Manual Técnico del Proyecto: Vector Sound

Juan David Torres Avila y Victor Manuel Rojas

15/11/2024

1 Introducción

Versión de QT creator utilizada: 6.8.0
Este documento técnico describe la estructura, funcionalidad y uso de las principales bibliotecas y componentes del proyecto **Vector Sound**. Este es un reproductor multimedia basado en *Qt*, que incluye funcionalidades para manejar listas de reproducción, reproducción de audio/video y controles de tiempo y volumen.

2 Componentes Principales

El código fuente utiliza varias clases y bibliotecas de *Qt*. A continuación, se explican los componentes clave:

2.1 Bibliotecas Utilizadas

- `<QMainWindow>`: Proporciona la estructura base de la ventana principal del proyecto.
- `<QMediaPlayer>`: Maneja la reproducción de medios. Se utiliza para reproducir tanto audio como video.
- `<QAudioOutput>`: Controla la salida de audio del reproductor.
- `<QLabel>`: Permite mostrar etiquetas de texto e imágenes en la interfaz gráfica.

- `<QSlider>`: Implementa barras deslizantes para ajustar el volumen y navegar por la duración de los archivos multimedia.
- `<QTimer>`: Proporciona temporizadores para actualizar la interfaz en intervalos regulares.
- `<QFileDialog>`: Abre cuadros de diálogo para seleccionar archivos multimedia.
- `<QPixmap>`: Muestra imágenes, como carátulas de álbum, en la interfaz gráfica.

```
#include <QMainWindow>
#include <QtMultimedia>
#include <QtMultimediaWidgets>
#include <QtCore>
#include <QtWidgets>
#include <QtGui>
```

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QTimer>
#include <QFileDialog>
```

2.2 Componentes del Proyecto

- **Clase** `MainWindow`: Es la clase principal que controla la interfaz gráfica y las funcionalidades del reproductor.
- `Player (QMediaPlayer)`: Maneja la reproducción de los archivos multimedia.
- `audioOutput (QAudioOutput)`: Controla el volumen y el estado de silencio del reproductor.
- `playlistWidget`: Una lista gráfica que muestra los archivos de la lista de reproducción.
- `videoWidget`: Un contenedor para mostrar videos.
- `albumArtLabel`: Muestra una imagen por defecto o carátulas de álbum.
- `timeUpdateTimer`: Temporizador que actualiza el tiempo actual y la barra de progreso.

3 Funciones Clave

Descripción del Código

Fragmento de Código

A continuación se describe un fragmento de código que configura un temporizador (‘QTimer‘) para actualizar el tiempo de reproducción en un reproductor multimedia.

```
// Configurar timer para actualizaci n del tiempo
timeUpdateTimer = new QTimer(this);
timeUpdateTimer->setInterval(1000);
connect(timeUpdateTimer, &QTimer::timeout, this, &
   MainWindow::updateCurrentTimeDisplay);
timeUpdateTimer->start();
```

1. Creación del Temporizador

```
timeUpdateTimer = new QTimer(this);
```

Se crea una nueva instancia de ‘QTimer‘ con ‘this‘ como contexto, lo que asegura que el temporizador esté vinculado a la instancia actual de ‘MainWindow‘.

2. Configuración del Intervalo

```
timeUpdateTimer->setInterval(1000);
```

Se establece el intervalo del temporizador en 1000 milisegundos (1 segundo), lo que define la frecuencia con la que se ejecutará su función asociada.

3. Conexión de Señal y Slot

```
connect(timeUpdateTimer, &QTimer::timeout, this, &
   MainWindow::updateCurrentTimeDisplay);
```

La señal ‘timeout()‘ del temporizador se conecta al slot ‘updateCurrentTimeDisplay()‘ de la clase ‘MainWindow‘. Esto asegura que cada vez que se alcance el intervalo del temporizador, se llame a la función encargada de actualizar el tiempo actual de reproducción.

4. Inicio del Temporizador

```
timeUpdateTimer->start();
```

Se activa el temporizador, iniciando la emisión de señales periódicas cada segundo.

Función del Slot: `updateCurrentTimeDisplay()`

El slot ‘updateCurrentTimeDisplay()‘ es llamado cada segundo para sincronizar los elementos de la interfaz relacionados con el tiempo de reproducción:

- Actualiza la barra deslizante de duración (‘horizontalSlider$_{D}uration$‘)$conlaposiciónactualdelmedio$.

Resumen

Este fragmento configura un temporizador utilizando la clase ‘QTimer‘ para actualizar elementos de la interfaz gráfica de manera periódica. La conexión entre el temporizador y el slot garantiza que los controles relacionados con el tiempo se mantengan sincronizados con el estado de reproducción.

“‘latex [12pt]article [utf8]inputenc listings xcolor

Fragmento de Código

El siguiente fragmento de código gestiona la selección de archivos multimedia y la configuración de la interfaz gráfica para reproducirlos.

Función 1: `on_action_Open_triggered()`

```
// Funcion para abrir el buscador de archivos para
   seleccionar un archivo multimedia
void MainWindow::on_action_Open_triggered()
{
    QString fileName = QFileDialog::getOpenFileName(this
       , tr("Open Media"), "",
    tr("Multimedia Files (*.mp3 *.mp4 *.wav *.avi)"));
    if (fileName.isEmpty())
        return;

    bool isVideo = fileName.endsWith(".mp4") || fileName
       .endsWith(".avi");
    displayMedia(fileName, isVideo);

    // Reset the play button to "play" icon when a new
       file is opened
    ui->pushButton_Play->setChecked(false);
    ui->pushButton_Play->setIcon(QIcon(":/Icons/play.png
       "));
}
```

Función 2: displayMedia()

```
// Funcion para mostrar el archivo multimedia
void MainWindow::displayMedia(const QString& fileName,
   bool isVideo)
{
    if (isVideo)
    {
        videoWidget->show();
        albumArtLabel->hide();
        Player->setVideoOutput(videoWidget);
    }
    else
    {
        videoWidget->hide();
        albumArtLabel->show();
```

```
14          QPixmap albumArt(":/images/default.png"); //
               Placeholder
15          albumArtLabel->setPixmap(albumArt.scaled(
               albumArtLabel->size(), Qt::KeepAspectRatio));
16          Player->setVideoOutput(nullptr);
17      }
18      Player->setSource(QUrl::fromLocalFile(fileName));
19      mediaTitleLabel->setText(QFileInfo(fileName).
           baseName());
20  }
```

Descripción del Código

1. `on_action_Open_triggered()`

- Propósito: Abre un cuadro de diálogo para seleccionar un archivo multimedia (audio o video).
- Funcionamiento:
  - Utiliza `QFileDialog::getOpenFileName()` para permitir al usuario seleccionar un archivo. Solo admite extensiones como `.mp3`, `.mp4`, `.wav`, y `.avi`.
  - Si no se selecciona un archivo, la función termina inmediatamente.
  - Verifica si el archivo seleccionado es de video mediante la extensión:

```
1           bool isVideo = fileName.endsWith(".mp4")
                || fileName.endsWith(".avi");
```

  - Llama a `displayMedia()` para configurar la interfaz de acuerdo al tipo de archivo.
  - Restablece el botón de reproducción para que muestre el ícono de "play".

2. `displayMedia()`

- Propósito: Configura la interfaz gráfica para reproducir el archivo multimedia seleccionado.

- Funcionamiento:
  - Si el archivo es de video:
    - Muestra el widget de video (`videoWidget`).
    - Oculta la etiqueta de carátula de álbum (`albumArtLabel`).
    - Configura el reproductor (`Player`) para enviar el video al widget de video.
  - Si el archivo es de audio:
    - Oculta el widget de video y muestra la carátula del álbum.
    - Escala la carátula para ajustarla al tamaño disponible manteniendo la proporción.
  - Configura el reproductor para usar el archivo seleccionado como fuente:

```
1         Player->setSource(QUrl::fromLocalFile(
              fileName));
```

  - Actualiza la etiqueta de título del archivo con su nombre base:

```
1         mediaTitleLabel->setText(QFileInfo(
              fileName).baseName());
```

Resumen

Estas funciones permiten:

- Seleccionar un archivo multimedia mediante un cuadro de diálogo.
- Configurar la interfaz gráfica para reproducir archivos de audio o video.
- Preparar el reproductor y restablecer el estado del botón de reproducción.

Este diseño asegura una experiencia de usuario clara y adaptable tanto para audio como para video.

4 Botones

4.1 `playSong()`

Inicia la reproducción del archivo seleccionado en la lista de reproducción.

4.2 `pauseSong()`

Pausa la reproducción del archivo actual.

4.3 `stopSong()`

Detiene completamente la reproducción actual.

4.4 `on_horizontalSlider_Duration_sliderReleased()`

Actualiza la posición del archivo multimedia en reproducción cuando el usuario interactúa con la barra de progreso.

4.5 `on_pushButton_Volume_clicked()`

Alterna el estado de silencio del reproductor.

5 Componentes de la Interfaz Gráfica

- **Botones de Control**:
  - `pushButton_Play`: Reproduce o pausa el archivo.
  - `pushButton_Stop`: Detiene la reproducción.
  - `pushButton_Volume`: Muestra u oculta la barra de volumen.
- **Barras Deslizantes**:
  - `horizontalSlider_Duration`: Controla la posición de reproducción.
  - `verticalSlider_Volume`: Ajusta el volumen del audio.

6 Conclusión

Este manual técnico proporciona una visión detallada de los componentes y bibliotecas utilizados en el proyecto. Los desarrolladores pueden utilizar esta información para modificar o expandir la funcionalidad del reproductor.