山西大学

课程设计报告

题　　目：　基于 QT的音乐播放器设计

系　　别：　软件学院　班　　级：　1402　，1403

姓　　名：　闫钰，张天峰，赵曙华，赵轩健，赵炀，周山

设计时间：　2016 年 12 月 19 日----2016　年 12 月 30 日

公司名称：　　上海杰普软件科技有限公司

一. 需求分析

1.1 编写目的

为明确软件需求、安排项目与进度、组织软件开发与测试、编写用户手册，撰写本文档。

本文档用于项目经理、设计人员、开发人员、测试人员等参考。

1.2 业务背景

在音乐播放器成为人们了广泛应用的计算机应用软件之后，人们的日常生活被极大地丰富，越来越多的人们开始使用音乐播放器来对计算机设备上的音乐文件播放，而互联网上的音乐目前也正以极大极丰富的产量在生产之中，每天都会有数以万计的新的音乐产生，而作为为用户需求考虑的软件开发人员来说，设计并实现一款音乐播放器应用程序则成为一种应用需求。目前，互联网上已经拥有大量的音乐播放器，这些播放器不仅使用方便快捷，而且往往拥有强大的功能，并且拥有十分友好的用户交互界面，广受用户的好评。但是用 Qt Creator 开发的音乐播放器小巧而功能齐全，方便移植到嵌入式平台下或其他平台下，只需一次编译就可在不同平台下运行播放。本文正是在考虑目前互联网上使用量较多的音乐播放器后，试图通过 Qt 开发出一款适合个人使用的音乐播放器软件，能够小巧方便的运行的个人计算机上。

1.3 项目目标

本项目的目标是开发一个可以播放主流的音乐文本格式的播放器。设计的主要实现功能是播放 MP3等格式的音乐文件，并且能控制播放，暂停，停止，音量控制，选择上一曲，选择下一曲，更改皮肤，歌曲列表文件的管理操作，在线播放，读取存储卡播放等多种播放控制，界面简明，操作合理。

1.4　参考资料

参考资料的名称、作者、版本、编写日期。

【1】《24 小时学通 QT编程》　著：Daiel Solin

【2】《KDE2/QT编程宝典》　电子工业出版社

【4】《软件工程实践教程》　　赵池龙等编，电工出版社出版

1.5　名词定义

文档中可能使用的各种术语或名词的定义与约定，大家可以根据需要删减。

编号	属于	说明
1.	QTCreator	图形界面设计
2.	Signal	触发信号
3.	Slot	接受信号的槽函数
4.	Connect	实现触发信号和槽函数的连接

1.6　运行环境

硬件环境：笔记本电脑一台

软件环境： Microsoft Windows xp　以上

开发工具： Qt Creator 5.0

调试工具：笔记本电脑

二．概要设计

2.1　整体功能分析

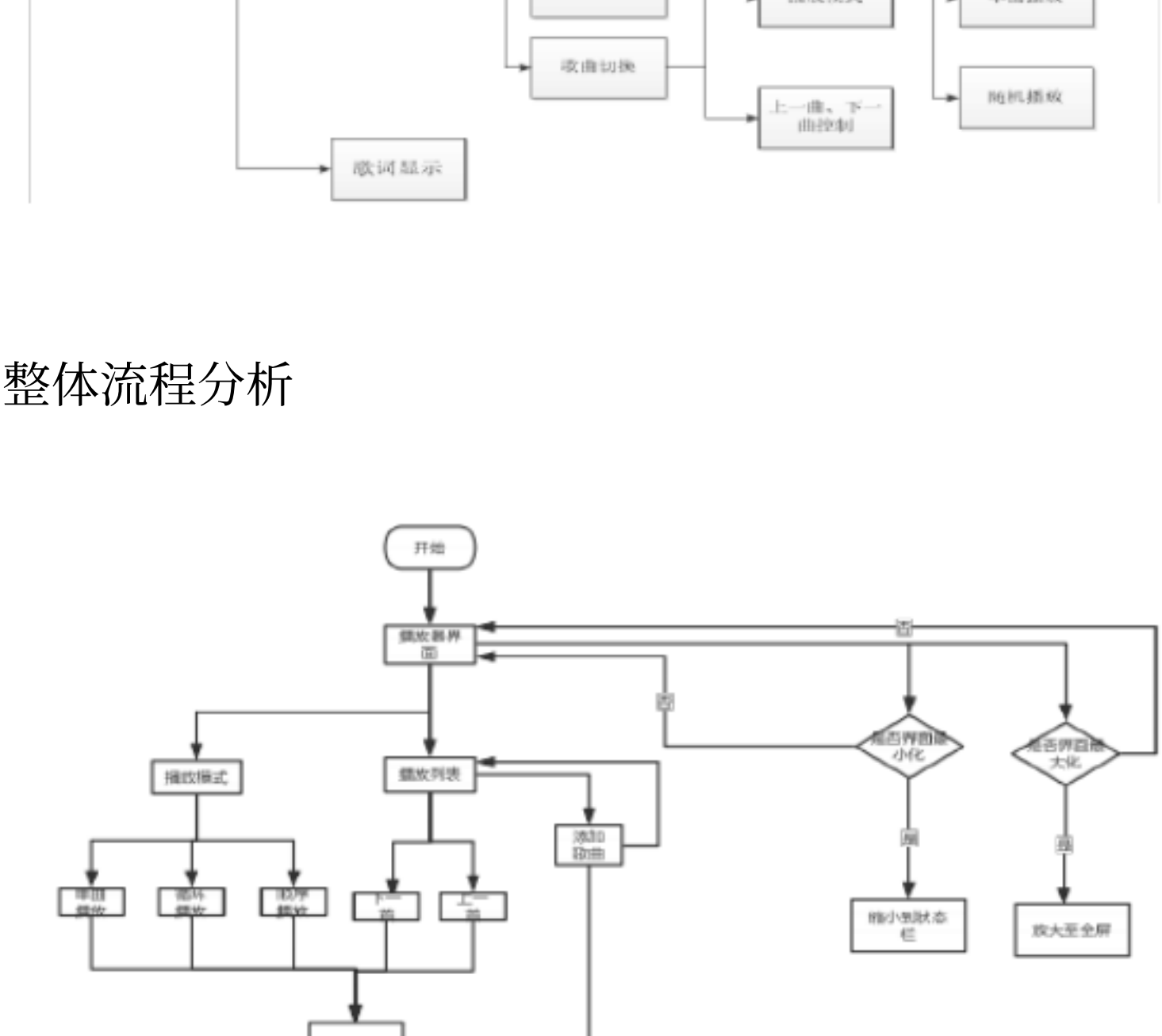
根据项目，我们可以获得项目系统的基本需求，　以下从不同角度来描述系统的需求。

系统的功能需求我们分为四部分来概括，　即界面设计、 音乐播放、 播放列表以及歌词显示。以下用表格及系统整体框架图分别进行描述：

2.1.1 系统功能表

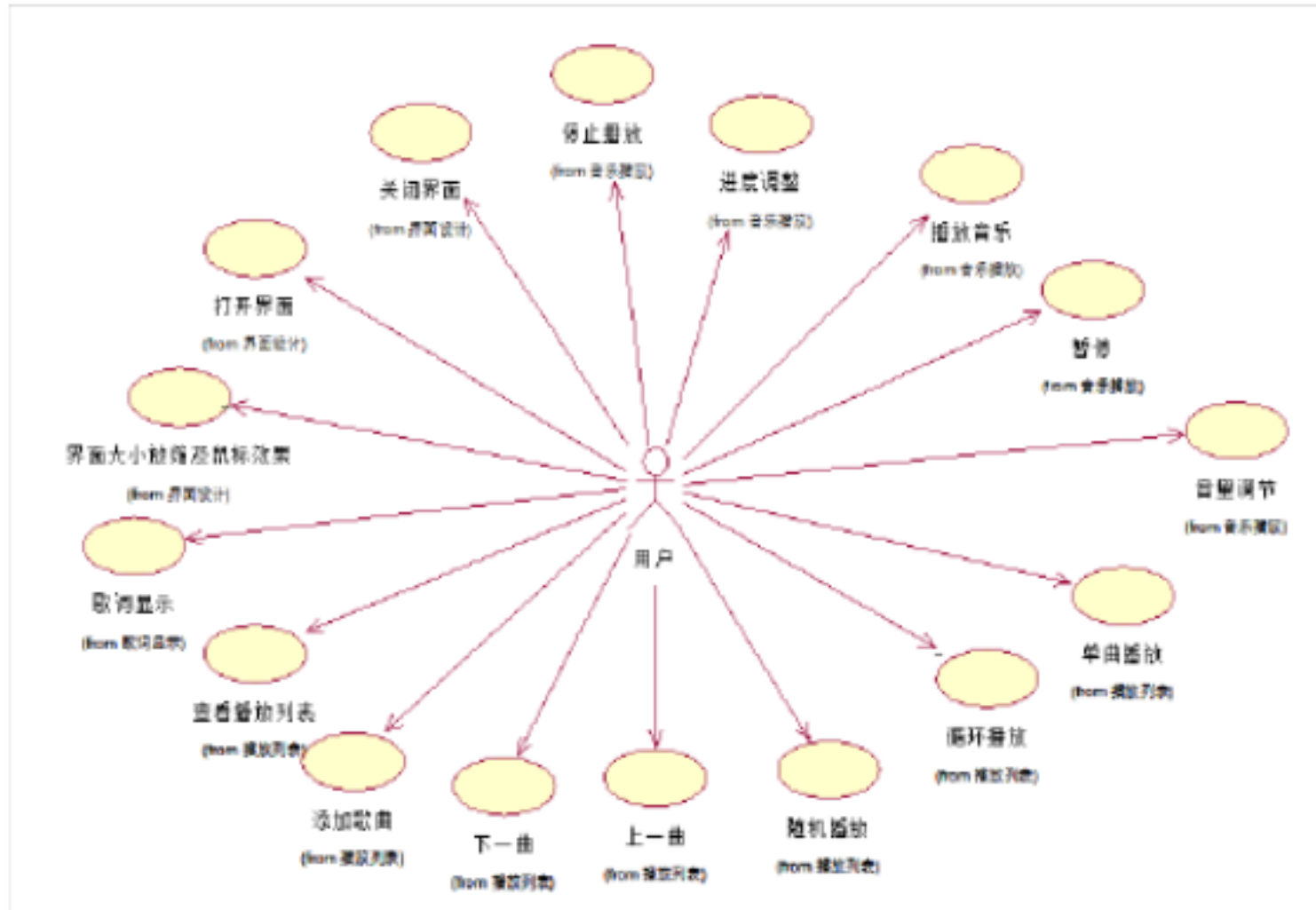
功能类别	子功能	子功能
界面设计	界面最大、最小化及关闭界面	无
	界面移动效果：鼠标事件	无
音乐播放	音乐播放功能	无
	音乐播放状态显示	无
	音乐播放控制	播放状态控制
		播放音量控制→调节音量控制按钮
		播放进度控制→拉动进度条
播放列表	获取播放列表	无
	添加歌曲资源	无
	歌曲切换	上一曲→寻找上一个歌曲 ID
		下一曲→寻找下一个歌曲 ID
		播放模式按钮→单曲循环
		播放模式按钮→列表循环
		播放模式按钮→随机播放
歌词显示	无	无

2.1.2　系统功能结构图



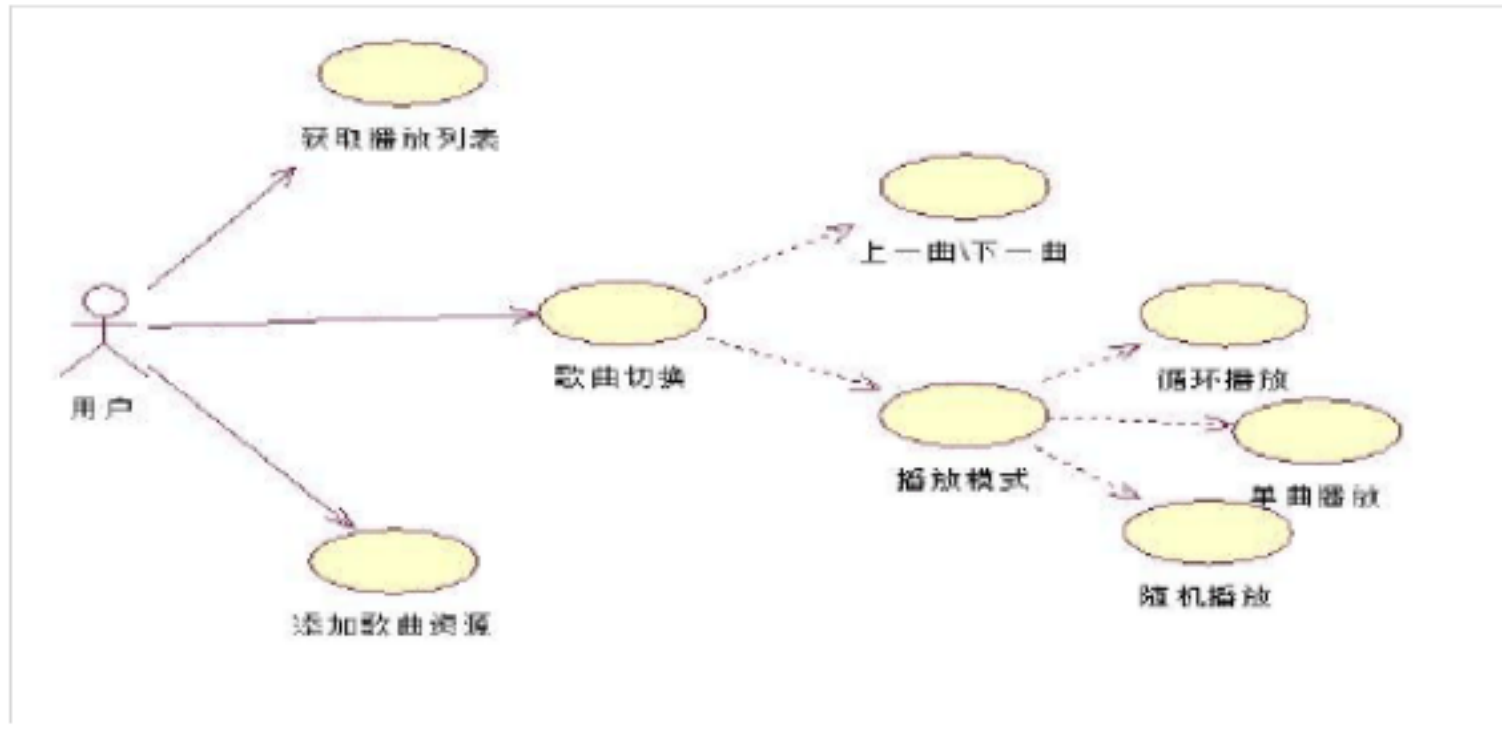
2.2　整体流程分析

2.3　整体用例分析

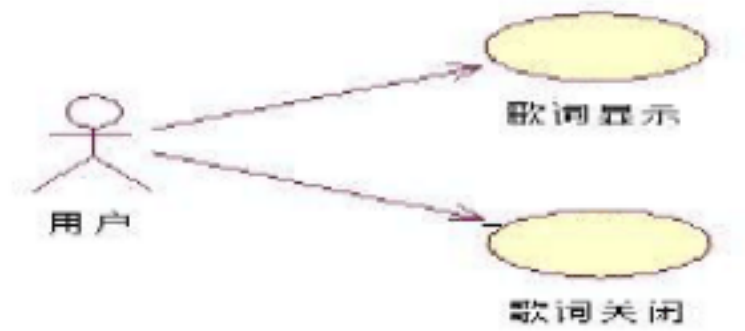


2.4　功能模块用例图

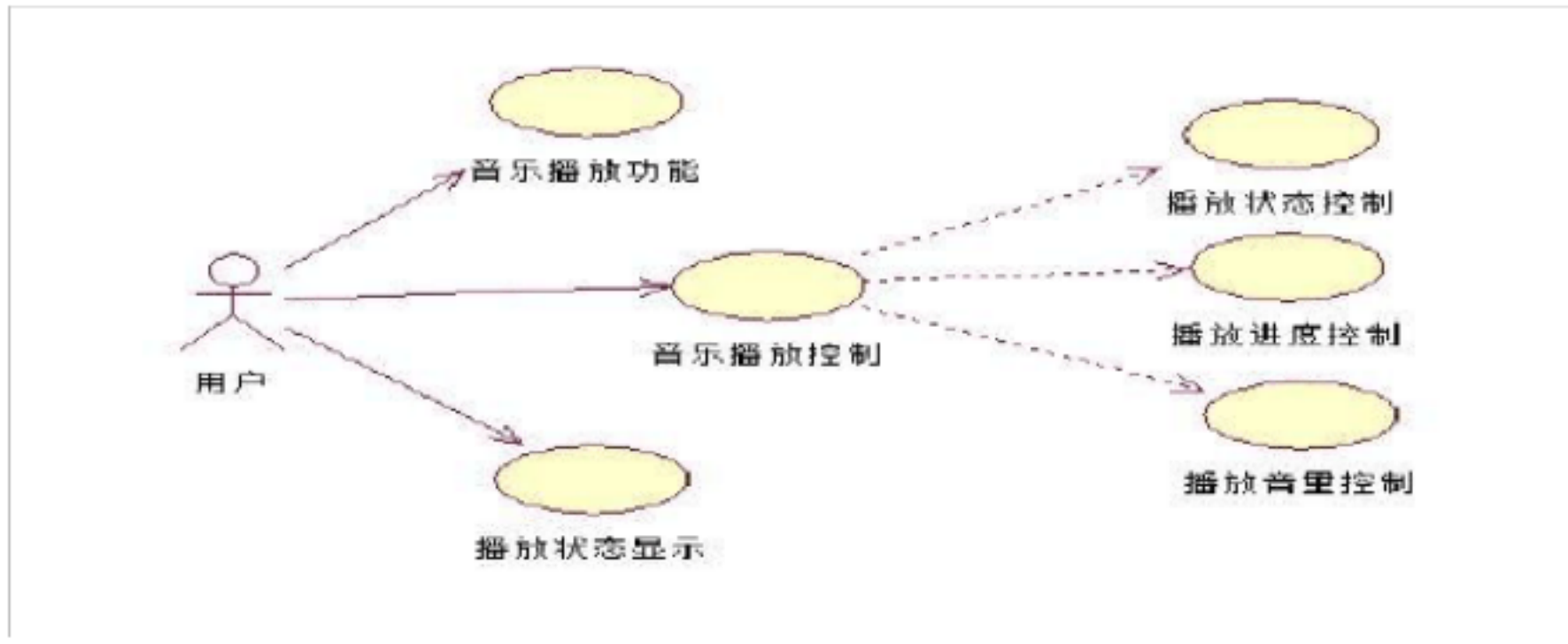
1. 播放列表模块用例图



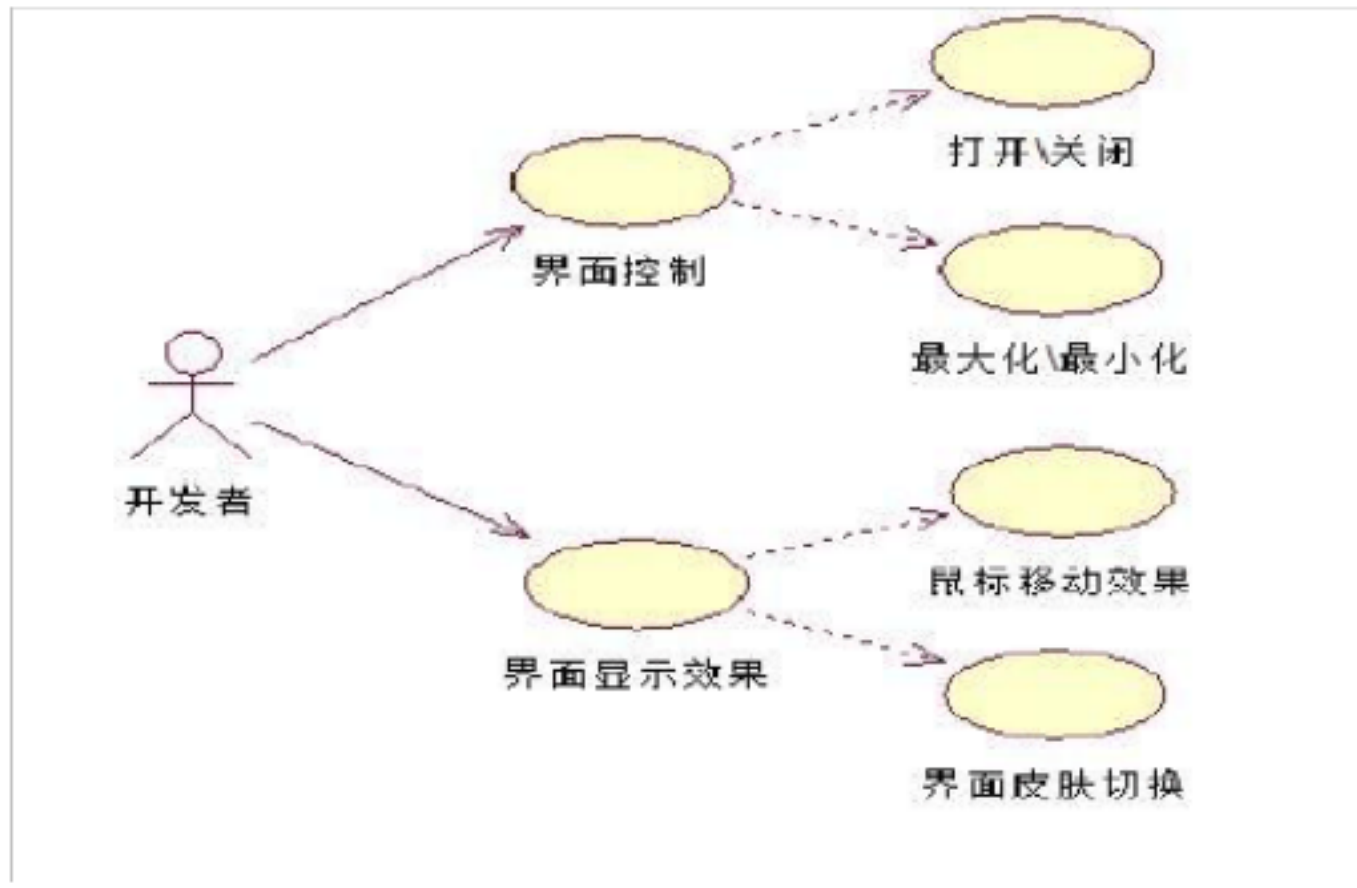
2. 歌词显示模块用例图



3. 音乐播放模块用例图



4. 界面设计模块用例图



三. 详细设计

3.1 主界面设计

音乐播放器包含的功能模块比较复杂，因此需要将各个功能模块进行划分。

在设计的过程中，我们将整个界面合理的划分为两个部分，包括：

3.1.1 、歌曲信息及管理

（1） 当前播放、时间条、三个操作按钮，音量控制

（2）播放列表（左）歌词显示（右）

（3）页面的最大化，最小化和关闭

3.1.2 、右边的歌词面板

歌词显示

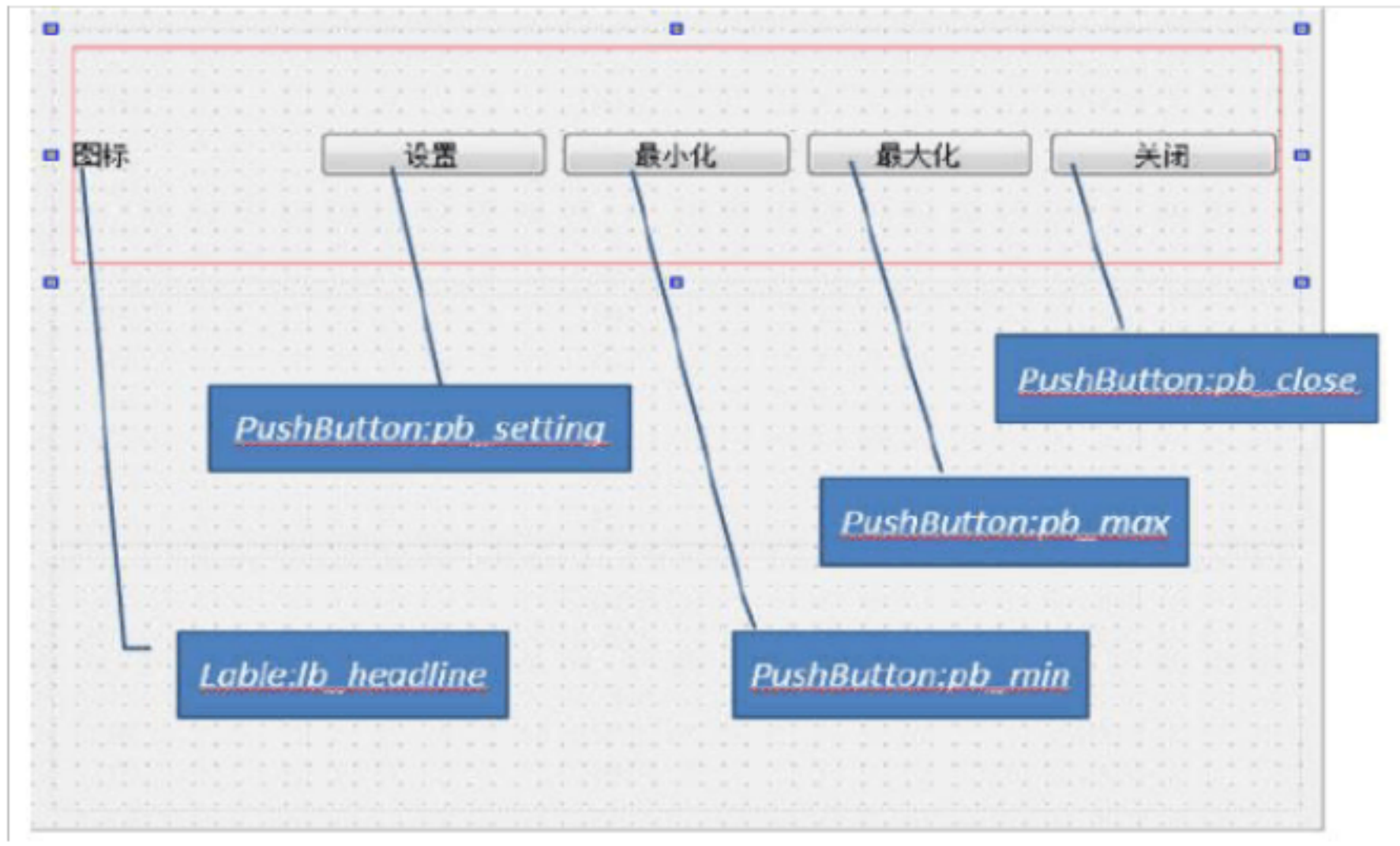


图 1. 界面设置

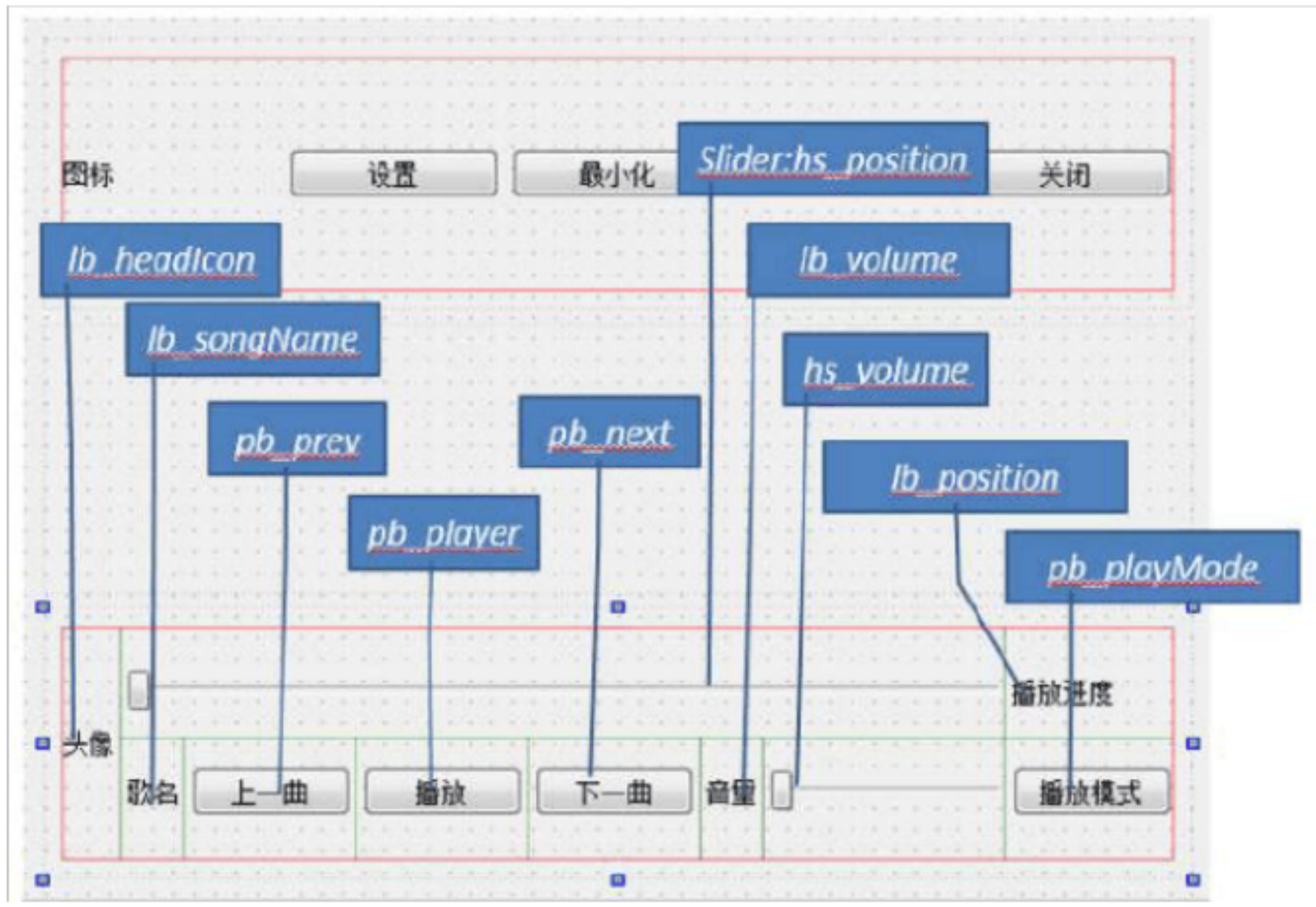


图 2. 界面设置

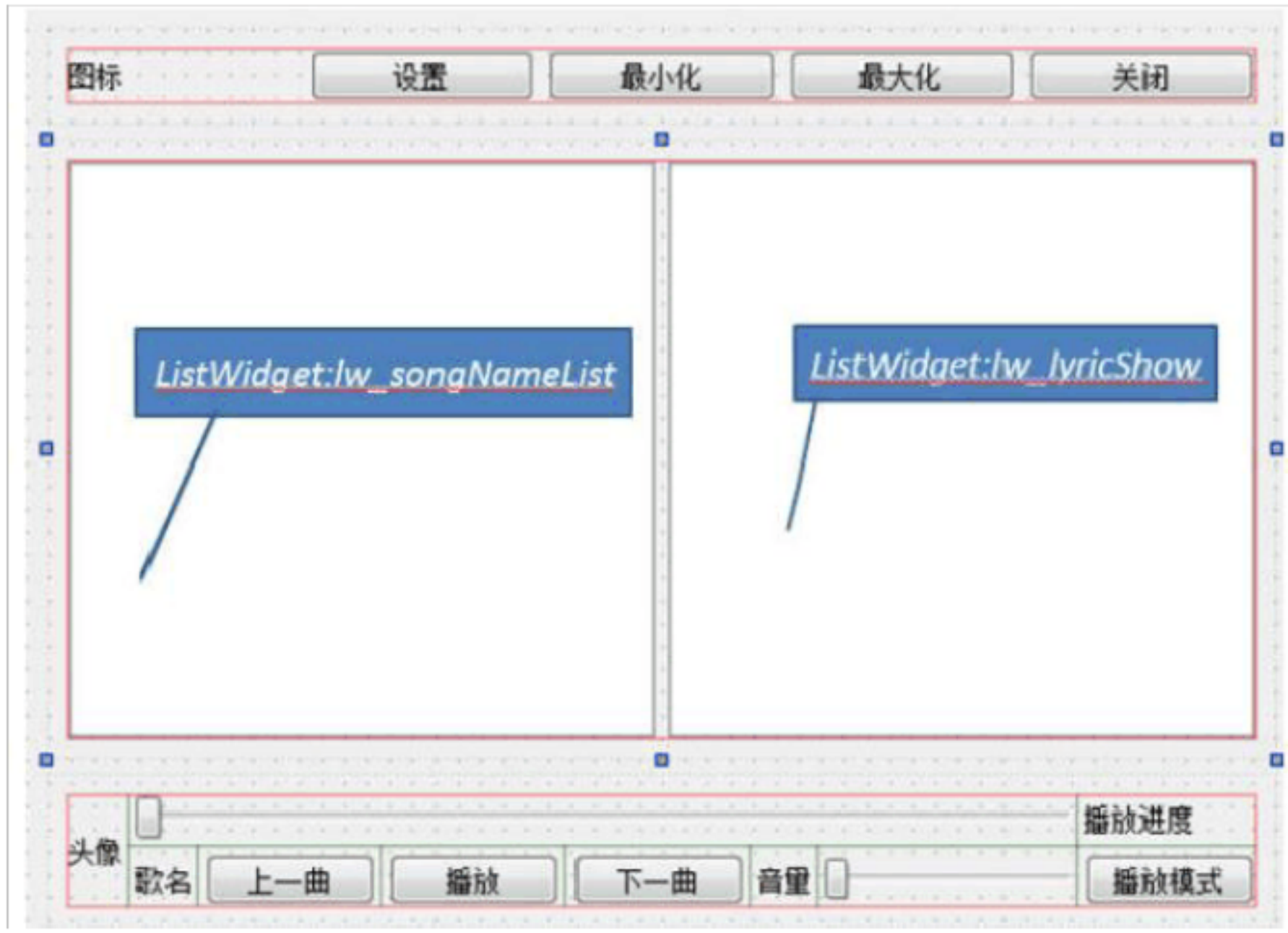


图 3. 界面设置

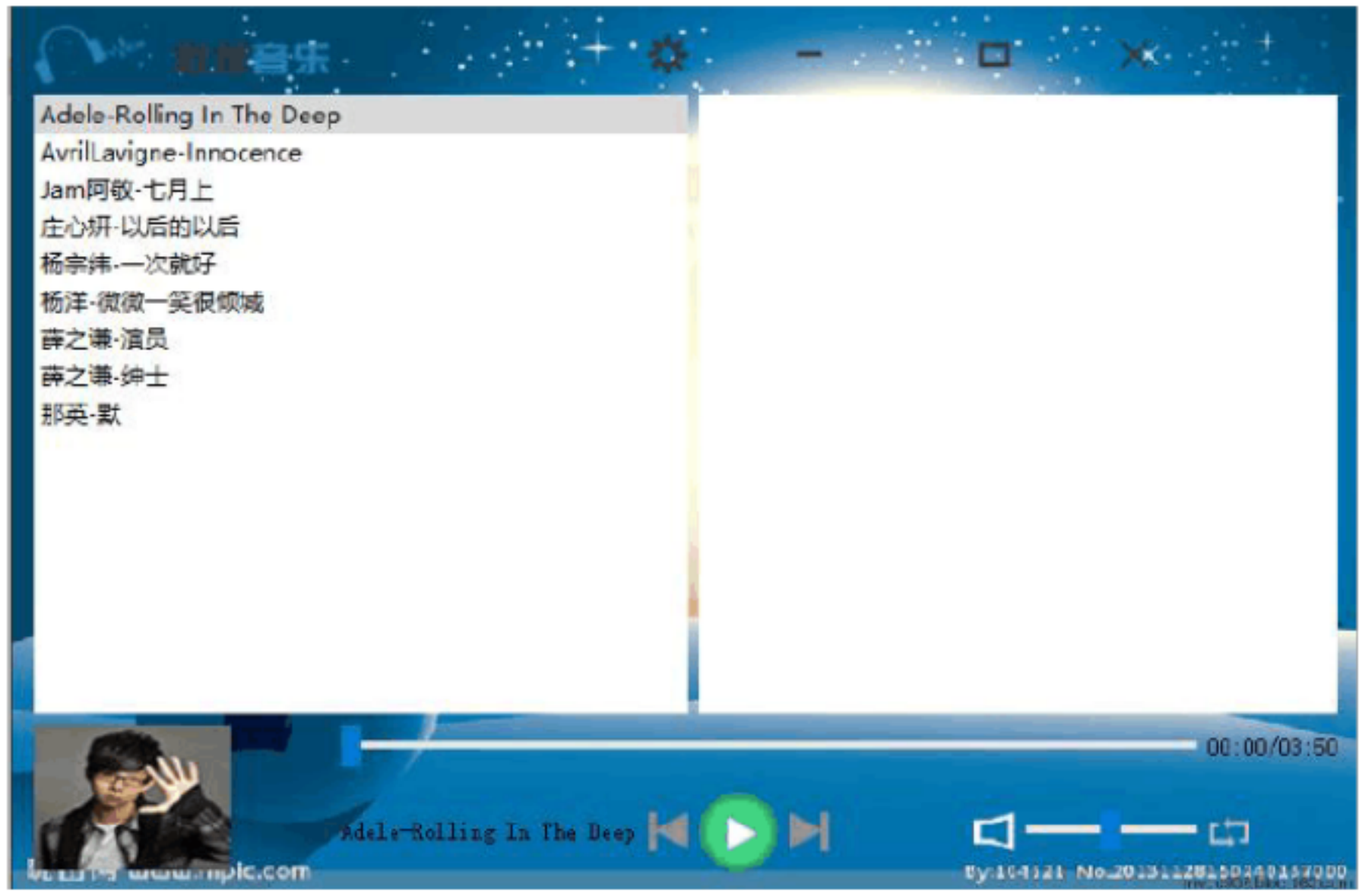


图 4. 播放界面

3.1.3　界面最大化、最小化、关闭代码实现

```
privateslots    :
//  界面最大化
void on_pb_min_clicked();
//  界面最小化
void on_pb_max_clicked();
//  界面关闭
void on_pb_close_clicked();
//  最大化、最小化和关闭实现
void Widget::on_pb_min_clicked()
{
this  ->showMinimized();
}
void Widget::on_pb_max_clicked()
{
if  (this  ->isMaximized())
{
this  ->showNormal();
}
else
{
this  ->showMaximized();
}
}
void Widget::on_pb_close_clicked()
{
this  ->close();
}
```

3.1.4　鼠标事件代码实现

```
//  鼠标事件
```

```
void Widget:: mousePressEvent( QMouseEvent*ev)
{
if (ev->buttons()== Qt:: LeftButton )
{
m_widgetMove=ev->globalPos()- this ->frameGeometry().topLeft();
ev->accept(); // 鼠标事件被系统接收
}
}
void Widget:: mouseMoveEvent(QMouseEvent*ev)
{
if (ev->buttons()== Qt:: LeftButton )
{
this ->move(ev->globalPos()- m_widgetMove);
ev->accept();
}
}
```

3.1.5 背景图片代码实现

```
// 背景图片
void Widget:: paintEvent (QPaintEvent *)
{
QPainter pa(this );
image->load(":/new/src/resource/icons/back6.png");
QRectsource( 0, 0, image->width(), image->height());
QRecttarget( 0, 0, this ->width(), this ->height());
pa.drawImage(target,* image,source);
}
```

3.2 音乐播放

3.2.1 音乐播放

```
void Widget::initPlayer( void )
{
```

```
m_songName="E:/Qt/yanyu/song/Adele-RollingInTheDeep.mp3";
m_player=new QMediaPlayer;
//m_player->setMedia(QUrl::fromLocalFile(m_songName));
m_player->setPlaylist(m_playList);
m_player->setVolume(50);
//m_player->play();
m_playIndex=0;
m_playList->setCurrentIndex(m_playIndex);
m_songName=m_songNameListat(m_playIndex);
connect(m_player,SIGNAL(durationChanged(qint64)),this,SLOT(slotDurationChanged(qint64)));connect(m_player,SIGNAL(positionChanged(qint64)),this,SLOT(slotPositionChanged(qint64)));
}
```

3.2.2 播放状态控制

（1）自定义相关槽

```
// 歌曲播放进度自定义槽
void slotPositionChanged(qint64 position);
// 当前歌曲总时长自定义槽
void slotDurationChanged(qint64 duration);
```

（2）进度条

```
// 进度条
void Widget::slotDurationChanged(qint64 duration)
{
m_currentPlayerTime=0;
m_totalPlayerTime=duration/1000;
QTime currentTime((m_currentPlayerTime/3600)%60,(m_currentPlayerTime/60)%60,
m_currentPlayerTime%60,(m_currentPlayerTime*1000)%1000);
QTime totalTime((m_totalPlayerTime/3600)%60,(m_totalPlayerTime/60)%60,
m_totalPlayerTime%60,(m_totalPlayerTime*1000)%1000);
```

```
m_playPosition =currentTime.toString( "mm:ss")+"/" +totalTime.toString( "
mm:ss");
ui ->lb_position ->setText( m_playPosition );
ui ->hs_position ->setMaximum(m_totalPlayerTime );
}
void Widget::slotPositionChanged( qint64 position)
{
m_currentPlayerTime =position/ 1000;
QTime currentTime(( m_currentPlayerTime / 3600)%60,( m_currentPlayerTime / 6
0)%60, m_currentPlayerTime %60,( m_currentPlayerTime * 1000)%1000);
QTime totalTime(( m_totalPlayerTime / 3600)%60,( m_totalPlayerTime / 60)%60,
m_totalPlayerTime %60,( m_totalPlayerTime * 1000)%1000);
m_playPosition =currentTime.toString( "mm:ss")+"/" +totalTime.toString( "
mm:ss");
ui ->lb_position ->setText( m_playPosition );
}
```

（3） 暂停，开始播放控制

```
void Widget::on_pb_player_clicked()
{
if (m_player ->state()== QMediaPlayer:: PlayingState )
{
m_player ->pause(); ui ->pb_player ->setStyleSheet( "border-image:url(:/ne
w/src/resource/images/playStartNormal.png);" );
}
else
{
m_player ->play(); ui ->pb_player ->setStyleSheet( "border-image:url(:/new
/src/resource/images/pausePressed.png);" );
}
}
```

（4）播放进度控制、音量控制

```
//  播放进度控制
void Widget::on_hs_position_sliderMoved(       int  position)
{
m_player ->setPosition(position*        1000);
}
//  播放音量控制
void Widget::on_hs_volume_sliderMoved(    int  position)
{
m_player ->setVolume(position);
}
```

3.3 播放列表

3.3.1   获取播放列表

```
//  初始化播放列表
void Widget::initPlayList(        void )
{
m_songPath="E:/Qt/yanyu/song"    ;
m_playList  =new QMediaPlaylist   ;
QDir dir(  m_songPath);
QFileInfoList     infos=dir.entryInfoList(          QStringList   ()<< "*.mp3"  , QDir::  Fil
es, QDir::  Name);
foreach  ( const QFileInfo   &info,infos)
{
m_songNameList.append(info.baseName());
m_playList  ->addMedia( QUrl(info.absoluteFilePath()));          //  添加歌曲资源
}
//  播放模式初始化
m_playMode=3;
m_playList  ->setPlaybackMode(  QMediaPlaylist   :: Loop);connect(   m_playList  ,
SIGNAL(currentIndexChanged(   int  )),  this  , SLOT(slotCurrentIndexChanged(    in
t )));
```

}

3.3.2　歌曲切换

（1）歌曲播放模式

```
// 播放模式
void Widget::on_pb_playMode_clicked()
{
m_playMode++;
m_playMode%=5;

switch(m_playMode)
{
caseQMediaPlaylist::CurrentItemOnce:
m_playList->setPlaybackMode(QMediaPlaylist::CurrentItemOnce);
ui->pb_playMode->setStyleSheet("border-image:url(:/new/src/resource/i
mages/currentItemOncePressed.png)");
//ui->pb_playMode->setText("单曲播放");
break;
caseQMediaPlaylist::CurrentItemInLoop:
m_playList->setPlaybackMode(QMediaPlaylist::Sequential);
ui->pb_playMode->setStyleSheet("border-image:url(:/new/src/resource/i
mages/currentItemInLoopPressed.png)");
//ui->pb_playMode->setText("单曲循环");
break;
caseQMediaPlaylist::Sequential:
m_playList->setPlaybackMode(QMediaPlaylist::Sequential);
ui->pb_playMode->setStyleSheet("border-image:url(:/new/src/resource/i
mages/sequentialPressed.png)");
//ui->pb_playMode->setText("顺序播放");
break;
caseQMediaPlaylist::Loop:
m_playList->setPlaybackMode(QMediaPlaylist::Loop);
```

```
ui ->pb_playMode->setStyleSheet(    "border-image:url(:/new/src/resource/i
mages/loopPressed.png)"   );
//ui->pb_playMode->setText("      循环播放 ");
break ;
caseQMediaPlaylist   ::  Random
m_playList  ->setPlaybackMode(  QMediaPlaylist   ::  Random);
ui ->pb_playMode->setStyleSheet(    "border-image:url(:/new/src/resource/i
mages/randomPressed.png)"  );
//ui->pb_playMode->setText("      随机播放 ");
break ;
default   :
break ;
}
}
```

（2）上一曲、下一曲切换

```
//  上一曲
void Widget::on_pb_prey_clicked()
{
m_playIndex =m_playList  ->previousIndex();
if  (m_playIndex ==-1)
{
m_player ->stop();
}
else
{
m_playList  ->setCurrentIndex(    m_playIndex );
}
}
//  下一曲
void Widget::on_pb_next_clicked()
{
```

```
//image->load(":/new/src/resource/icons/background.png");
//this->update();
m_playIndex =m_playList  ->nextIndex();
if  ( m_playIndex ==-1)
{
m_player ->stop();
}
else
{
m_playList  ->setCurrentIndex(     m_playIndex );
}
}
```

3.4　歌词显示

3.4.1　获取歌词及歌词跟踪

```
//  获取歌词
void Widget::getCurrentLyric(       void )
{
m_lyricPath  ="E:/Qt/yanyu/lyric"        ;
QString fileName= m_lyricPath  +"/"  +m_songName+".lrc"    ;
QFile lyricFile(fileName);
if  (!lyricFile.        open( QIODevice::  ReadOnly))
{
qDebug()<< "Error:FileOpened"    ;
}
else
{
quint64  index= 0;
QString  line,lyric,pos;
QStringList    posAndLyric;
QTextStreamin(&lyricFile);
```

```
ui ->lw_lyricShow  ->clear();
m_lyricList    .clear();
for  ( int  i= 0;i<  ui ->lw_lyricShow  ->count();i++)
{
ui ->lw_lyricShow  ->takeItem(i);
}
m_mapLyricIndex .clear();    //  歌词跟踪
while  (!in.atEnd())
{
line=in.readLine();
posAndLyric=line.split(       "]"  );
lyric=posAndLyric.at(      1);
pos=posAndLyric.at(    0).mid(  0, 6);
QStringList    time=pos.remove(  "["  ).split(     ":"  );
quint64  t=time[   0].toInt()*      60+time[  1].toInt();
m_mapLyricIndex .insert(t,index++);       //  歌词跟踪逻辑数据记录
m_lyricList    .append(lyric);
posAndLyric.clear();
}
lyricFile.      close ();
}
ui ->lw_lyricShow  ->addItems(  m_lyricList    );
for  ( int  i= 0;i<  m_lyricList    .size();i++)
{
ui ->lw_lyricShow  ->item(i)->setTextAlignment(       Qt::  AlignCenter  );
}
}
```

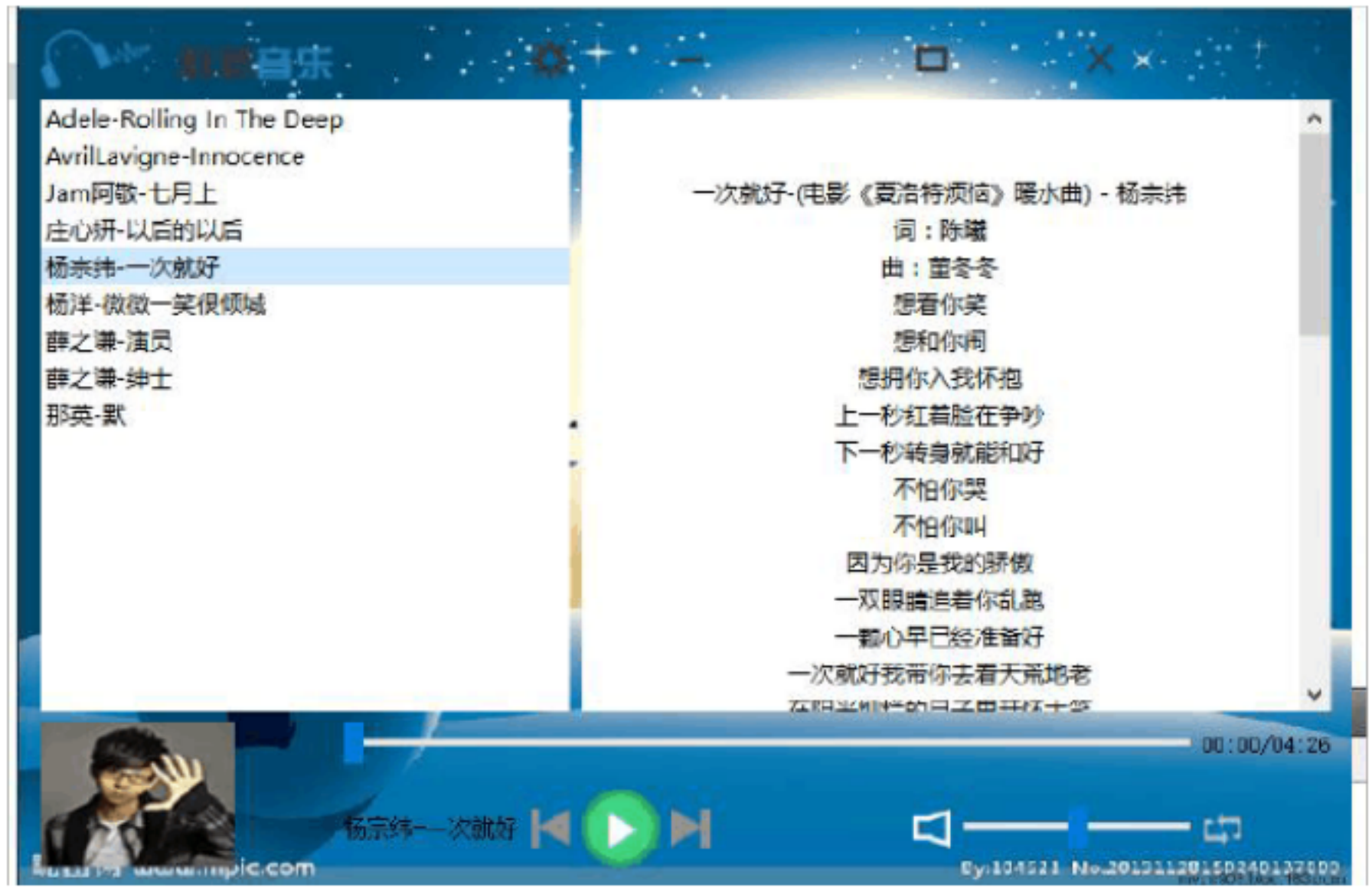


图 5 运行效果

四. 心得体会

这次的软件课程设计题目 , 是非常贴近我们生活的音乐播放器，这一类的应用软件大大的娱乐和方便了我们的课余生活，但是亲自动手设计一个音乐播放器对我们来说却有一定的难度。随着我们对于这方面的学习，从最初的需求分析、搜集资料、确定程序语言，到中期的代码编写和检测完善，再到后面的软件演示和文档编写，我们真正学到了不少使用的技术。

本次软件课程设计采用了自主完成的方式，锻炼我们独立解决问题的能力，但是由于对软件的功能认识不足，导致了我们一些好的灵感和创意在技术上去无法实现，也是不小的遗憾。但是通过这次软件课程设计，使我们对程序设计的整个流程以及代码的熟悉与规范有了新的提高。亲手完成一次软件课程设计，一方面，在理论上它让我们学会了相关知识。另一方面，通过实践对以后的软件学习乃至参加工作都是由极大的价值的。

最后要感谢来自上海杰普软件公司的指导老师对我们的悉心指导，及时这次设计最后的实现结果与最初的构想还存在一点差距，但在设计过程中的学习和收获的财富，将使我们受益终身。