OSW 팀 프로젝트 6조
2019024702 이다빈 / 2021025650 이민경 / 2021016680 이준호 / 2022089798 조사예

기본 코드 설명

```
import os
import sys
import queue
import platform

from PyQt5 import QtWidgets, QtGui, QtCore
import vlc
#import network


class MiniPlayer(QtWidgets.QMainWindow):
    """Stripped-down PyQt5-based media player class to sync with "master" video.
    """

    def __init__(self, data_queue, master=None):
        QtWidgets.QMainWindow.__init__(self, master)
        self.setWindowTitle("Mini Player")
        self.statusbar = self.statusBar()
        self.statusbar.showMessage("Ready")

        # Create a basic vlc instance
        self.instance = vlc.Instance()

        self.media = None

        # Create an empty vlc media player
        self.mediaplayer = self.instance.media_player_new()

        self.init_ui()
        self.open_file()

        self.timer = QtCore.QTimer(self)
        self.timer.setInterval(10)
        self.timer.timeout.connect(self.update_ui)

        self.data_queue = data_queue
        self.timer.start()
```

1. 네트워크 주석처리

```
from PyQt5 import QtWidgets, QtGui, QtCore
import vlc
from network import Client
```

from network **import** Client
로컬 컴퓨터의 파일 속 영상이 재생되는 것이므로 네트워크를 이용할 필요가 없어 주석처리함
(main 함수의 client 호출 부분 또한 주석처리함)

```
#     _ = Client.network("localhost", 8888, data_queue)
```

2. MiniPlayer Class 생성

class MiniPlayer(QtWidgets.QMainWindow):

PtQt5 패키지에서 비디오 플레이 화면과 위젯을 구성하는 “QtWidgets.QMainWindow” 클래스를 상속받아 사용함.

class PySide2.QtWidgets.QMainWindow([*parent=None*[, *flags=Qt.WindowFlags()*]])

```
param parent:
   PySide2.QtWidgets.QWidget

param flags:
   WindowFlags
```

Constructs a `QMainWindow` with the given `parent` and the specified widget `flags` .

`QMainWindow` sets the `Window` flag itself, and will hence always be created as a top-level widget.

3. 클래스 초기화

def __init__(self, data_queue, master=None)

vlc.Instance를 생성하여 영상을 재생한다.

그 과정에서 self.init_ui(), self.open_file() 등의 함수를 만들어 화면을 구성하고 영상을 재생한다.

4.화면 **UI** 구성

```
def init_ui(self):
    """Set up the user interface
    """
    self.widget = QtWidgets.QWidget(self)
    self.setCentralWidget(self.widget)

    # In this widget, the video will be drawn
    if platform.system() == "Darwin":  # for MacOS
        self.videoframe = QtWidgets.QMacCocoaViewContainer(0)
    else:
        self.videoframe = QtWidgets.QFrame()

    self.palette = self.videoframe.palette()
    self.palette.setColor(QtGui.QPalette.Window, QtGui.QColor(0, 0, 0))
    self.videoframe.setPalette(self.palette)
    self.videoframe.setAutoFillBackground(True)

    self.vboxlayout = QtWidgets.QVBoxLayout()
    self.vboxlayout.addWidget(self.videoframe)
    self.widget.setLayout(self.vboxlayout)
```

init_ui(self) 함수

기본적으로 보이는 비디오 화면의 색생과 프레임, 레이아웃 등 화면 UI를 구성한다.

5. 재생 영상 선택

```
def open_file(self):
    """Open a media file in a MediaPlayer
    """

    if not "HYU_ERICA.mp4":
        return

    # getOpenFileName returns a tuple, so use only the actual file name
    self.media = self.instance.media_new("HYU_ERICA.mp4")

    # Put the media in the media player
    self.mediaplayer.set_media(self.media)

    # Parse the metadata of the file
    self.media.parse()

    # Set the title of the track as the window title
    self.setWindowTitle("{}".format(self.media.get_meta(0)))

    # The media player has to be 'connected' to the QFrame (otherwise the
    # video would be displayed in it's own window). This is platform
    # specific, so we must give the ID of the QFrame (or similar object) to
    # vlc. Different platforms have different functions for this
    if platform.system() == "Linux":  # for Linux using the X Server
        self.mediaplayer.set_xwindow(int(self.videoframe.winId()))
    elif platform.system() == "Windows":  # for Windows
        self.mediaplayer.set_hwnd(int(self.videoframe.winId()))
    elif platform.system() == "Darwin":  # for MacOS
        self.mediaplayer.set_nsobject(int(self.videoframe.winId()))

    # Start playing the video as soon as it loads
    self.mediaplayer.play()
```

Open_file(self) 함수

기존 코드는 실행 시 영상 파일을 선택할 수 있는 창이 뜨는데 우리의 코드는 정해진 영상 하나만 실행하면 되므로 선택 창을 표시하는 대신 파일 경로를 바로 지정하여 영상을 실행할 수 있도록 수정하였다.

```
def open_file(self):
    """Open a media file in a MediaPlayer
    """
    dialog_txt = "Choose Media File"
    filename = QtWidgets.QFileDialog.getOpenFileName(self, dialog_txt, os.path.expanduser('~'))
    if not filename[0]:
        return

    # getOpenFileName returns a tuple, so use only the actual file name
    self.media = self.instance.media_new(filename[0])
```

dialog_txt = "Choose Media File"
filename = QtWidgets.QFileDialog.getOpenFileName(self, dialog_txt, os.path.expanduser('~'))
-> 재생할 영상 선택에 사용하는 것이므로 삭제

```
 if not filename[0]:
    return

# getOpenFileName returns a tuple, so use only the actual file name
self.media = self.instance.media_new(filename[0])
```

-> filename[0]을 "HYU_ERICA.mp4"로 지정하여 영상 바로 불러오기

6. 영상 플레이 중 화면 **UI(**상태 표시바**)**

```
def update_ui(self):
    self.update_statusbar()

    try:
        val = self.data_queue.get(block=False)
    except queue.Empty:
        return

    if val == '<':
        self.mediaplayer.set_rate(self.mediaplayer.get_rate() * 0.5)
        return
    if val == '>':
        self.mediaplayer.set_rate(self.mediaplayer.get_rate() * 2)
        return
    if val == 'P':
        self.mediaplayer.play()
        return
    if val == 'p':
        self.mediaplayer.pause()
        return
    if val == 'S':
        self.mediaplayer.stop()
        return

    val = int(val)
    if val != self.mediaplayer.get_time():
        self.mediaplayer.set_time(val)

def update_statusbar(self):
    mtime = QtCore.QTime(0, 0, 0, 0)
    time = mtime.addMSecs(self.mediaplayer.get_time())
    self.statusbar.showMessage(time.toString())
```

Update_ui 함수

Update_statusbar(self) 함수

현재 재생 중인 미디어의 시간을 밀리초 단위로 가져와서 상태 표시줄에 시간 형식으로 표시하는 역할

QtCore.QTime을 사용하여 0시 0분 0초 0밀리초로 초기화된 m.time 객체 생성한다
self.mediaplayer.get_time()을 호출하여 현재 재생 시간을 가져온다
addMSecs()를 사용하여 m.time에 get_time() 반환값을 더해 현재 재생 시간을 계산한다
time.toString()을 호출하여 계산된 시간을 문자열로 변환한다
self.statusbar.showMessage()를 사용하여 상태표시바에 변환된 시간 표시

7. Main 실행 함수
화면 크기와 실행 시간 등을 지정하여 실행한다.

```
def main():
    """Entry point for our simple vlc player
    """
    app = QtWidgets.QApplication(sys.argv)

    data_queue = queue.Queue()

    player = MiniPlayer(data_queue)
    player.show()
    player.resize(480, 480)

#    _ = Client.network("localhost", 8888, data_queue)
    sys.exit(app.exec_())


if __name__ == "__main__":
    main()
```

8. 코드 추가사항

*화면 오버레이 기능 추가

```
52          # Create an empty vlc media player
53          self.mediaplayer = self.instance.media_player_new()
54
55          self.init_ui()
56          self.open_file()
57
58          self.timer = QtCore.QTimer(self)
59          self.timer.setInterval(10)
60          self.timer.timeout.connect(self.update_ui)
61
62          self.data_queue = data_queue
63          self.timer.start()
```

추가 전

```
        # Create an empty vlc media player
        self.mediaplayer = self.instance.media_player_new()

        self.init_ui()
        self.open_file()

        self.timer = QtCore.QTimer(self)
        self.timer.setInterval(10)
        self.timer.timeout.connect(self.update_ui)

        self.data_queue = data_queue

        # Set window flags to stay on top
        self.setWindowFlags(QtCore.Qt.WindowStaysOnTopHint)
        self.timer.start()
```

추가 후