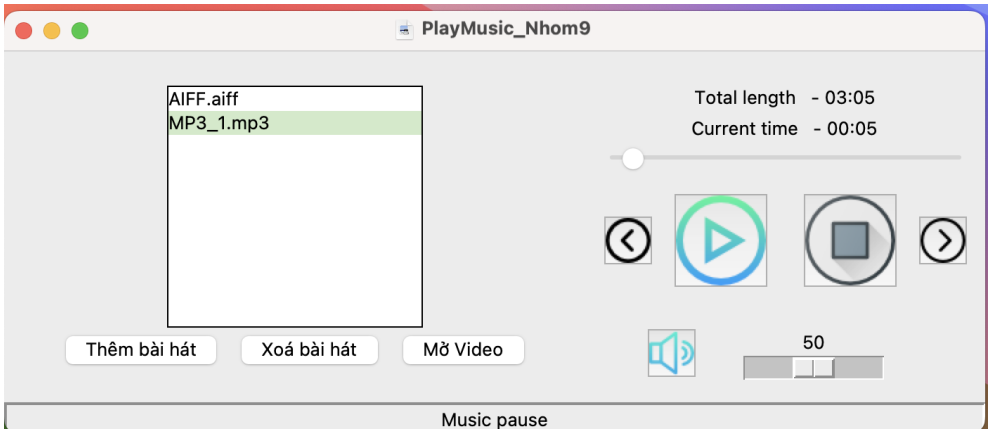
Navigate to the Project Directory:

cd PlayAudio_PlayVideo

Run the Application:

python main.py

5 Sử dụng ứng dụng



1. Thêm, xoá các bài hát vào danh sách phát và chọn video:
   - Bạn click vào nút thêm bài hát và lần lượt thêm các audio mà bạn yêu thích vào.
   - Bạn chọn bài hát mà bạn không thích ở danh sách phát và bạn click vào nút "Xoá bài hát" để bạn xoá bài hát ra khỏi danh sách phát.
   - Bạn click vào nút "Mở Video" và thực hiện chọn video có trên mấy mình để bắt đầu play chúng, khi bạn muốn tắt video bạn có thể ấn "q".
2. Play, pause, stop:

- Sau khi bạn thêm được danh sách bài hát, bạn có thể ấn nút "Play" để bắt đầu play bài đầu tiên, hoặc bạn có thể chọn bài hát mà bạn muốn "Play" có trong danh sách phát và bắt đầu play nó.
- Bạn có thể tạm ngưng bài hát bằng cách ấn vào nút "Pause" và bắt chúng play tiếp tục bằng cách ấn vào nút play.
- Bạn có thể ngưng hoàn toàn bài hát bằng cách ấn vào nút "Stop".

3. Chuyển bài:
   - Bạn có thể chuyển bài xuống bằng cách ấn vào nút "Next".
   - Bạn có thể chuyển bài xuống bằng cách ấn vào nút "Prever".
4. Điều chỉnh âm thanh, và vị trí đoạn nhạc:
   - Bạn có thể kéo tăng âm lượng hoặc giảm âm lượng bằng cách kéo lên xuống ở nút lên xuống âm lượng. Hoặc bạn có thể để âm lượng yêm lặng bằng cách ấn vào nút "mute".
   - Bạn có thể điều chỉnh vị trí đoạn nhạc bằng cách kéo qua lại ở scroll chạy thời gian của đoạn nhạc.
5. Xử lý lỗi:
   - Nếu có bất kỳ lỗi nào xảy ra trong quá trình, các thông báo lỗi sẽ được hiển thị trong một cửa sổ pop-up.

Tài liệu

[1] Python Là Gì? Tất Tần Tật Về Ngôn Ngữ Lập Trình Python: https://s.net.vn/KIrI, truy cập ngày 28/4/2024.

[2] Python là gì? Các kiến thức cần biết về lập trình Python: https://s.net.vn/l8Eq, truy cập ngày 28/4/2024.

[3] Tkinter Python là gì? Tất cả những gì bạn cần biết về Tkinter: https://www.icantech.vn/kham-pha/tkinter, truy cập ngày 28/4/2024.