Chart System

软件工程 专业

学生　　指导老师　黄武

1 项目说明

1.1项目概述

运用学到的理论知识构建了一个Chart 系统，可以打开任意一个文件，以二进制的方式读取，然后以曲线形式显示在自己构建的Chart 窗口中。

使用Python3 的语言编程，并使用了 numpy 等库，UI 基于PyQt5，自行绘制Canvas 实现了Chart System。程序能够以二进制形读取任何格式的文件，然后进行绘图。在读取二进制文件时，每次读取一个字节的二进制数，然后转为整数型存入 List 并返回，文件读取完成后，对数组进行处理。然后进行画图，画图的时候，上半部分是第一个通道（原始数据），下半部分是第二个通道（对数据进行处理）；纵轴的值即是从文件读取出来的数组的元素值，横轴以0开始，步长为1（可以在配置文件中修改，具有默认值），开始增加，与y 值一一对应。此外，显示的图片的线条颜色，粗细和点的大小、颜色等等属性都可以在配置文件中修改，根据自己喜好。在菜单栏中，可以由用户选择对数据进行的操作，并且在第二个通道中显示。实现了图像的显示、放大，缩放，以及对数据进行 sin，cos 求值等功能。以及实现了用户保存生成的最后的图像，选择文件路径和文件名等。

1.2项目需求

1. 可以显示多个通道，比如 1 通道用于显示原始数据，2 通道用于显示对原始数据的处理（微分、积分等处理）；

2. 具有数据拖动查找功能，通常而言，屏幕宽度并不能够显示整个数据文件，因此通过前后拖动滚动条来定位不同时间段的数据；

3. 通用用户交互可以压缩和扩展显示数据。

1.3 质量要求

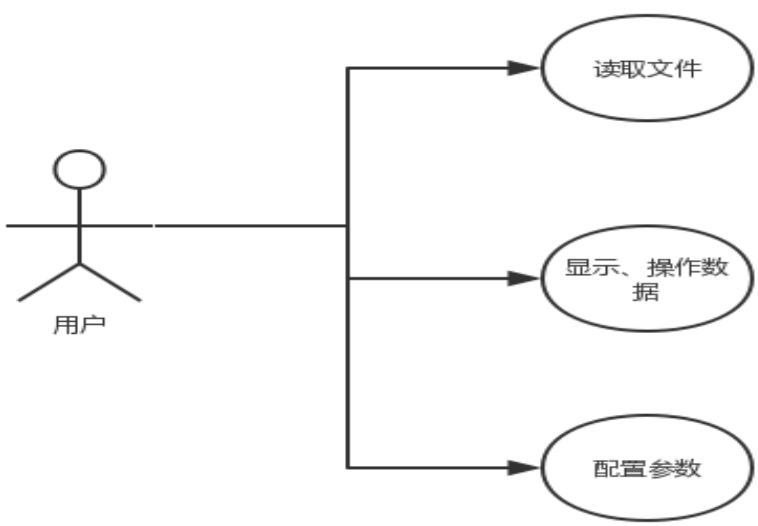
1. 编码具有良好的风格，便于阅读与理解；
2. 要有完整的构建管理过程，包括设计、编码到测试；
3. 要有完整的构建文档，从基本需求、构架设计到源码以及测试文档等。

1.4 开发简介

- 开发环境：Mac
- 软件适用：由于不依赖于操作系统，软件适用于 macOS，Linux，windows
- 开发工具：PyCharm
- 开发语言：Python
- 主要依赖库：PyQt5
- 要求：普通 PC，安装好 Python 和 PyQt5
- 本项目 GitHub 地址：https://github.com/Hzzone/chart-system

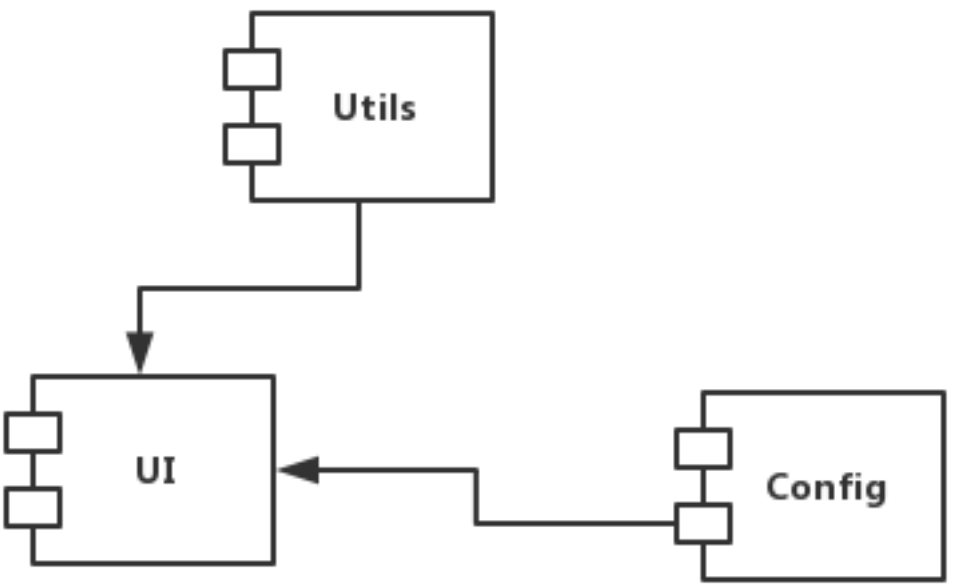
2 设计文档

2.1 UML 用例图



系统需要用户能够选择文件并读取，并且显示在通道上，选择对数据的操作时显示在第二个通道上；当用户滑动时缩小放大两个通道。用户还可以修改配置文件，对显示的各项参数进行配置。

2.2 UML 构件图



系统主要由三个模块构成：

- Utils 模块：工具模块，包括读取文件功能，和对数据进行处理的各类函数，被 UI 模块直接调用。
- Config 模块：配置模块，在这个模块中设置了 UI 模块的各类参数，包括第一、第二通道线条颜色、粗细和点的大小、颜色等，以及点之间的间距。
- UI 模块：用于绘制 UI 界面，直接在 Canvas 上进行绘制各类曲线和坐标，并且接受用户的点击时间：选择文件、数据操作、保存图片等。

2.3 模块描述

2.3.1 Utils 模块

函数名	read_file
输入	path: 文件路径。
返回	数据 List
说明	将文件以二进制的方式进行读取，进入循环，每次读取一个字节并转换成整数，然后将这个整数加入到一个列表里，当读取到的数据为空表明已经读完，退出循环。循环结束后，整个二进制文件就转变成了一个列表并返回。

函数名	sin
输入	原始数据 List
返回	处理后的数据 List
说明	对数据求 sin 值并返回

函数名	cos
输入	原始数据 List
返回	处理后的数据 List
说明	对数据求 cos 值并返回

2.3.2 Config 模块

参数名	描述	默认值

channels	最小通道数	2
first_channel_line_color	第一通道线的颜色	Qt.black
second_channel_line_color	第二通道线的颜色	Qt.black
first_channel_point_color	第一通道点的颜色	Qt.red
second_channel_point_color	第二通道点的颜色	Qt.red
first_channel_line_spacing	第一通道线的粗细	5
second_channel_line_spacing	第二通道线的粗细	5
first_channel_line_type	第一通道线的类型	Qt.SolidLine
second_channel_line_type	第二通道线的类型	Qt.SolidLine
point_size	点的大小	10
min_channel_size	最小通道的大小	270
default_slider_value	slider 默认值	50
default_slider_interval	slider 默认值	0.3
font_size	字体大小	5
window_min_size	最小大小	100
coordinate_interval	坐标间距	20
y_axes_distance	纵坐标字体离坐标的间隔	10

2.3.3 UI 模块

2.3.1 Canvas 类

Canvas 类继承 Qwidget，用于在 Canvas 上绘制曲线和坐标点等界面信息。
初始时有以下属性：

```
### 要绘制的数据和每个点之间的间隔
self.data = data
### 处理后的数据
self.new_data = None
### 坐标间隔
self.interval = interval
### 信号
self.my_sender = None

### 布局，网状
self.mainlayout = QGridLayout(self)
self.initial = True
```

Canvas 类拥有一下方法进行绘制：

函数名	paintEvent
输入	无
返回	无
说明	画图事件，每次 update 都会进入，需要的是在更换间隔和数据时进行刷新。分别画线和画点。并且根据 my_sender 信号的类型刷新 UI，重新画图，包括新的曲线等。

函数名	update_canvas
输入	无
返回	无
说明	信号槽，调用 paintEvent 方法

函数名	drawPoints
输入	qp:画笔
返回	无
说明	绘制坐标和点，不作任何处理

函数名	drawLines
输入	qp:画笔
返回	无
说明	绘制曲线，不作任何处理

2.3.1 ApplicationWindow 类

ApplicationWindow 类继承 QMainWindow，是应用主窗口，接受用户的各种操作，并传给 Canvas 类进行刷新，包括打开文件、读取文件、刷新 UI、保存图片、选择数据操作等。

初始时有以下属性：

```
'''
菜单栏UI
'''
bar = self.menuBar()
file = bar.addMenu("File")

open_file_action = QAction("Open", self)
file.addAction(open_file_action)
## 设置 open 按钮的槽
open_file_action.triggered.connect(self.open_file)

save_image_action = QAction("Save", self)
## 添加快捷键
```

```
        save_image_action.setShortcut("Ctrl+S")
        file.addAction(save_image_action)
        #### 槽
        save_image_action.triggered.connect(self.save_image)



        #### 绘图界面
        self.scrollArea = QScrollArea(self)
        # data = utils.read_file("/Users/HZzone/Desktop/test1.py")
        data = []
        self.canvas = Canvas(data=data, interval=config.default_interval,
parent=self)
        self.scrollArea.setWidget(self.canvas)
        self.scrollArea.setWidgetResizable(False)
        self.scrollArea.widget().resize((len(data)+2)*config.default_interval,
config.channels*config.min_channel_size)
        # self.scrollArea.widget().
        layout = QGridLayout(self)
        layout.addWidget(self.scrollArea, 0, 0)
        w = QWidget()
        self.slider = QSlider(Qt.Horizontal, w)
        layout.addWidget(self.slider, 1, 0)
        self.slider.valueChanged.connect(self.canvas.update_canvas)
        w.setLayout(layout)
        self.setCentralWidget(w)

    self.setMinimumHeight(config.channels*(config.min_channel_size+config.window_
min_size))
        self.setMaximumWidth(2*config.min_channel_size)

        '''
        定义各种操作，显示在第二个通道中的图形
        '''
        options = bar.addMenu("Options")
        sin_action = QAction("sin", self)
        sin_action.triggered.connect(self.canvas.update_canvas)



        cos_action = QAction("cos", self)
        cos_action.triggered.connect(self.canvas.update_canvas)

        options.addAction(sin_action)
        options.addAction(cos_action)
```

```
#### 退出，添加退出快捷键
quit_menu = bar.addMenu("Quit")
quit_action = QAction("Quit", self)
quit_action.setShortcut(Qt.CTRL + Qt.Key_Q)
# quit_action.triggered.connect(quit)
quit_menu.addAction(quit_action)

### 所有菜单栏的槽，输出按的键
file.triggered[QAction].connect(self.processtrigger)
options.triggered[QAction].connect(self.processtrigger)
    self.setWindowTitle("Chart System")
```

函数名	open_file
输入	无
返回	无
说明	打开文件，用户选择

函数名	save_image
输入	无
返回	无
说明	保存图片，并让用户选择路径

3 测试与界面

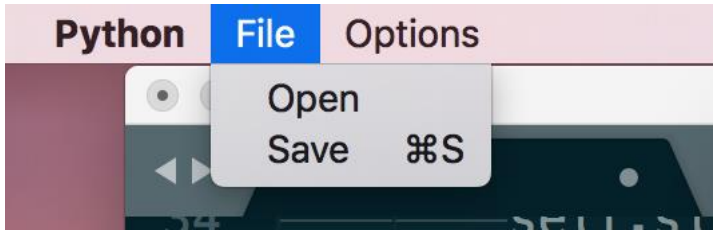
3.1 测试目的

确保Chart System能够正常运行，各个模块都能完成其功能。

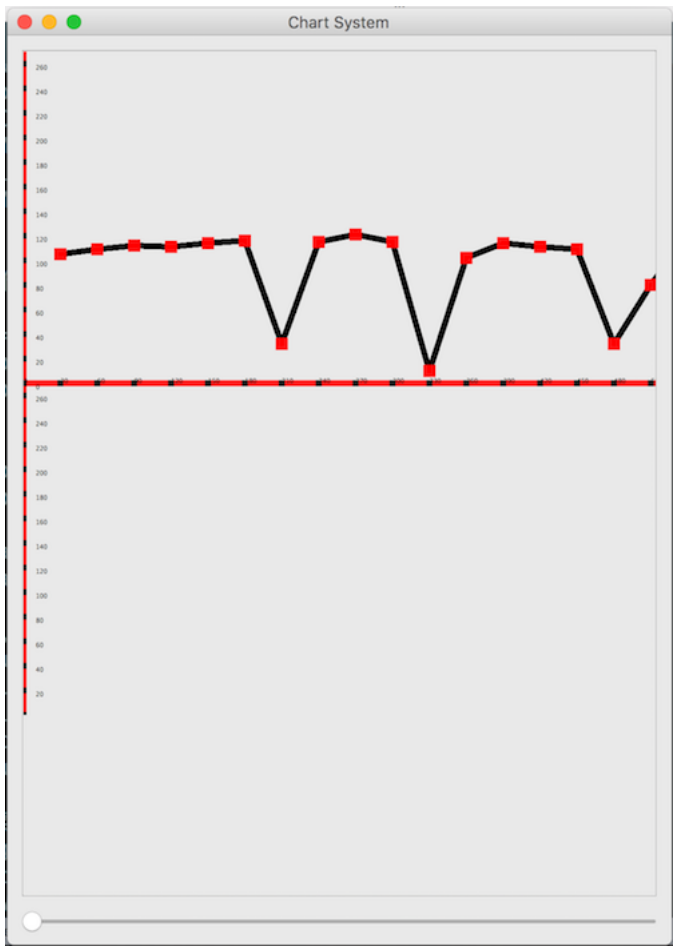
3.2 测试流程

打乱操作顺序对软件的UI功能进行随机测试，对utils库中非UI则编写测试用例进行测试

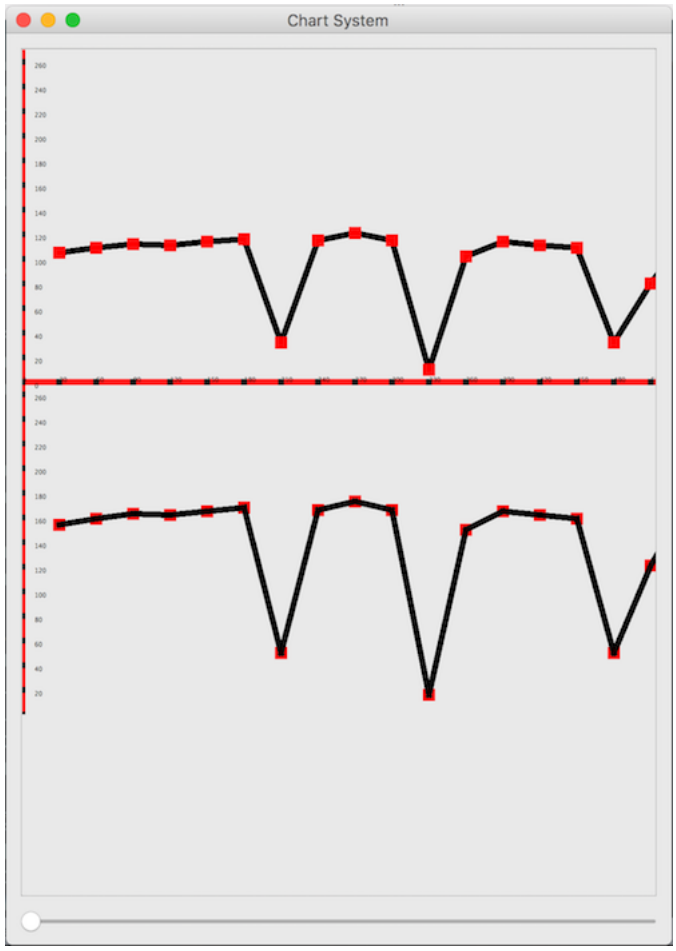
左右滑动触摸板对窗口进行滑动查看，底部滑块对通道进行放大和缩小，顶部菜单栏选择各种操作。



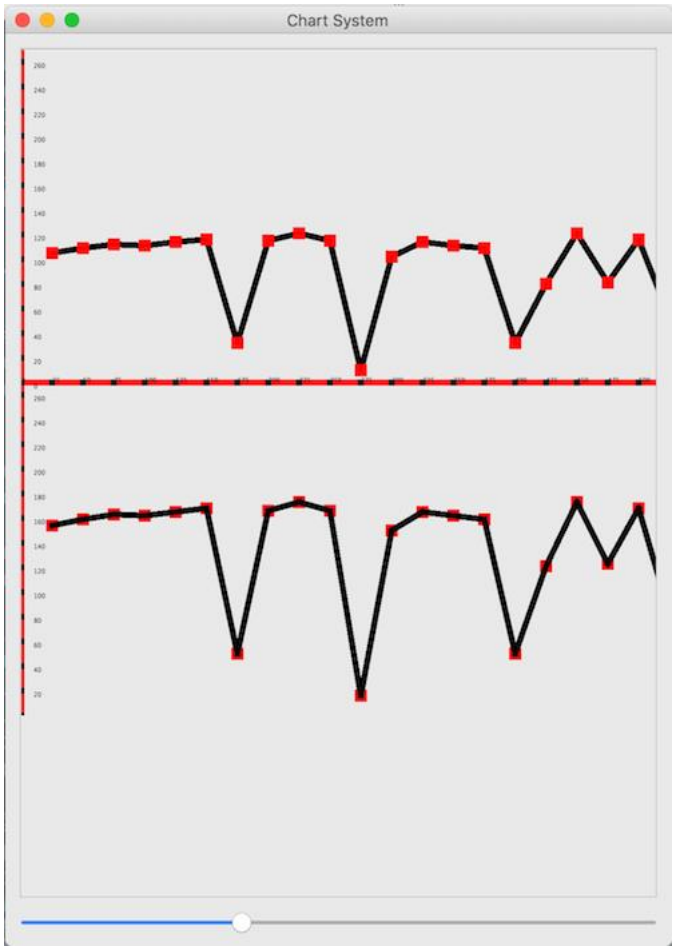
3.2.1打开文件



3.2.2数据操作



3.2.3放大和缩小



3.2.4滑动

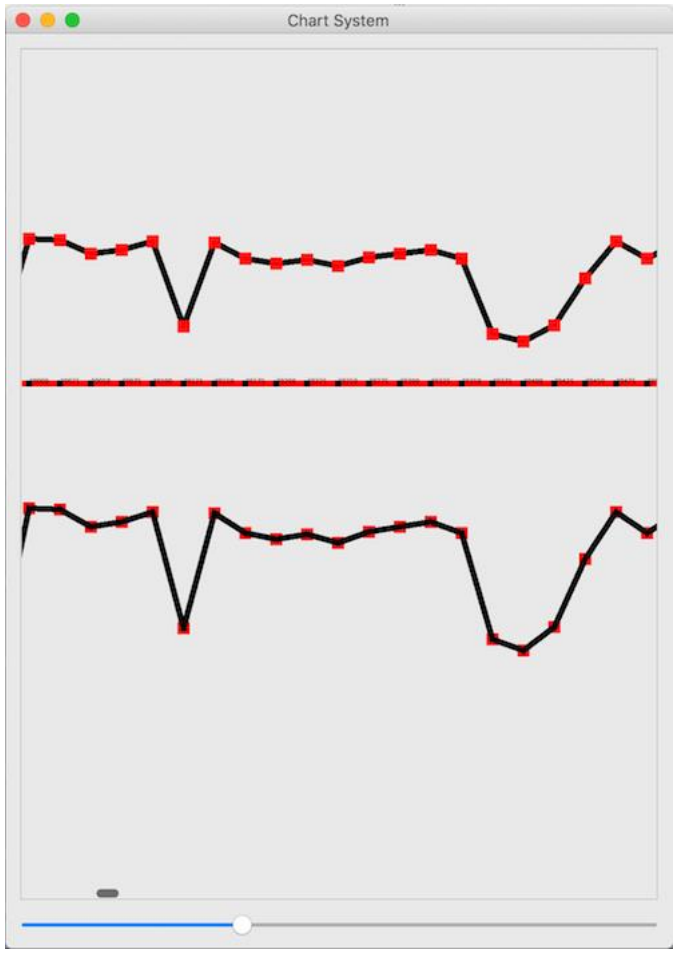


Chart System