# 目录

[一 目录 1](#_Toc23245046)

[二 设计课题 2](#_Toc23245047)

[1 项目概述 2](#_Toc23245048)

[2 功能需求 2](#_Toc23245049)

[3 选题原因 2](#_Toc23245050)

[4 开发环境 3](#_Toc23245051)

[4.1 开发环境介绍 3](#_Toc23245052)

[4.2 开发环境下载、安装和配置 3](#_Toc23245053)

[5 工程文件的目录结构和文件 4](#_Toc23245054)

[6 程序的难点和核心技术 4](#_Toc23245055)

[6.1 程序的难点 4](#_Toc23245056)

[6.2 核心技术 5](#_Toc23245057)

[三 设计思路 5](#_Toc23245058)

[1 总体思路 5](#_Toc23245059)

[1.1 图形设计思路 5](#_Toc23245060)

[1.2 画布设计思路 6](#_Toc23245061)

[1.3 操作设计思路 6](#_Toc23245062)

[1.4 吸附操作设置 6](#_Toc23245063)

[1.5 保存读取设置 6](#_Toc23245064)

[2 详细设计 6](#_Toc23245065)

[2.1 具体流程 7](#_Toc23245066)

[2.2 界面设计 8](#_Toc23245067)

[2.3 图形设计 10](#_Toc23245068)

[2.4 画布设计 20](#_Toc23245069)

[2.5 操作设计 22](#_Toc23245070)

[四 运行和测试过程 22](#_Toc23245071)

[五 使用说明 25](#_Toc23245072)

# 设计课题

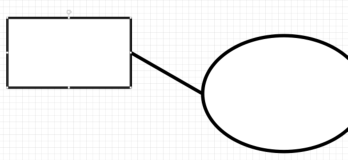
## 项目概述

实现一个流程图绘制软件，可以放置不同的图形，可以进行连线，以表示流程过程等；

## 功能需求

3.2流程图设计工具：

* 类似visio 或word 中绘图的功能。支持绘制“直线，矩形，圆角矩形，菱形，圆，梯形，单向箭头，双向箭头，文本信息”等9 个基本图形；
* 用户可以随意选择上述8 个基本图形，用鼠标在窗口中绘制出来（可以使拖拉出来或直接放置）。可以用户鼠标选择窗口中已经绘制好的图形，改变它的位置和/ 或大小。图形可以增加文本注释，注释位置默认在左上角，但是可以被用户用鼠标拖动位置；
* 窗口的图形被选中后出现4-8 吸点，以便用户用鼠标拖拉，改变大小。注意示意图中，矩形被选中后，出现了8 个吸点；
* 当“线条”或“箭头”连接一个图形的时候，其端点只能连接到图形的吸点上。如图所示。且连接的时候，当线条靠近吸点的时候，软件能自动完成连接；
* 图形颜色可设置；
* 流程图可保存；

 

注意示意图中，矩形或椭圆被选中后，出现了8个/4个吸点。

## 开发环境

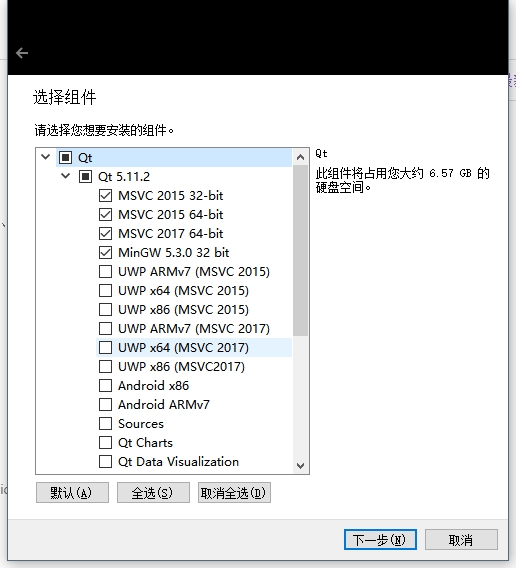
### 开发环境介绍

使用了QT Creator 4.7.1可视化IDE和QT 5.11.2开发库，QT由Qt Company开发的跨平台C++图形用户界面应用程序开发框架。它既可以开发GUI程序，也可用于开发非GUI程序，比如控制台工具和服务器。Qt是面向对象的框架，使用特殊的代码生成扩展以及一些宏，Qt很容易扩展，并且允许真正地组件编程，其实现的信号槽机制很好的简化了VC++中消息传递的设计；

### 开发环境下载、安装和配置

QT Creator下载目录：<http://download.qt.io/archive/>

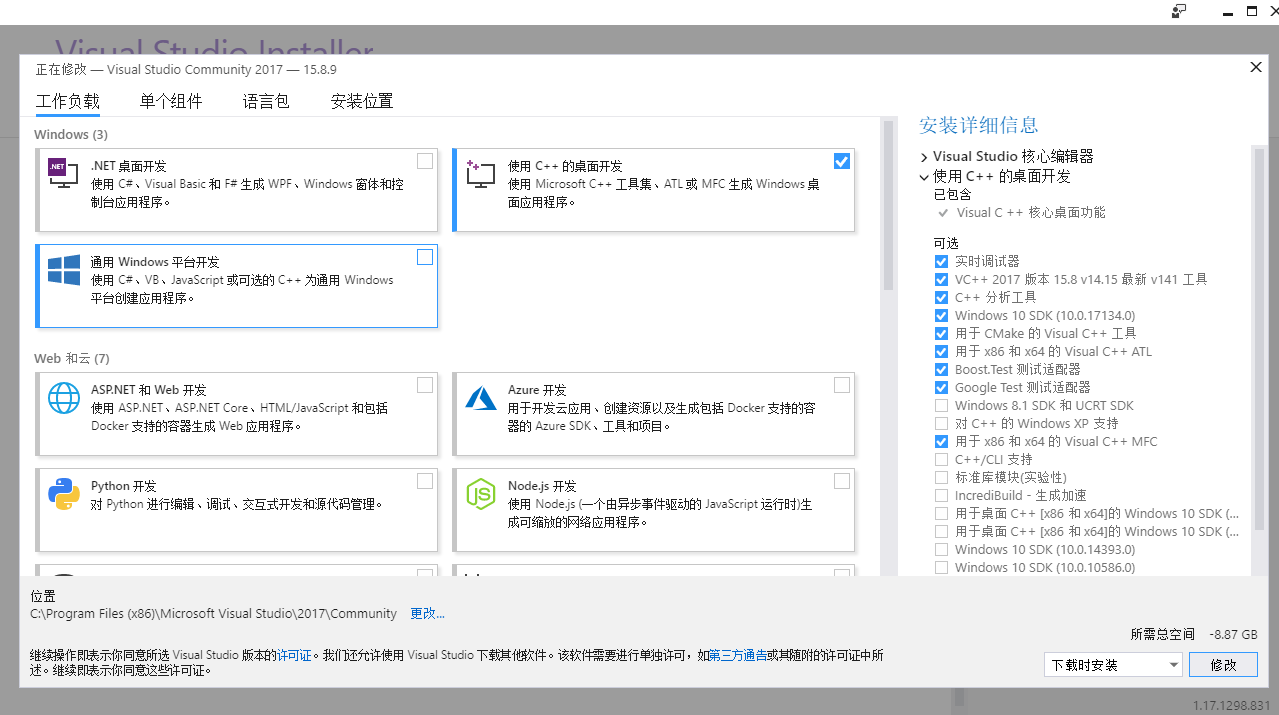
选择qt/5.11/5.11.2/目录下的.exe文件，在windows下安装即可，安装时必须勾选MSVC2017；



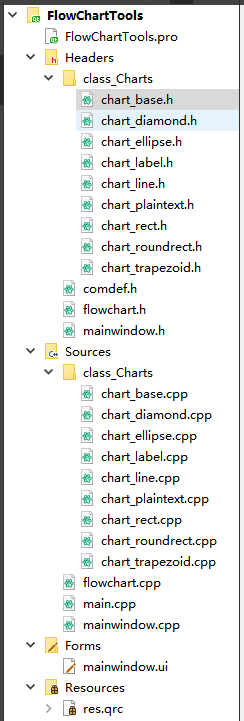
如果需要调试，那么必须安装VS2017；

VS2017下载地址：<https://visualstudio.microsoft.com/zh-hans/downloads/>

VS下载安装需要勾选C++部分；



## 工程文件的目录结构和文件



## 程序的难点和核心技术

### 程序的难点

图形基类的设计、图形绘制的设计、磁力点（用于吸附的点）属性的设计、大小点（改变大小的点）属性的设计、鼠标事件的设计、重绘事件的设计、文本的设计、连线线型的设计、连线自动吸附的设计、画布的设计、工具栏的设计、信号事件触发的设计、文件读取保存设计等；

### 核心技术

1. 更方便的创建各种图形，由于基类已经设计好，因此如果用户想定义自己的类型，只需要提供几个必要的信息即可，包括磁力点个数（默认4）、大小点个数（默认4）、图形名称、图形绘制形状、磁力点位置函数（可选）、大小点位置函数（可选）等信息即可
2. 提供了对图形的管理容器，可以很方便的添加删除图形；
3. 由于画布定义了键盘事件，因此如果要增加快捷键，将非常容易；

# 设计思路

## 总体思路

QT官方库中包含了一个实现图形绘制、操作等的QGraphicsScene类，这个类是一个界面类，用于管理和显示QGraphicItem对象，其所包含的操作有：平移、缩放、旋转、碰撞检测等，并且可以同时对多个QGraphicItem对象进行相同的操作，QGraphicItem对象可以自定义图形形状，然而一个棘手的问题就是，QGraphicItem类并没有实现改变大小、吸附连线等，而且据文档所说和网上查阅，要实现改变大小，需要重写其自定义的绘图事件和鼠标事件，特别是绘图事件，这将改变图形的很多数据成员信息，所以虽然这个官方库功能强大，特别是有碰撞检测，但是应用场合并不是用来做一般的图形绘制，而是更高级的类似游戏和CAD类的软件，所以决定自己设计大体框架；

从QT的QGraphicsScene类和QGraphicItem类的源码中我也学习到了一些东西，源码可见（Qt根目录\5.11.2\Src\qtbase\src\widgets\graphicsview）。首先，QGraphicsScene类和QGraphicItem类互为友元类，因此QGraphicsScene类对象可以和QGraphicItem类对象相互访问对方的私有和保护成员而不需要调用公共接口，其次，整个框架分为了QGraphicsScene场景部件、QGraphicsView视图部件、QGraphicsItem图形部件，即整个框架分为多个部分分别进行编程设计，据此进行了一下界面设计思路构想：

应设计三个部件，左边的图像选择部件，中心的图形绘制部件（简称画布），上部的图形信息部件；

### 图形设计思路

使用一个抽象基类来定义各种图形所共有的数据和方法，该类需要继承自另一个类，这个类可以实现绘制事件，然后从自定义的基类派生出各种不同图形的子类，其中子类主要实现绘制和其他一些异于基类的数据和方法。

抽象基类主要包含位置信息、起点终点信息、文本信息、绘制大小点的虚函数、绘制磁力点的虚函数、绘制图像的纯虚函数等；

### 画布设计思路

自定义了一个类，该类继承自另一个类，这个类需要作为图形类抽象基类的友元类，使其可以访问图形抽象基类的私有成员；这个类需要实现读取保存操作，且实现一组管理所包含图形的数据和方法；

### 操作设计思路

对于各种操作，实现思路是重载事件处理函数和设置响应事件；

### 吸附操作设置

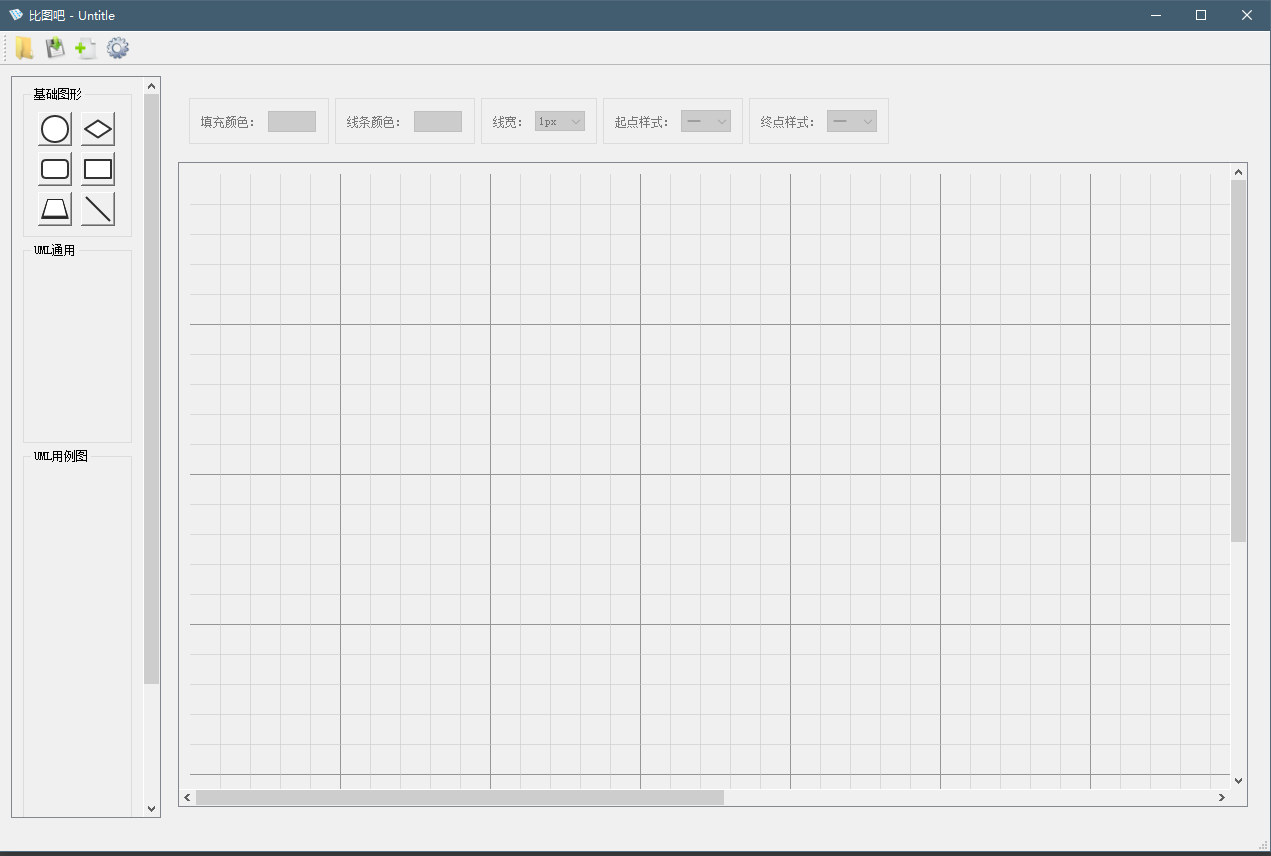
移动连线时，对画布上的非连线图形进行遍历，当处于图形磁力点一定范围内时，实现自动吸附功能，即直接改变连线的相对做到，其值为相应磁力点的坐标；

### 保存读取设置

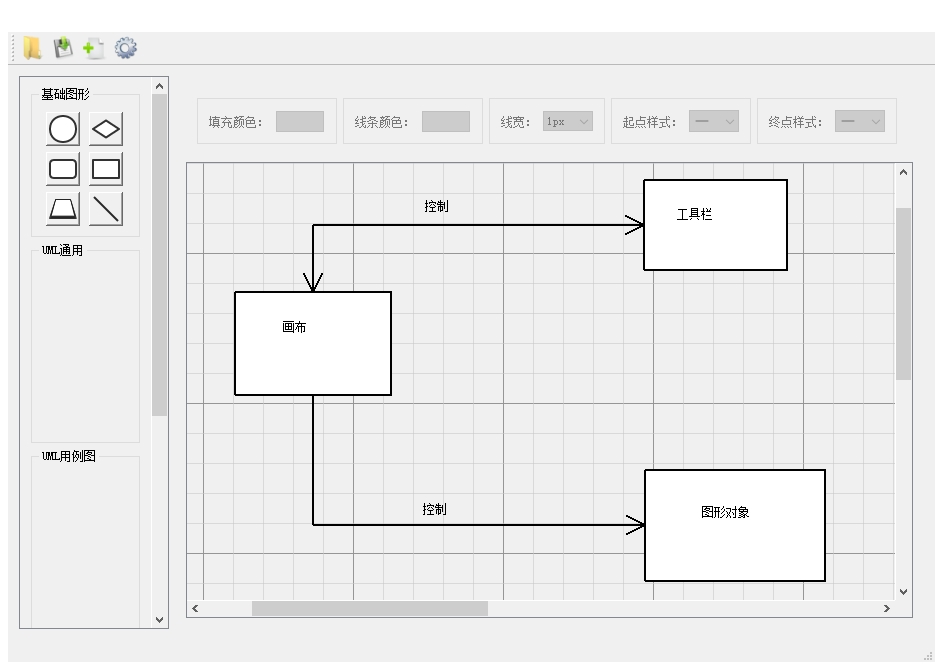
由于此工程包含多个类，因此对每个类实现了输出功能，当触发读取（保存）事件时，画布将调用相应类的保存（读取）函数，如果类有递归嵌套，则会递归调用相应的保存（读取）函数，最终实现保存（读取）；

## 详细设计

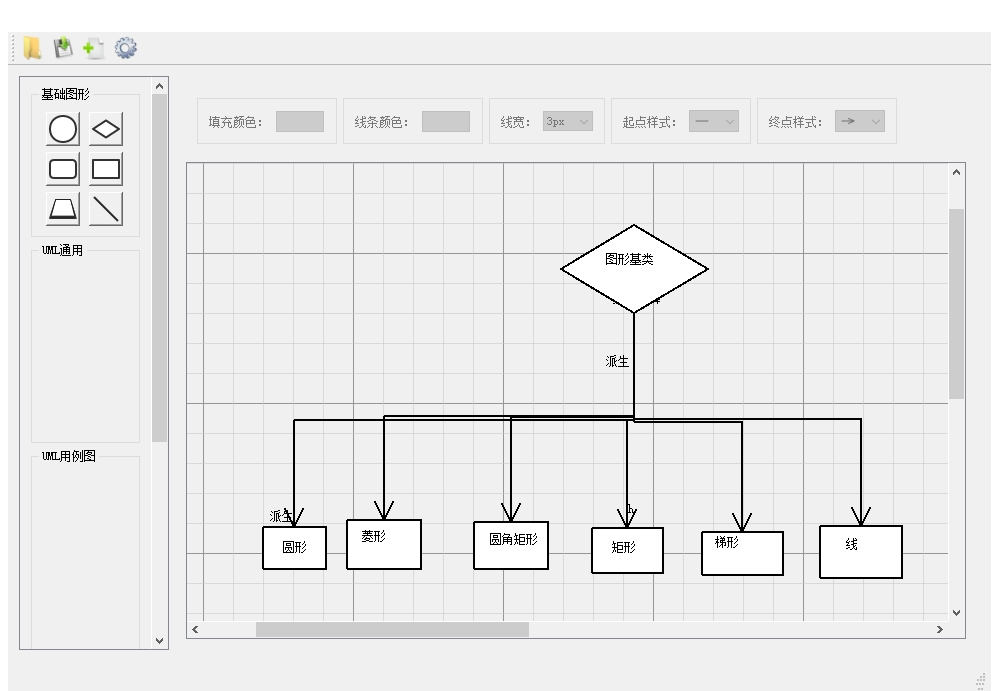
界面：



### 具体流程



三个组件的关系如图所示；画布和工具栏相互控制，画布用于设置工具栏显示的信息，工具栏用于向画布通知改变的信息；画布单独控制图形对象，包括管理图形对象、改变图形对象信息等；



各种图形和基类的关系，其中有一个图形基类，所有图形都是由该基类派生，其中需要定义一些不同图形不同的属性，主要是绘制属性，而对于线这种类型，其需要定义的东西较多，因为线不仅仅是一个图形，更需要和其他图形进行组合，实现流程图的性态；

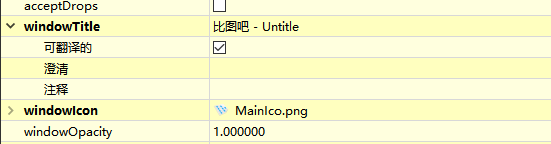
### 界面设计

除去工具栏还包括3个其他控件，左边的图像选择部件，中心的图形绘制部件（简称画布），上部的图形信息部件，如下图所示：



#### 主窗体

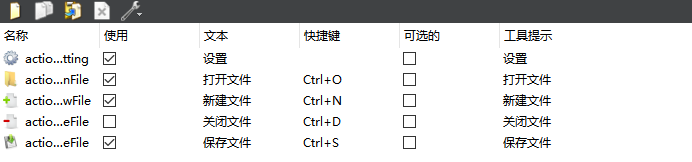
主窗体（QMainWindow类对象）设置了水平自动布局（QHBoxLayout类对象），使得部件可以更好地充满整个窗体，同时设置了窗体的名称（windowTitle属性）和图标（windowIcon属性），分别如下图所示：



#### 工具栏

工具栏添加了如图所示的几个工具（QAction类对象），分别是读取、保存、新建、设置，其都设置了快捷键、图标和文本提示信息，这些工具用于点击事件后执行相应的操作；





#### 图像选择部件

图像选择空间包含一个可以滚动的滚动区域部件（QScrollArea类对象），这个部件自动布局为垂直布局，次部件包括了三个组合框（QGroupBox 类对象），第一个名称为 “基础图形”，第二个为“UML通用”，第三个为“UML用例图”，其中第一个基础图形组包含了6个按钮（QPushButton类对象），此组合框设置为栅格自动布局，其中6个按钮分别设置了图标（通过styleSheet属性），清空了文本内容，并且设置了固定大小，使其更美观，这些按钮用于实现鼠标点击后新建对应图形的功能；

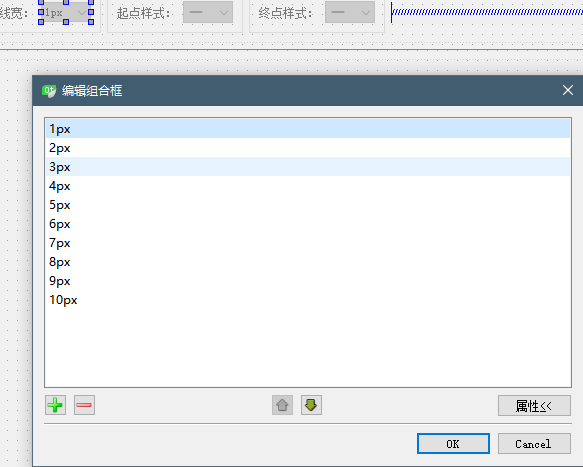
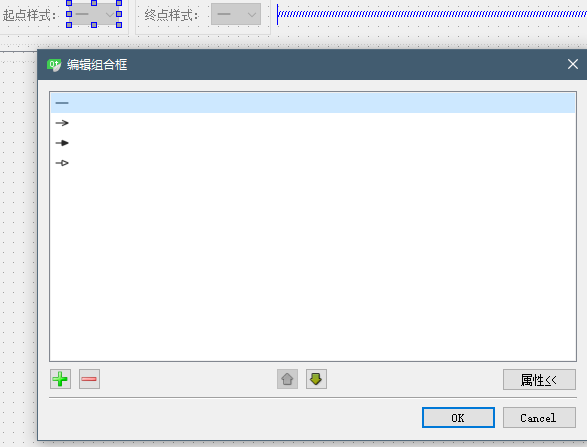
#### 图形绘制部件

图形绘制部件主要由一个滚动区域部件组成，其中包含了一个QWidget部件，但是此QWidget部件进行了类型提升，提升为了一个自定义的类，该类继承自QWidget类，名称为flowChart\_Widget，这个类将实现图形绘制、图形管理等功能；

#### 图形信息部件

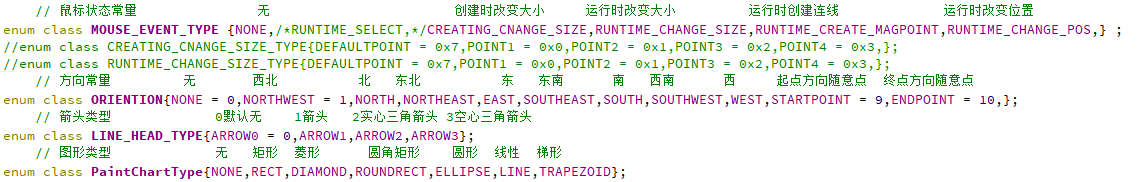
图形信息部件包含一个QWidget，其中放置了5个组合框，最右边有一个水平间隔部件，用于将几个组合框顶到左边；每个组合框都包含了一个标签（QLabel类对象）和一个按钮或者列表框（QComboBox类对象），默认5各组合框设置为disabled状态，只有运行时选中图形或连线才能激活某一部分组合框；易知前面两个用于设置颜色信息，后面三个用于设置线条信息，两个颜色设置按钮使用styleSheet属性来改变显示的颜色，同时点击将会产生颜色选择对话框（QColorDialog类对象）；线宽设置了线宽的可选数据，选择相应的内容将会触发设置线宽事件，从而改变线框；后面两个线样式提供了图形化的可选样式信息，其中每一个设置了图标，同样的点击会触发设置样式事件；



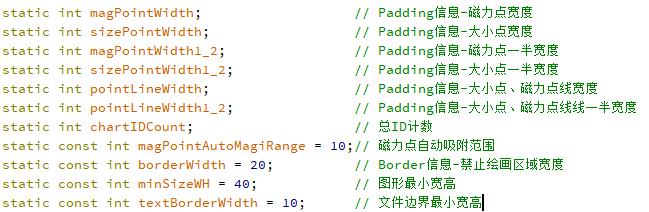
### 图形设计

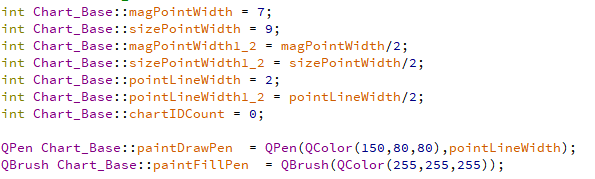
#### 基本信息

所有文件都有一个公共的头文件comdef.h，此文件包含一些常量的定义，如下所示；

#### 图形基类

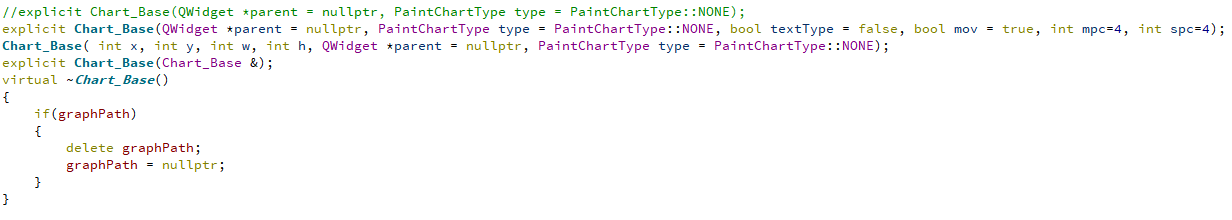
* 类名Chart\_Base，基类QWidget，注意，此类为纯虚基类，无法实例化；
* 静态常量、静态变量：





如上为基本常量和静态变量，包括一些基本信息，还有两个笔刷类型和填充类型静态变量，用于设置边框颜色和填充颜色，其中线型无法使用填充颜色；

* 构造函数和析构函数



这里定义了复制构造函数，可以用于图形的复制粘贴操作，不过因为时间原因这里没有实现；显式默认构造函数有多个参数，其中mpc表示磁力点个数，spc表示大小点个数，mov表示图形是否可移动，一般不适用；textType表示文本类型，线型和其他图形不同，还有一个指示图形类型的变量type，最后就是图形父控件parent；

* 私有数据成员：



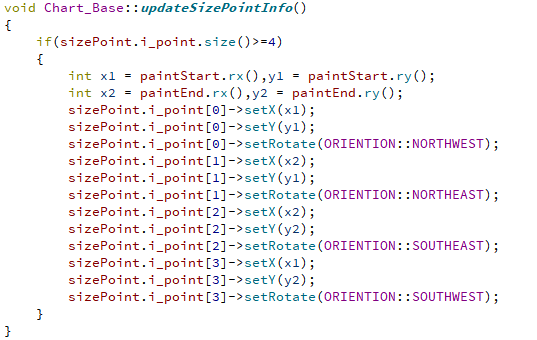
除了一些基本变量外还有一个文本类型，文本类型使用了共用体定义了基本的文本对象，其中tmpEdit表示编辑时显示的编辑框；其中文本框在基本图形上都为PlainText类型，而线型为Label类型，不过基本图形可以设置为Label类型，线型设置为PlainText会不太美观； 同时还重定义了<<和>>运算符，用于保存时使用；

* 私有成员方法



* 如下为大小点、磁力点更新函数：

这两个函数都为虚函数；默认为4个大小点，分别对应4个边角位置，除了线型是2个外，其他都是4个大小点，大小点包括x y坐标以及自身的方向信息；默认4个磁力点分别对应4个中点位置，线型无磁力点，矩形和梯形为8个磁力点；



* 保护数据成员：



这里主要包含了点信息的定义，其中大小点继承自一个点公共基类i\_pointbase，其中定义了点的位置、点的范围还有点的朝向；而磁力点中间还有一个过度类，其中定义了图形包含的起始线的指针和终止线的指针，通过这个设定，就可以实现改变图形位置同时改变线的位置；

这里还定义了诸如图形形状graphPath的指针对象、起始点终止点范围、颜色等信息；

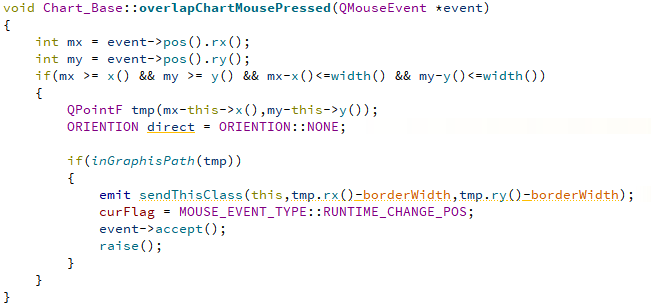
* 公有成员函数：



公有数据成员大部分都被画布直接或间接调用，主要用于设置图形的信息；公有成员还重定义了<<和>>运算符，用于实现读取保存；图形的ID值是唯一的，ID值用于读取时连线用；

* 鼠标按下事件重叠处理函数如下图所示；

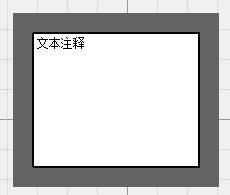
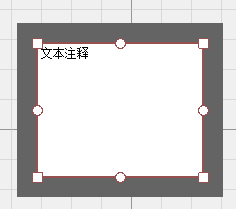
因为多个图形可能重叠，因此需要在一个图形放弃鼠标事件时将事件传递给其他图形，默认事件传递只在子控件和父控件之间传递，即如果某个图形不接受事件，则该事件将传递给画布，这会导致其他图形不会处理，这里直接分析一遍鼠标事件跟其他图形的关系，类似鼠标事件函数；同样的鼠标移动事件也有一个对应的重叠处理函数；



* 改变大小和位置函数如右图所示：

该函数用于设置图形的大小，并重新绘制图形，首先x y分别是鼠标的位置，然后根据widgetStart和widgetEnd中x和y的大小来设置需要改变的相应的大小；默认widgetStart是建立图形时按下的那个点，而widgetEnd是释放鼠标时的那个点，所以两个点大小不一定，必须进行比较；

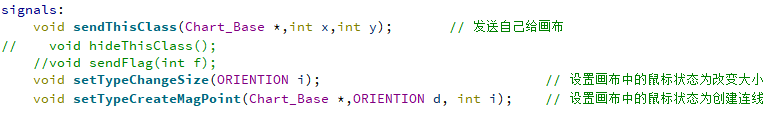
比较晚后，根据传入的type参数指定的方向来进行大小改变，图形有两层绘制范围，一层是图形边界，一层是绘制边界，如下第一个图所示，其中包括全灰色的部分就是整个图形范围，而白色部分就是绘制范围，如果不这么设定，那么大小点和边界点的绘制将超出图形的范围，如下第二个图所示，课件大小点和磁力点的红色范围已经超过了绘制范围；基本图形的只包括两个范围，而线型包括三个范围，还有一个线头绘制范围；

 大小计算结束后，进行更新，后面的各种update函数就是用于更新图形信息的函数，包括更新位置信息、绘图信息、大小点位置信息、大小点范围信息、磁力点位置信息、磁力点范围信息、磁力点上连线的位置信息、文本位置信息等；

相对来说，改变位置信息的函数就简单的多，只需要更改控件的位置即可，不需要更改具体内部信息；

* 信号成员：



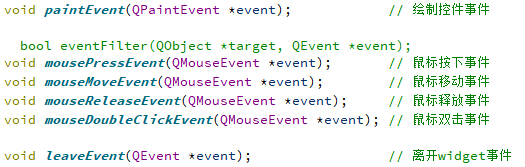
这几个信号用于在发生事件时给画布传递参数；如点击事件则将调用sendThisClass函数，告诉画布这个图形被点击了，注意接下来可能的改变位置的事件；

* 槽成员：



主要用于设置是选中、非选中、鼠标悬停、鼠标非悬停等状态；还有两个槽用于改变文本的位置；

* 保护成员事件：





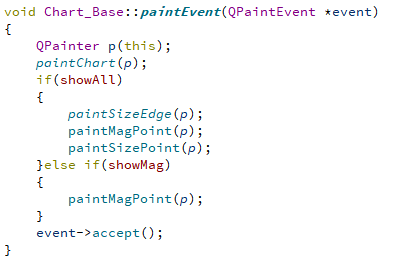
* 鼠标按下事件函数如右图所示；

如果图形已选中并且鼠标位置在大小点上，则设置当前图形的鼠标状态为改变图形大小状态；如果图形已选中或者在图像范围内，且鼠标位置在磁力点上，则设置当前图形的鼠标状态为创建连线状态；如果鼠标位置在图形中，则设置为改变图形位置状态；否则，当前图形的鼠标状态设置为无

* 如下图为鼠标移动事件处理函数；

根据鼠标位置进行鼠标图形的设置，每个函数都返回一个索引值，索引值标识着方向信息，根据方向设置鼠标状态；



* 绘制事件如下图所示

paintChart为图形的默认形状，必须绘制；

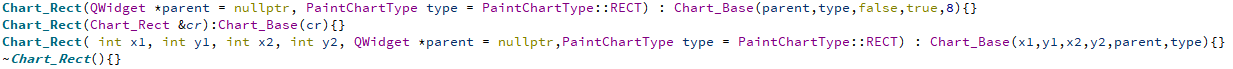
paintSizeEdge为图形的边框信息，默认只有在选中时绘制，且此函数为虚函数，其中line型重定义为空函数；

paintMagPoint为图形的磁力点绘制函数，用于绘制磁力点，在选中和鼠标悬浮时显示；

paintSizePoint为图形的大小点绘制函数；

#### 矩形

* 类名Chart\_Rect，基类Chart\_Base
* 构造函数、析构函数：



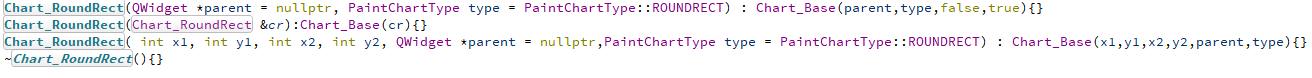
可见，这里设置了8个磁力点，如果不适用默认磁力点个数，则必须重定义updateMagPointInfo函数；

* 私有函数：



#### 圆角矩形

* 类名Chart\_RoundRect，基类Chart\_Base
* 构造函数、析构函数：

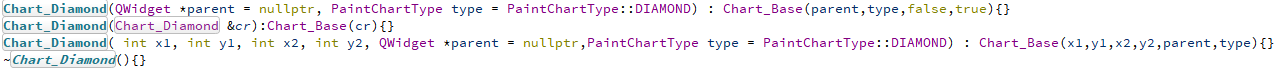


* 私有函数：



#### 菱形

* 类名Chart\_Diamond，基类Chart\_Base
* 构造函数、析构函数：



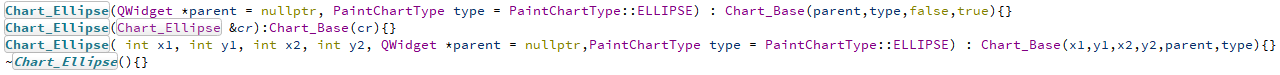
* 私有函数：



#### 圆形

类名Chart\_Ellipse，基类Chart\_Base

* 构造函数、析构函数：



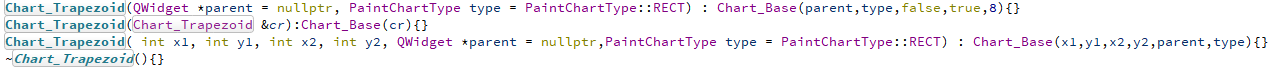
* 私有函数：



#### 梯形

类名Chart\_Trapezoid，基类Chart\_Base

* 构造函数、析构函数：



这里也是8个磁力点；

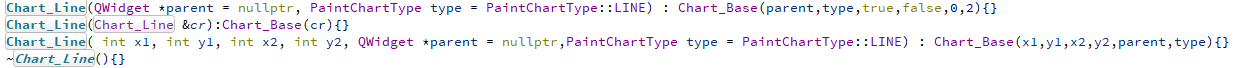
* 私有函数：



#### 线

类名Chart\_Line，基类Chart\_Base

* 构造函数、析构函数：



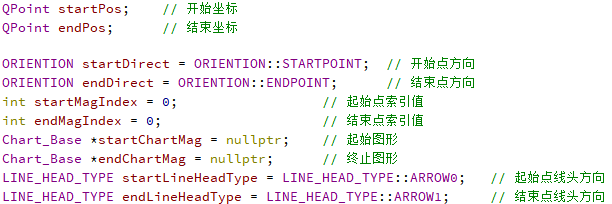
这里没有磁力点，只有2个大小点，且线是不可移动的；

* 静态数据成员、静态常量：



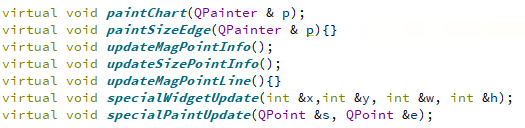
表示了线头的绘制宽度（默认都是正方形）、线头长度、宽度、线头柄长度；

* 私有数据成员：



这里定义了线的开始位置和结束位置，注意这里和图形基类Chart\_Base中的paintStart、widgetStart并不相同；这里还定义了线头所连图形、线尾所连推行、线头所连磁力点磁力点索引值、线尾所连磁力点索引值、线头方向、线尾方向等；

* 私有虚函数：



需要重定义很多函数；

* 私有成员函数：



用于绘制连线和线头样式的函数；

* 如下为绘制线头样式的函数的一部分：

由于每个位置的方向都不一致，所以需要每个方向都定义一遍；



* 公有成员函数



主要用于对点信息的操作；最后还重定义了<<和>>运算符，用于读取和保存

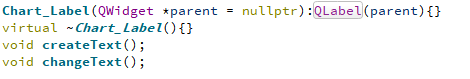
* 信号成员：



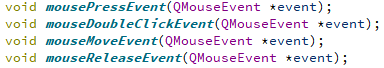
#### 简单文本

类名Chart\_Label，基类QLabel

* 构造函数、析构函数：



* 保护事件函数：



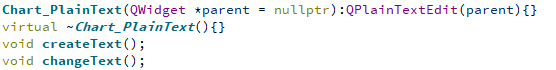
* 信号方法



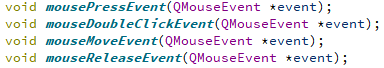
#### 富文本

类名Chart\_PlainText，基类QPlainTextEdit

* 构造函数、析构函数：



* 保护事件函数：



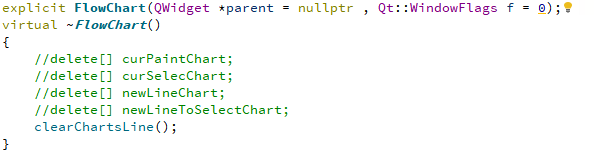
* 信号方法



### 画布设计

类名FlowChart，基类QWidget

* 构造函数、析构函数：

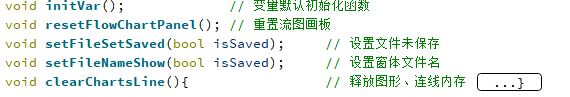


* 私有数据成员：



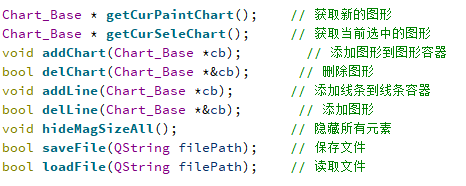
类型说明如上，其中包括两个容器，一个是图形容器，一个是线型容器，两个容器单独工作，不过都是基类指针类型，得益于多态性，可以让基类指针指向子类对象；

* 私有成员函数：



此部分主要用于初始化以及读取保存文件所用到的私有函数；

* 公有成员函数





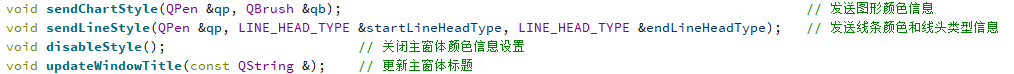
此部分包括添加删除图形连线、读取保存文件等；

* 删除图形的函数如下所示；

删除图形除了要从图形容器中删除图形，还要递归到图形中删除与该图形所有连线，同时，还要设置连线的另一端，使其所连的线从线容器中删除，因此逻辑有一点复杂；



* 信号成员：



主要用于改变主窗体的信息；

* 槽成员：

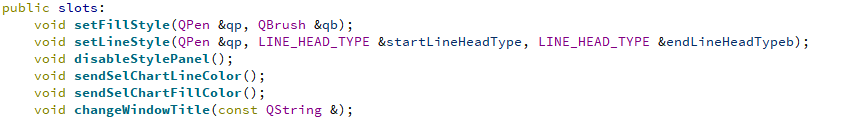


前面8个信号为主窗体的操作；后面几个setPaintChart为每种图形一个，也为主窗体的操作，必须为每个图形定义，作用为点击按钮创建一个图形；

### 操作设计

类名MainWindow，基类QMainWindow

* 信号函数：



与画布操作的函数；

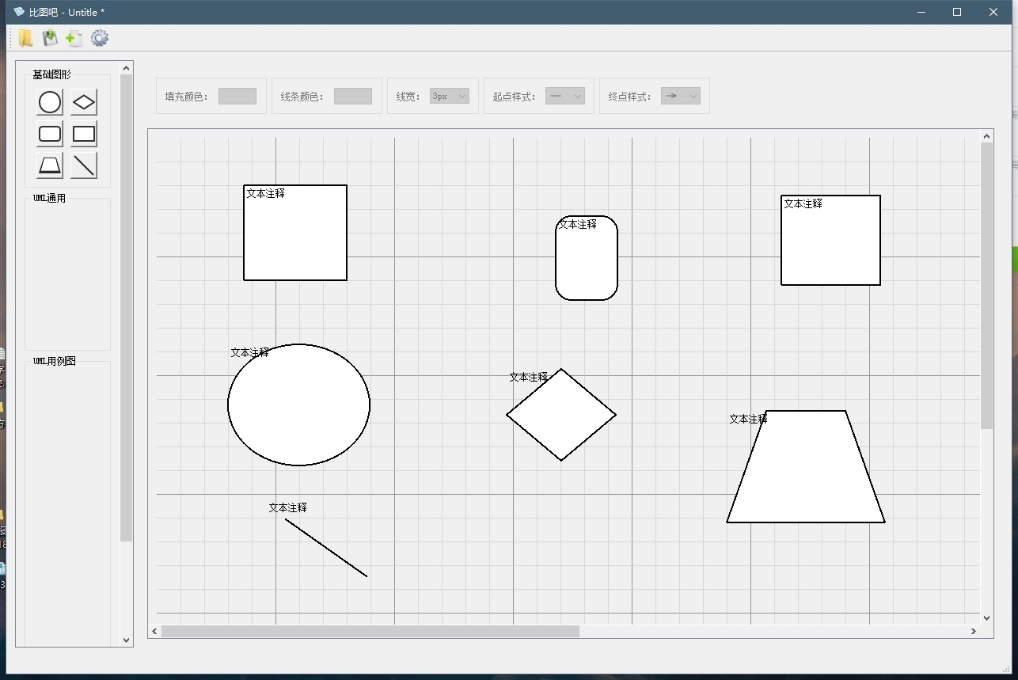
* 具体信号连接实现：



全部为与画布的信号槽连接

# 运行和测试过程

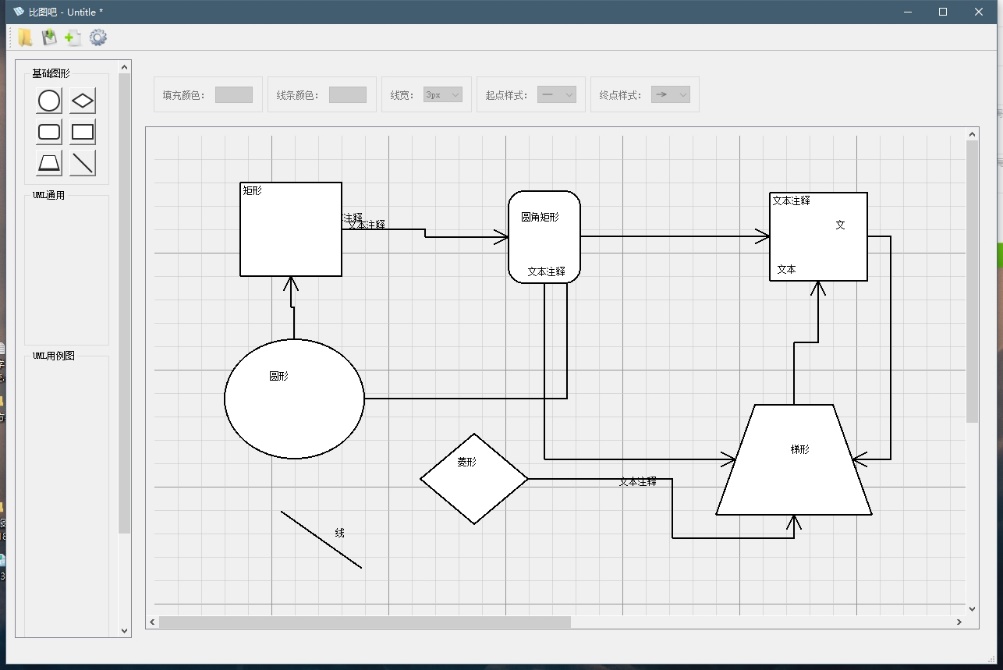
创建图形：



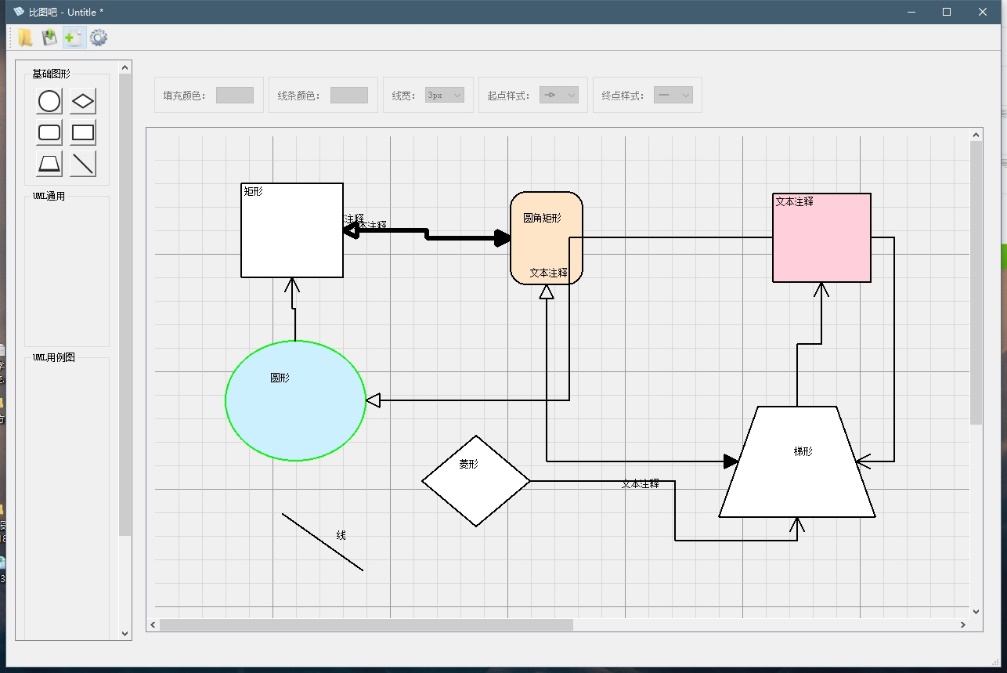
更改图形文本信息：



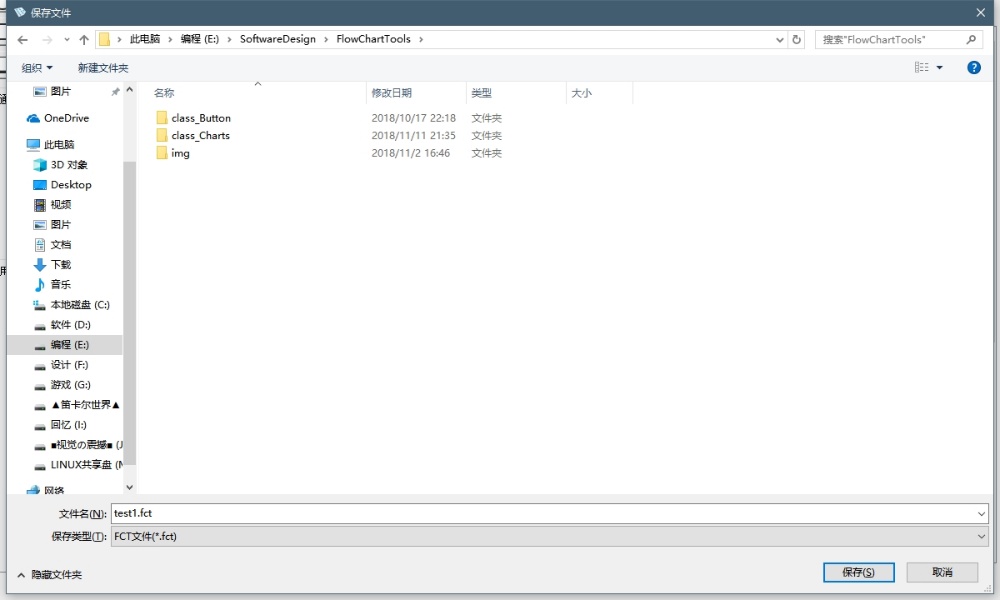
进行连线：



设置图形的各种信息：



保存图形：

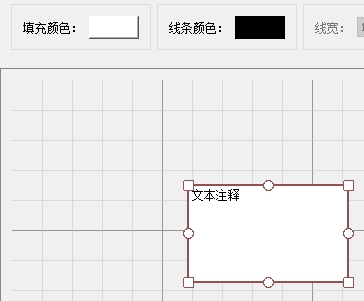
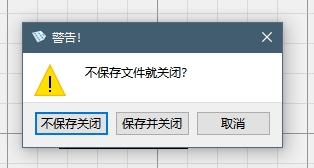


打开图形，可以看到标题栏的改变：





# 使用说明

1. 点击一个图形按钮，然后在画布上创建图形；创建好后可以调整图形的大小、位置等信息，如果不想要图形，可以点击delete按键删除；
2. 鼠标移动到图形时会有磁力点出现，图形被选中时也会出现磁力点，鼠标移动到图形的不同位置鼠标的图标会发生改变，当编程十字时，就表明可以创建连线，当然，也可以使用连线按钮创建连线；
3. 移动连线，到任何图形上的磁力点附近将自动吸附，松开鼠标则完成连线；
4. 点击连线可以改变连线的属性，分别是线条颜色、线宽、线头类型、线尾类型；点击图形可以改变图形的属性，分别是填充颜色、线条颜色；D:\Users\ziteng\Desktop\2018-11-11_22-50-57.jpg
5. 点击保存按钮D:\Users\ziteng\Desktop\2018-11-11_22-48-00.jpg可以进行保存，如果是新文件，则会弹出输入文件名的对话框；点击打开按钮D:\Users\ziteng\Desktop\2018-11-11_22-47-46.jpg会打开一个新文件，如果当前文件未保存，则会弹出提示框；同样的，点击新建按钮D:\Users\ziteng\Desktop\2018-11-11_22-49-42.jpg如果未保存一样会弹出提示框；