Mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include<QProcess>
#include<QAction>
#include<QMenu>
#include<QMediaPlayer>
#include<QMediaPlaylist>
#include<QSystemTrayIcon>
#include<QMouseEvent>
#include<QCloseEvent>
#include<QVideoWidget>
#include<QDesktopServices>
#include<QDir>
#include<QFileDialog>
#include<QMediaObject>
#include<QAudioOutput>
#include"dialog.h"

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
private slots:
    void positionChanged(qint64);
    void durationChanged(qint64);
    void playTo(int,int);
    void updateSongList(int);

    void showOrHideSongList();
    void importSongs();
    void playLast();
    void playOrPause();
    void playNext();
```

```
        void plusSound();
        void reduceSound();

        void setPlaybackModel1();
        void setPlaybackModel2();
        void setPlaybackModel3();
        void setPlaybackModel4();

        void support();
        void aboutUs();

        void lyric();
        void deleteSong();
        void addSong();
        void luzhi();

        void setPosition(int);

        void iconActivated(QSystemTrayIcon::ActivationReason);

        void on_toolButton_clicked();

private:
        Ui::MainWindow * ui;
        void createContextMenu();
        void createSysytemTrayIcon();
        QPoint dragPos;
        int volume;

        QAction * restoreAction;
        QAction * quitAction;
        QAction * seperatorAction1;
        QAction * seperatorAction2;
        QAction * seperatorAction3;
        QAction * seperatorAction4;
        QAction * seperatorAction5;
        QAction * seperatorAction6;
        QAction * seperatorAction7;

        QMenu *trayContextMenu;

        QMediaPlayer *player;
        QMediaPlaylist * playList;
        QSystemTrayIcon *trayIcon;
        bool b;
        int z;
//      QMediaObject *media_object;
protected:
        void mouseMoveEvent(QMouseEvent *);
        void mousePressEvent(QMouseEvent *);
        void closeEvent(QCloseEvent *);
public:
        void con();
};

#endif // MAINWINDOW_H
```

Mainwindow.cpp

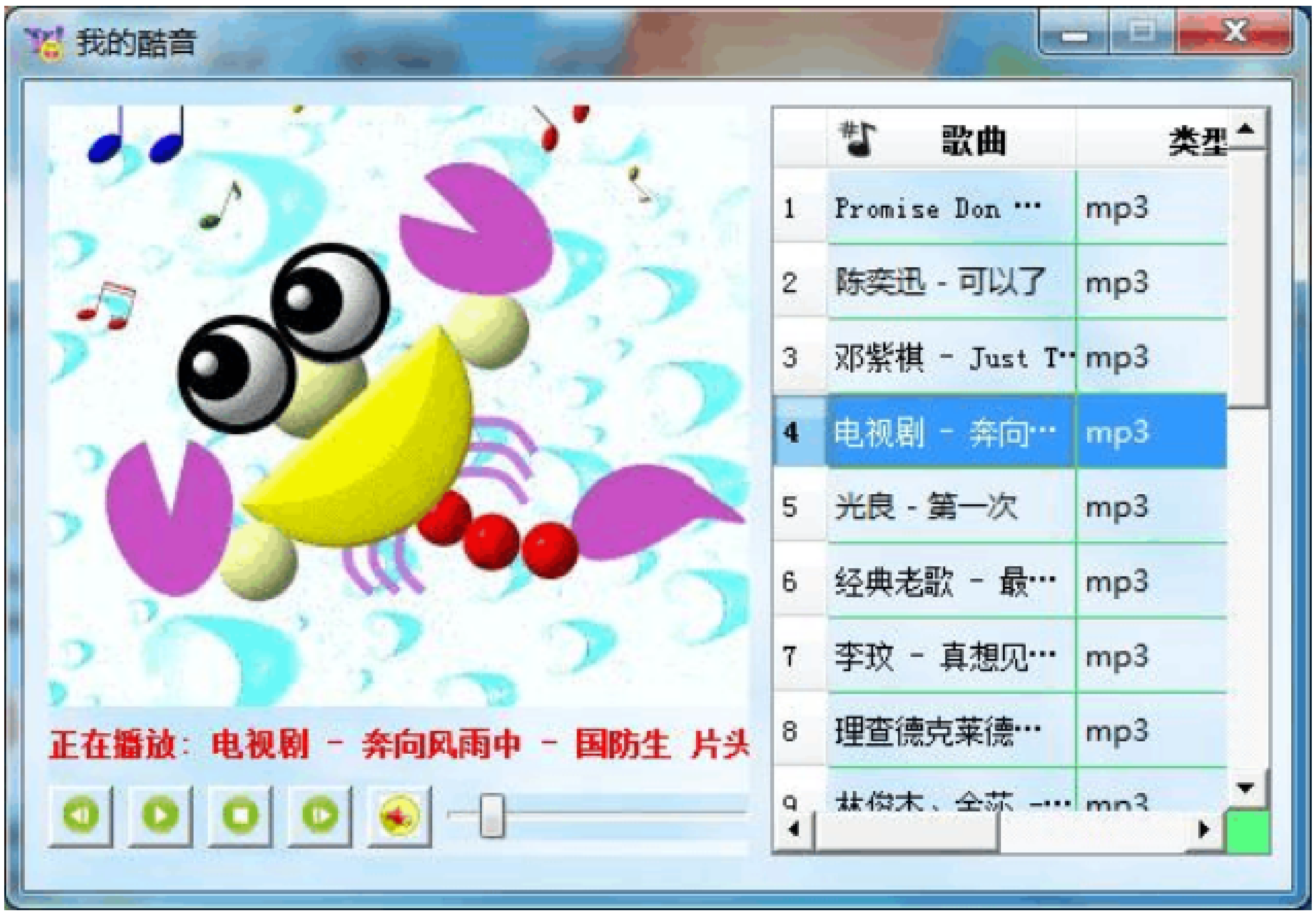
```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include<QMessageBox>
#include<QDebug>
#include<QFile>
#include<QIODevice>
#include<QMovie>
#include<QMultimedia>
MainWindow::MainWindow(QWidget *parent) :
        QMainWindow(parent),
        ui(new Ui::MainWindow)
{
        ui->setupUi(this);

        this->setFixedSize(this->width(),this->height());
        ui->horizontalSlider->setRange(0,0);

        ui->label_2->show();

        ui->textEdit->hide();

        QMovie *movie=new QMovie(":/new/prefix1/images/ 搞笑.gif");
        ui->label_2->setMovie(movie);
        movie->start();

        volume=80;

        createContextMenu();
        createSysytemTrayIcon();
        playList=new QMediaPlaylist;
        playList->setPlaybackMode(QMediaPlaylist::Loop);
        player=new QMediaPlayer;
        player->setPlaylist(playList);
        player->setVolume(volume);
        b=false;
        z=-1;
        con();

}
void MainWindow::con()
{

connect(ui->horizontalSlider,SIGNAL(sliderMoved(int)),this,SLOT(setPosition(int)
));

connect(ui->tableWidget,SIGNAL(cellClicked(int,int)),this,SLOT(playTo(int,int)));


connect(player,SIGNAL(positionChanged(qint64)),this,SLOT(positionChanged(qint
64)));

connect(player,SIGNAL(durationChanged(qint64)),this,SLOT(durationChanged(qint
64)));

connect(playList,SIGNAL(currentIndexChanged(int)),this,SLOT(updateSongList(int
)));

        connect(ui->action_SongList, SIGNAL(triggered()), this,
SLOT(showOrHideSongList()));
        connect(ui->action_Import, SIGNAL(triggered()), this, SLOT(importSongs()));
        connect(ui->action_Last, SIGNAL(triggered()), this, SLOT(playLast()));
        connect(ui->action_Play, SIGNAL(triggered()), this, SLOT(playOrPause()));
        connect(ui->action_Stop, SIGNAL(triggered()),  player, SLOT(stop()));
        connect(ui->action_Next, SIGNAL(triggered()), this, SLOT(playNext()));
        connect(ui->action_SoundPlus, SIGNAL(triggered()), this,
SLOT(plusSound()));
        connect(ui->action_SoundReduce, SIGNAL(triggered()), this,
SLOT(reduceSound()));
        connect(ui->action_mode1, SIGNAL(triggered()), this,
SLOT(setPlaybackModel1()));
        connect(ui->action_mode2, SIGNAL(triggered()), this,
SLOT(setPlaybackModel2()));
```

```
    connect(ui->action_mode3, SIGNAL(triggered()), this,
SLOT(setPlaybackModel3()));
    connect(ui->action_mode4, SIGNAL(triggered()), this,
SLOT(setPlaybackModel4()));
    connect(ui->action_support, SIGNAL(triggered()), this, SLOT(support()));
    connect(ui->action_about, SIGNAL(triggered()), this, SLOT(aboutUs()));
    connect(ui->action_Quit, SIGNAL(triggered()), this, SLOT(close()));
    connect(ui->action_lyric,SIGNAL(triggered(bool)),this,SLOT(lyric()));
    connect(ui->action_delete,SIGNAL(triggered(bool)),this,SLOT(deleteSong()));
    connect(ui->action_add,SIGNAL(triggered(bool)),this,SLOT(addSong()));
    connect(ui->action_luzhi,SIGNAL(triggered(bool)),this,SLOT(luzhi()));

    connect(ui->Last,SIGNAL(clicked(bool)),this,SLOT(playLast()));
    connect(ui->Play,SIGNAL(clicked(bool)),this,SLOT(playOrPause()));
    connect(ui->Stop,SIGNAL(clicked(bool)), player,SLOT(stop()));
    connect(ui->Next,SIGNAL(clicked(bool)),this,SLOT(playNext()));

}
void MainWindow::createContextMenu()
{
    seperatorAction1=new QAction(this);
    seperatorAction1->setSeparator(true);
    seperatorAction2=new QAction(this);
    seperatorAction2->setSeparator(true);
    seperatorAction3=new QAction(this);
    seperatorAction3->setSeparator(true);
    seperatorAction4=new QAction(this);
    seperatorAction4->setSeparator(true);
    seperatorAction5=new QAction(this);
    seperatorAction5->setSeparator(true);
    seperatorAction6=new QAction(this);
    seperatorAction6->setSeparator(true);
    seperatorAction7=new QAction(this);
    seperatorAction7->setSeparator(true);

    addAction(ui->action_SongList);
    addAction(ui->action_Import);
    addAction(seperatorAction7);
    addAction(ui->action_lyric);
    addAction(ui->action_luzhi);
    addAction(seperatorAction6);
    addAction(ui->action_delete);
    addAction(ui->action_add);
    addAction(seperatorAction1);
    addAction(ui->action_Last);
    addAction(ui->action_Play);
    addAction(ui->action_Stop);
    addAction(ui->action_Next);
    addAction(seperatorAction2);
    addAction(ui->action_mode1);
    addAction(ui->action_mode2);
    addAction(ui->action_mode3);
    addAction(ui->action_mode4);
    addAction(seperatorAction3);
    addAction(ui->action_SoundPlus);
    addAction(ui->action_SoundReduce);
    addAction(seperatorAction4);
    addAction(ui->action_support);
    addAction(ui->action_about);
    addAction(seperatorAction5);
    addAction(ui->action_Quit);

    setContextMenuPolicy(Qt::ActionsContextMenu);
}
void MainWindow::createSysytemTrayIcon()
{
    trayIcon=new QSystemTrayIcon(this);
    trayIcon->setIcon(QIcon(tr(":/new/prefix1/images/icon.ico")));
    trayIcon->setToolTip(tr(" 酷音听听  -  音乐播放器 "));

    restoreAction=new QAction(tr(" 打开主面板 "), this);
    connect(restoreAction, SIGNAL(triggered()), this, SLOT(show()));

    quitAction=new QAction(tr(" 退出 "), this);
    connect(quitAction, SIGNAL(triggered()), qApp, SLOT(quit()));

    trayContextMenu=new QMenu(this);
    trayContextMenu->addAction(ui->action_SongList);
    trayContextMenu->addAction(ui->action_Import);
    trayContextMenu->addSeparator();
    trayContextMenu->addAction(ui->action_lyric);
    trayContextMenu->addAction(ui->action_luzhi);
    trayContextMenu->addSeparator();
    trayContextMenu->addAction(ui->action_delete);
    trayContextMenu->addAction(ui->action_add);
    trayContextMenu->addSeparator();
    trayContextMenu->addAction(ui->action_Last);
    trayContextMenu->addAction(ui->action_Play);
    trayContextMenu->addAction(ui->action_Stop);
    trayContextMenu->addAction(ui->action_Next);
    trayContextMenu->addSeparator();
    trayContextMenu->addAction(ui->action_mode1);
    trayContextMenu->addAction(ui->action_mode2);
    trayContextMenu->addAction(ui->action_mode3);
    trayContextMenu->addAction(ui->action_mode4);
    trayContextMenu->addSeparator();
    trayContextMenu->addAction(ui->action_SoundPlus);
    trayContextMenu->addAction(ui->action_SoundReduce);
    trayContextMenu->addSeparator();
    trayContextMenu->addAction(ui->action_support);
    trayContextMenu->addAction(ui->action_about);
    trayContextMenu->addSeparator();
    trayContextMenu->addAction( restoreAction);
    trayContextMenu->addAction(quitAction);
    trayIcon->setContextMenu(trayContextMenu);

    trayIcon->show();
    connect(trayIcon, SIGNAL(activated(QSystemTrayIcon::ActivationReason)), \
            this, SLOT(iconActivated(QSystemTrayIcon::ActivationReason)));
}
void MainWindow::iconActivated(QSystemTrayIcon::ActivationReason reason)
{
    switch(reason) {
        case QSystemTrayIcon::DoubleClick:
        case QSystemTrayIcon::Trigger:
            if(this->isVisible()==true) {
                ;
            } else {
                this->show();
                this->activateWindow();
            }
            break;
        default:
            break;
        }
}
void MainWindow:: closeEvent(QCloseEvent *event)
{
    if( trayIcon->isVisible()) {
            hide();
            trayIcon->showMessage(tr("提示"), tr(" 酷音听听将在后台运行  "));
            event->ignore();
        } else {
            event->accept();
        }
}
void MainWindow:: mouseMoveEvent(QMouseEvent *event)
{
    if(event->button()==Qt::LeftButton) {
            dragPos=event->globalPos()-frameGeometry().topLeft();
            event->accept();
        }
}
void MainWindow:: mousePressEvent(QMouseEvent *event)
{
    if(event->button()==Qt::LeftButton) {
        dragPos=event->globalPos()-frameGeometry().topLeft();
        event->accept();
    }
}

void MainWindow::positionChanged(qint64 position)
{
    ui->horizontalSlider->setValue(position);
}
void MainWindow::durationChanged(qint64 duration)
{
    ui->horizontalSlider->setRange(0,duration);
}
void MainWindow::updateSongList(int i)
{
    ui->tableWidget->selectRow(i);
    ui->label->setText(tr("正在播放 : %1").arg(ui->tableWidget->item(i,
0)->text()));
    ui->label->setStyleSheet("color:red;font-weight:bold");
}
void MainWindow::showOrHideSongList()
{
    if(ui->tableWidget->isHidden()) {
        ui->tableWidget->show();
    } else {
        ui->tableWidget->hide();
    }
}
void MainWindow::playTo(int i,int)
{
    playList->setCurrentIndex(i);
    player->play();
    b=true;
    z=i;
   // qDebug()<<"pppppppppppp";
}
void MainWindow::playLast()
{
    int currentIndex=playList->currentIndex();
    if(--currentIndex<0) currentIndex=0;
    playList->setCurrentIndex(currentIndex);
    player->play();
}
void MainWindow::playOrPause()
{
    if(ui->Play->text()==tr(" 播放 ")) {
        player->play();
        ui->Play->setText(tr("暂停"));
    } else {
        player->pause();
        ui->Play->setText(tr("播放"));
    }
}
void MainWindow::playNext()
{
    int currentIndex=playList->currentIndex();
    if(++currentIndex== playList->mediaCount()) currentIndex=0;
    playList->setCurrentIndex(currentIndex);
    player->play();
}
void MainWindow::plusSound()
{
    volume+=5;
    if(volume>=100) {
        volume=100;
        ui->action_SoundPlus->setEnabled(false);
    }
    player->setVolume(volume);

    if(! ui->action_SoundReduce->isEnabled())
        ui->action_SoundReduce->setEnabled(true);

}
```

```
void MainWindow::reduceSound()
{
    volume-=5;
    if(volume<=0) {
        volume=0;
        ui->action_SoundReduce->setEnabled(false);
    }
    player->setVolume(volume);

    if(! ui->action_SoundPlus->isEnabled())
        ui->action_SoundPlus->setEnabled(true);
}
void MainWindow::setPosition(int i)
{
    player->setPosition(i);
}
void MainWindow::setPlaybackModel1()
{
    playList->setPlaybackMode(QMediaPlaylist::Loop);
}
void MainWindow::setPlaybackModel2()
{
    playList->setPlaybackMode(QMediaPlaylist::Random);

}
void MainWindow::setPlaybackModel3()
{
    playList->setPlaybackMode(QMediaPlaylist::CurrentItemInLoop);
}

void MainWindow::setPlaybackModel4()
{
    playList->setPlaybackMode(QMediaPlaylist::Sequential);
}

void MainWindow::support()
{
    QMessageBox::about(this, tr("关于酷音听听 "), \
                              tr("酷音听听，感谢您的使用！ "));
}

void MainWindow::aboutUs()
{
    const QUrl url("http://www.baidu.com");
    QDesktopServices::openUrl(url);
}

void MainWindow::lyric()
{
    QString str=ui->tableWidget->item(z,0)->text();
  //    QString stra("11111");
   // qDebug()<<stra;
  //    qDebug()<<str;
   // qDebug()<<z;
    if(ui->action_lyric->text()==" 显示歌词 ")
    {
        QFile file("C:/KuGou/Lyric/"+str+".krc");
        //qDebug()<<file.fileName();
        if(!file. open(QIODevice::ReadOnly|QIODevice::Text))
        {
            QMessageBox::warning(this, tr("提示"), tr(" 当前目录下未找到歌词
文件"));
        }
        else
        {
            QTextStream in(&file);
            QString result=in.readAll();
            ui->textEdit->setText(result);
            ui->textEdit->show();
            //ui->graphicsView->hide();
            ui->label_2->hide();
        }
        ui->action_lyric->setText("隐藏歌词 ");
    }
    else
    {
        ui->textEdit->hide();
       // ui->graphicsView->show();
        ui->label_2->show();
        ui->action_lyric->setText("显示歌词 ");
    }
}

void MainWindow::deleteSong()
{
    if (QMessageBox::Yes == QMessageBox::question(this,
        tr("提示"),
        tr("您确定删除这首歌吗？  "),
                QMessageBox::Yes | QMessageBox::No,
                QMessageBox::Yes))
        {
                ui->tableWidget->removeRow(z);
                playList->removeMedia(z);
        }
}

void MainWindow::addSong()
{
        QString path=QFileDialog::getOpenFileName(this,"增添歌曲
",QDir::homePath(),"*.mp3");
        path=QDir::toNativeSeparators(path);
        playList->addMedia(QUrl::fromLocalFile(path));
        QString fileName=path.split("\\").last();
        int rownum=ui->tableWidget->rowCount();
        ui->tableWidget->insertRow(rownum);
        ui->tableWidget->setItem(rownum, 0, new
QTableWidgetItem(fileName.split(".").front()));
        ui->tableWidget->setItem(rownum, 1, new
QTableWidgetItem(fileName.split(".").last()));
}
void MainWindow::luzhi()
{
        Dialog *dialog=new Dialog;
        dialog->show();
}
void MainWindow::importSongs()
{
        QString name=QDir::homePath();
        QStringList pathList=QFileDialog::getOpenFileNames(this,tr("选择歌曲
"),name,tr("*.mp3"));
        for(int i=0;i<pathList.size();i++)
        {
                QString path=QDir::toNativeSeparators(pathList.at(i));
                if(!path.isEmpty())
                {
                        playList->addMedia(QUrl::fromLocalFile(path));
                        QString fileName=path.split("\\").last();
                        int rownum=ui->tableWidget->rowCount();
                        ui->tableWidget->insertRow(rownum);
                        ui->tableWidget->setItem(rownum, 0, new
QTableWidgetItem(fileName.split(".").front()));
```

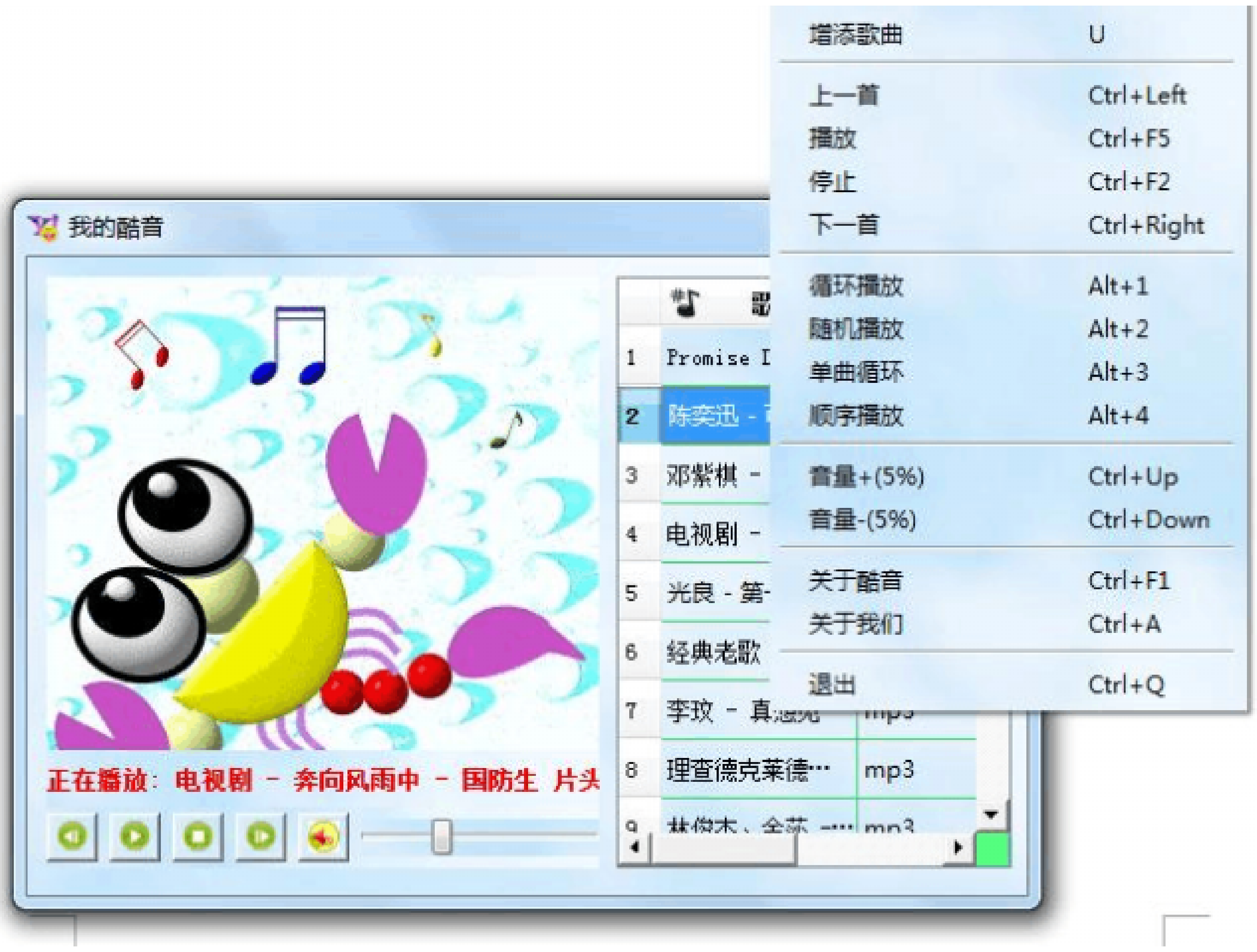
```
            ui->tableWidget->setItem(rownum, 1, new
QTableWidgetItem(fileName.split(".").last()));
        }
        //ui->tableWidget->setItem(rownum, 2, new QTableWidgetItem(path));
    }

}
MainWindow::~ MainWindow()
{
    deleteui;
}
void MainWindow::on_toolButton_clicked()
{
    if(b)
    {
        player->setVolume(0);

    }
    else {
        player->setVolume(volume);
    }
    b=!b;
}
```

运行截图：



正在播放：电视剧 － 奔向风雨中 － 国防生 片头

	歌曲	类型
1	Promise Don …	mp3
2	陈奕迅 - 可以了	mp3
3	邓紫棋 － Just T…	mp3
4	电视剧 － 奔向…	mp3
5	光良 - 第一次	mp3
6	经典老歌 － 最…	mp3
7	李玟 － 真想见…	mp3
8	理查德克莱德…	mp3
9	林俊杰、金莎 -…	mp3



正在播放：电视剧 － 奔向风雨中 － 国防生 片头