**Spec设计文档**

目录

[1 系统需求 1](#_Toc31740)

[1.1 功能描述 2](#_Toc28711)

[2 系统分析 2](#_Toc9905)

[2.1 解决思路 2](#_Toc25186)

[2.2模式描述 3](#_Toc2070)

[3 系统设计 4](#_Toc26774)

[3.1 代码框架 4](#_Toc32365)

[3.2 UML类图 5](#_Toc21072)

[3.2.1控件类 5](#_Toc9249)

[3.3 界面设计 6](#_Toc21485)

[4 系统主要编码实现 8](#_Toc4869)

[5 测试与运行结果 9](#_Toc5526)

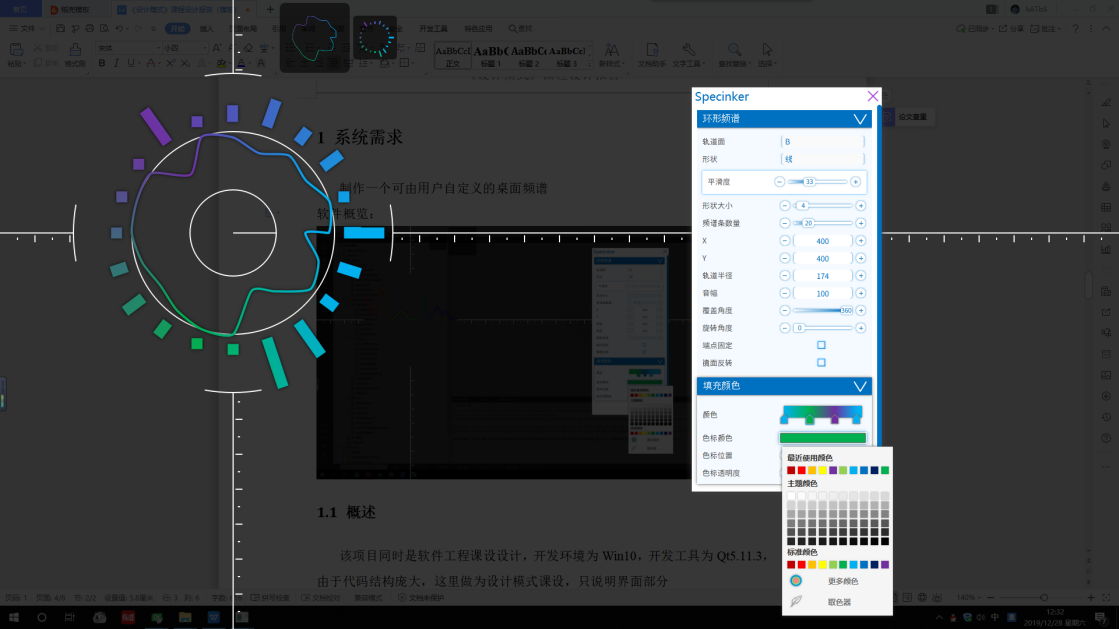
# 1 系统需求

需求就是完成作业的同时做一点炫酷的东西，该软件能够让用户自己动手设计音乐的可视化动画，这个项目同时也是软件工程课程设计。

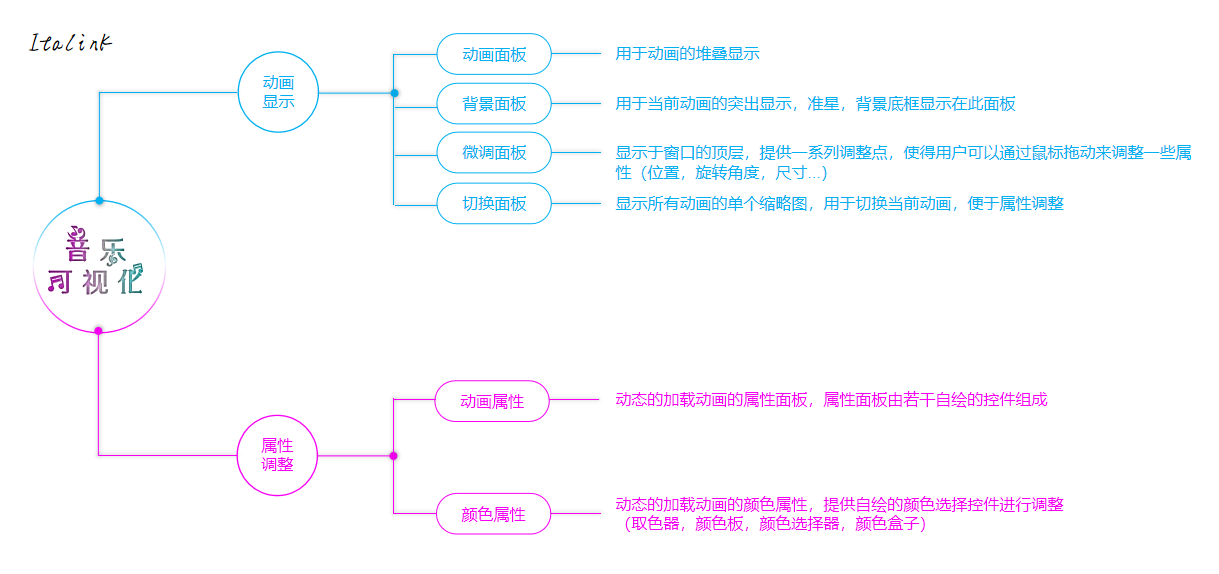
原理：通过监听（实时录制）系统的音频输出设备，分析其音频的波形数据，根据提供的动画生成函数生成动画，用户可以通过一个属性窗口对这些动画进行管理和修改。

提交的是项目中的界面（前端）部分。

**软件概览：**



**界面框架：**



## 功能描述

功能概括：

（1）引擎：监听（实时录制）系统音频输出设备，分析其音频波形

（2）可视化：动画生成器提供一个图形绘制函数，通过从引擎得到的波形数据，进行动画绘制。

（3）属性可定制：动画生成器提供一个属性面板，用户可调整其属性来修改动画。

# 2 系统分析

## 2.1 问题及解决思路

**问题1：在不同环境下使用这个软件，比如窗口尺寸不同，软件如何自适应？**

使用享元模式和单例模式，创建一个Config类，使用饿汉式单例。在类的构建时，动态的去读取这些信息，而软件设计中是通过Config类来进行配置，因此可以使得软件能够适应不同环境。

**问题2：如何构建出一套完整的颜色选择控件，使得其中不同的控件可以共同协作调整同一个色值？**

使用中介者和观察者模式，构造一个颜色中心（color center），所有的颜色控件都是对它内部的颜色进行修改，当一个颜色控件捕获到一个颜色后，首先修改颜色中心的color值，然后颜色中心发送一个刷新信号，各个控件再刷新其内部的颜色。

**问题3：一个被嵌套了很多层的窗口对象，已经“埋”得很深了，这时如果一个外部类使用它，难道一层一层的通过参数传递？**

使用单例模式，深刻体会到单例模式的用处，单例模式的巧妙之处不在于“单例”，更为有用的是提供了一个全局的公共访问点，其中很多窗口类都用到了单例，颜色控件中的取色器，颜色板，颜色选择器，显示窗口，以及属性窗口，由于软件中除了主线程之外，还有一个引擎的波形分析线程，并没有涉及到界面操作，因此无需考虑线程安全，又由于所有Qt窗口对象创建之前，必须先构造一个QApplication对象，所以无法使用饿汉单例在类的构建时就创建单例对象。因此，这里用到的所有单例大都是懒汉式。

**问题4：很多时候会需要创建“相似”的动画生成器，应该如何处理？**

使用原型模式，对动画模式进行深拷贝，动画类存在一个抽象的clone函数，主要是拷贝原型的属性值，以及颜色值。

**问题5：每一个微调控件都有各自的值修改信号，如果每一个控件都按各自的信号，那么处理的时候就需要对它们一个一个的进行处理，这无疑是一个非常繁琐的事情，且很容易出错，也不容易排查。**

使用了适配器模式，一些控件是自己实现的，一些是控件是使用自带控件。为了让所有的微调控件能够被统一的处理，因此我创建了一个adjuster类，内部提供一个valueChange信号。各个控件只需继承自adjuster，并且连接valueChange信号，即可。

**问题6：有时候选择一个动画的属性之后，可能还会需要对这个属性进行细微的设计，比如我选择“点”类型频谱，那么应该可以修改它的“点大小”，如果修改为“线”类型频谱，那么就应该可以修改它的“线宽”。也就是说，可能存在一些动态的属性，那么该怎么解决这个问题？**

使用组合模式，使得控件能“包含”控件，当父控件的值变动时，切换相应的子控件。

**问题7：只提供一个动画的设计是否使得软件过于“乏力”，能不能再进一步使得整个软件能绘制其他的自定义动画？**

的确，单纯的绘制一个类型的频谱动画，最多也就是将这些动画组成不同的形状，用户要不了多久就会失去兴趣。因此我是这样思考动画的扩展：一个动画无非就是连续显示多张图片。所以，要实现一个可定制的动画，需要的就是一个属性调节面板（动画属性+颜色属性）加一个根据这些属性来绘制动画的“图片生成器”。综上，我对项目进行了重构，且将动画向上抽象，构建了一个抽象的动画类。只需提供一个动画的生成函数，以及属性面板，就可扩展额外动画，类图在下。

## 2.2模式描述。

（1）享元模式：Config类加载环境信息，完成软件的自适应设置

（2）单例模式：提供全局公共访问点

（3）中介者模式、观察者模式：实现颜色公用，观察中介者的状态

（4）原型模式：动画对象的深拷贝。

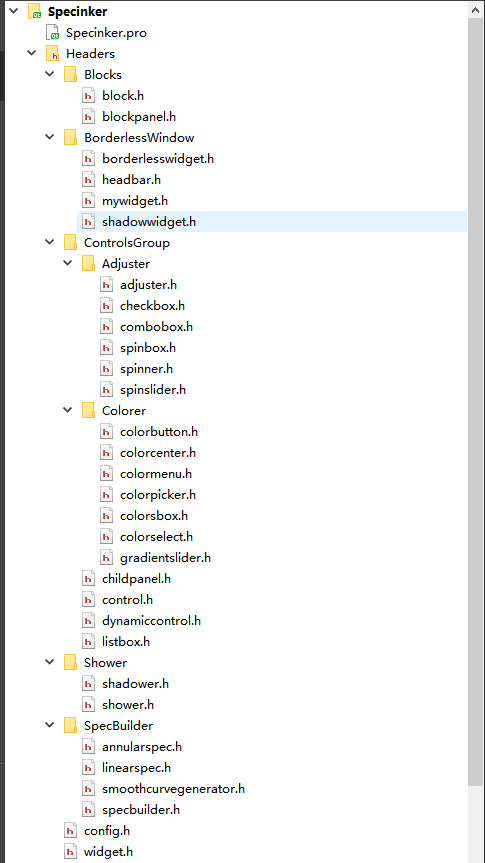
（5）适配器模式：解决自绘控件与原生控件的信号差异问题

（6）组合模式：实现动态属性

（7）迭代器模式：使用Qt内置迭代器。

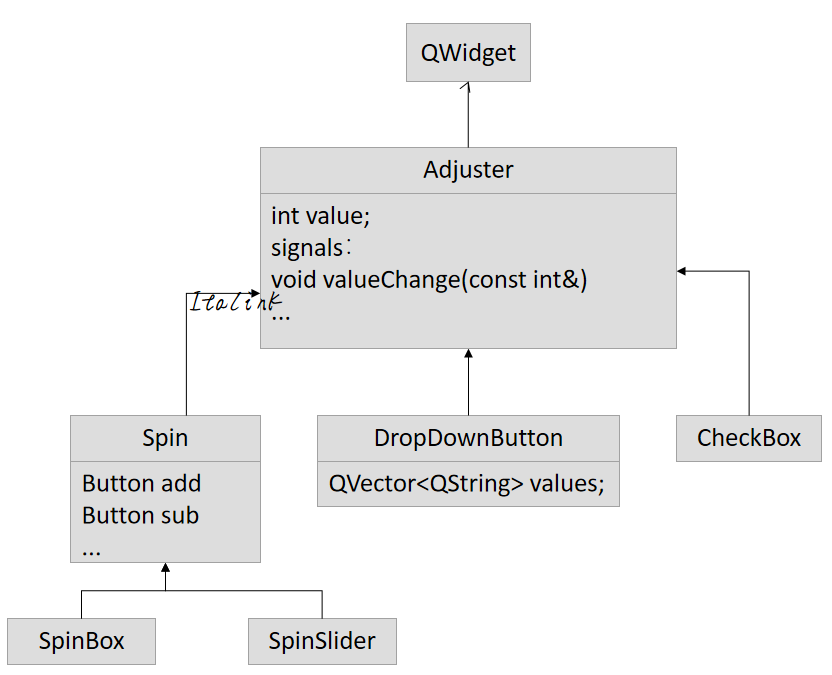
# 3 系统设计

## 3.1 代码框架

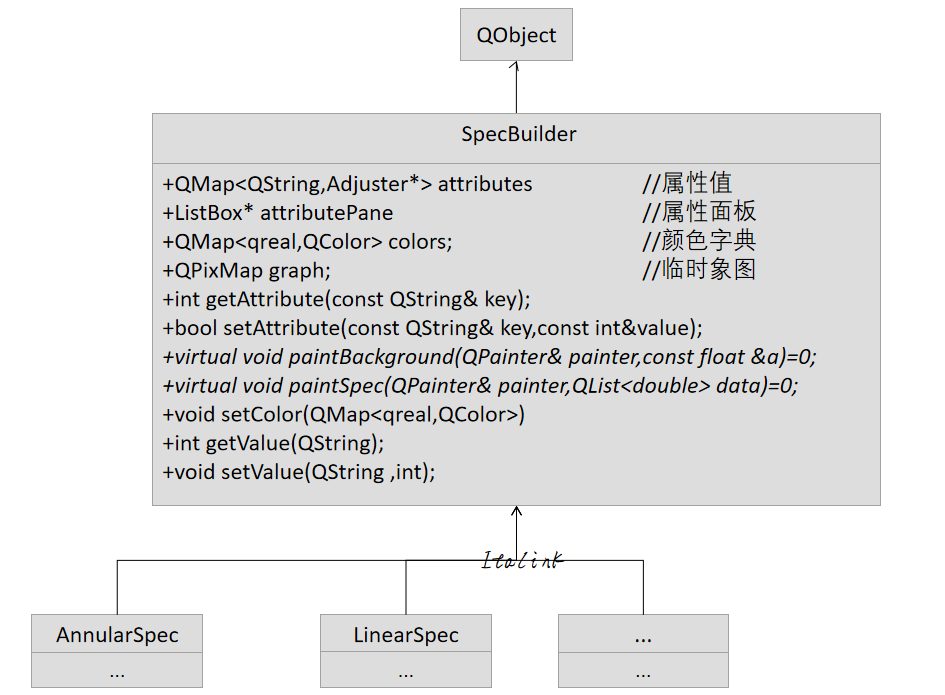


## 3.2 UML类图

## 3.2.1控件类



**3.2.2动画生成器类**

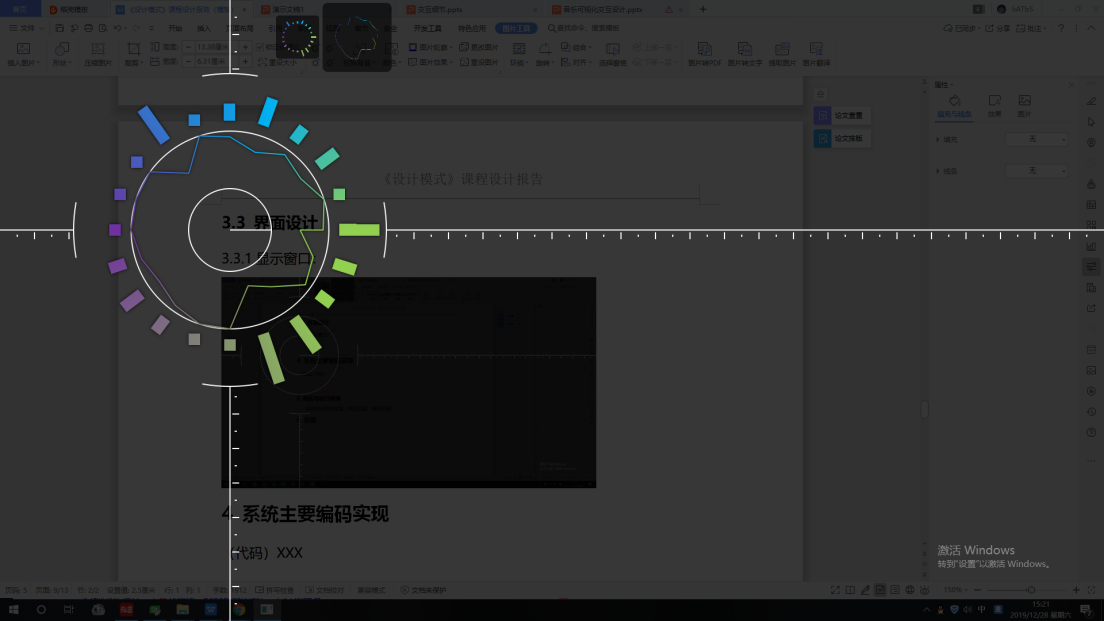


## 3.2.3 ......

## 太多了，画不动了，我就一敲代码的T.T，想看类图老师打开源码的头文件，类的结构一目了然

## 3.3 界面设计

3.3.1显示窗口：



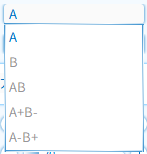
显示界面由四部分构成：

* 背景面板：绘制当前动画的底框（准星、轨道、...）
* 动画面板：在这个面板上进行动画绘制
* 切换面板：位于顶部中间的窗口，内部有一系列透明小黑框，用于选择当前动画和显示缩略图
* 微调面板：（doing...）

3.3.2属性窗口



3.3.3自绘微调控件



组合框（ComboBox）



微调滑动条（SpinSlider）



微调框（SpinBox）

3.3.4 颜色选择控件

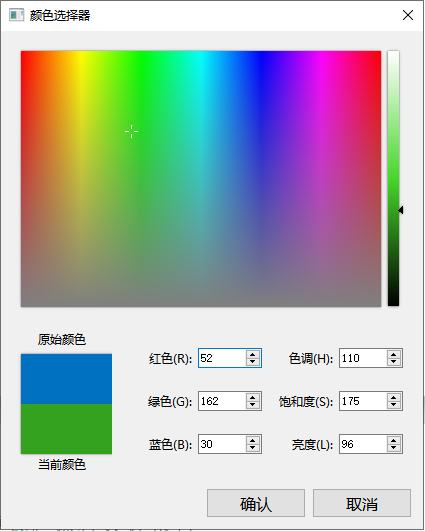
颜色盒子（ColorBox）



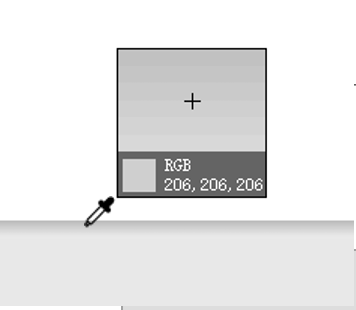
颜色板（ColorPanel）



颜色选择器（ColorSelecter）



取色器（ColorPicker）



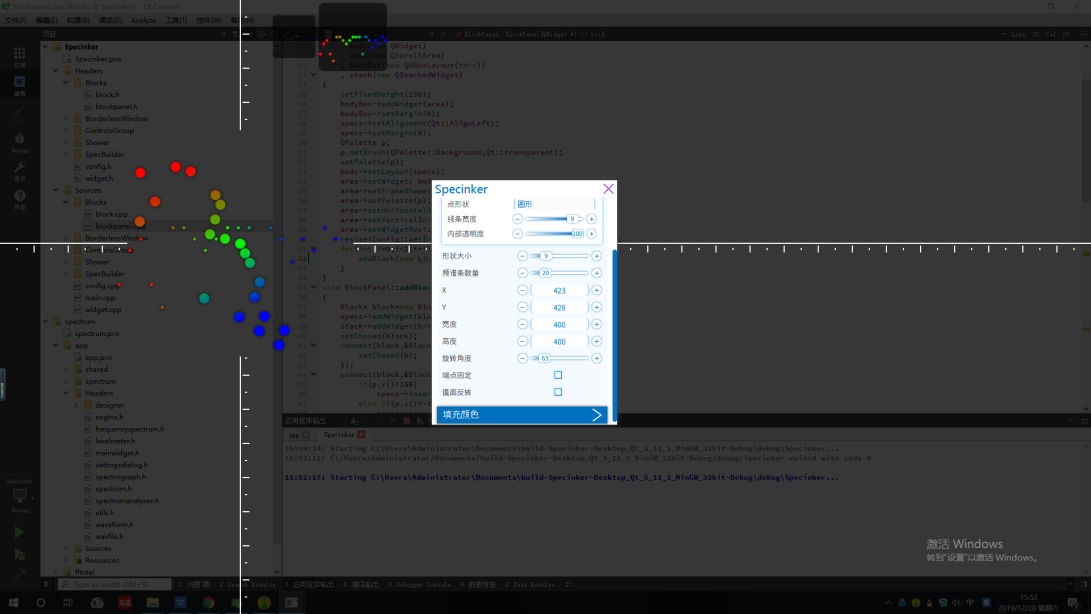
# 4 系统主要编码实现

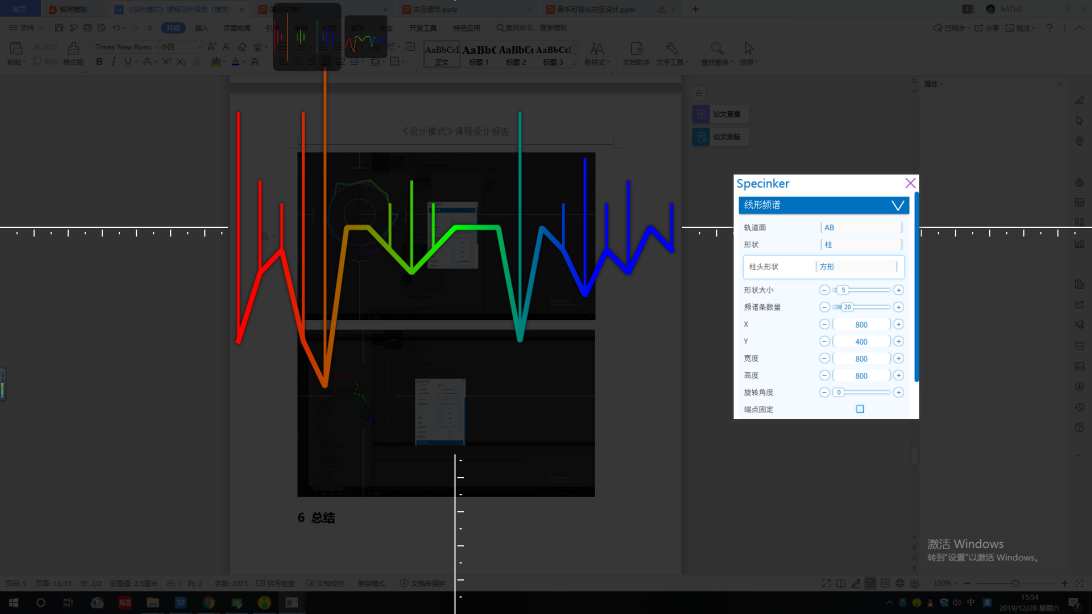
代码庞大，请看代码附件

## 5 测试与运行结果

简单进行的3个测试

## 





# 6 总结

通过这次课程的学习，学到了很多有用的设计模式，代码整体的结构也比原来清晰很多。这个项目同时也是软件工程的课程设计，历时三个月，因为之前写的太乱，且可扩展性很差，所以用学到设计模式对软件进行重构，效果显著！

# 参考文献

参考文献必须是在课程设计中真正阅读和运用过的，文献按照在正文中的出现顺序排列。

[1]《java设计模式》-刘伟著

[2] Qt官方开发文档