**“C++程序设计与训练”课程大作业**

**项目报告**

**项目名称：**

**简易绘图工具的制作**

**姓名：** 邵键准

**学号：**2014011517

**班级：** 自44

**日期：** 2015-9-20

目 录

[1系统功能设计 4](#_Toc430512166)

[1.0前言 4](#_Toc430512167)

[1.1总体功能描述 4](#_Toc430512168)

[1.2功能点说明 6](#_Toc430512169)

[2系统总体结构 10](#_Toc430512170)

[2.1概要设计 10](#_Toc430512171)

[2.2小组成员分工 10](#_Toc430512172)

[3本人工作内容 12](#_Toc430512173)

[3.1 GUI的设计 12](#_Toc430512174)

[3.1.1 程序启动和关闭动画设计 12](#_Toc430512175)

[3.1.2 程序启动界面的设计 13](#_Toc430512176)

[3.1.3 程序主体绘图界面设计 14](#_Toc430512177)

[3.1.4 程序图标的添加 15](#_Toc430512178)

[3.2 逻辑层 16](#_Toc430512179)

[3.2.1 工作窗口WorkWindow部分 16](#_Toc430512180)

[3.2.2 启动窗口MainWindow部分 23](#_Toc430512181)

[3.2.3 颜色拾取窗口StyleSettingWindow部分 24](#_Toc430512182)

[3.3 代码版本控制 24](#_Toc430512183)

[3.4 使用手册的编写 26](#_Toc430512184)

[4项目总结 27](#_Toc430512185)

[4.1 OOP模式的优越性 27](#_Toc430512186)

[4.2 版本控制的重要性 27](#_Toc430512187)

[4.3 自主解决困难的能力 27](#_Toc430512188)

[4.4 团队合作的技能 28](#_Toc430512189)

[5相关问题的说明 30](#_Toc430512190)

[5.1 开发、测试环境 30](#_Toc430512191)

[5.2 开发语言 30](#_Toc430512192)

[5.3 使用方式 30](#_Toc430512193)

# 1系统功能设计

## 1.0前言

当初选择这么一个软件来编写纯粹是出于对玩弄文字游戏的xx管理系统的不喜，但我们没有料想到，经实践表明，涉及实时跟踪鼠标键盘事件和实时绘图的软件编写难度远大于主要通过文字实现信息交互的xx管理系统。仅仅实现一个屏幕上图形的框选功能就让我改了六七遍代码，我的队友董渊更是间断地找出了五个bug，在GitHub上刷出了一排的提交记录。当终于能够把整体功能流畅地实现时，我们对软件开发者的了解与敬意又加深了一层。

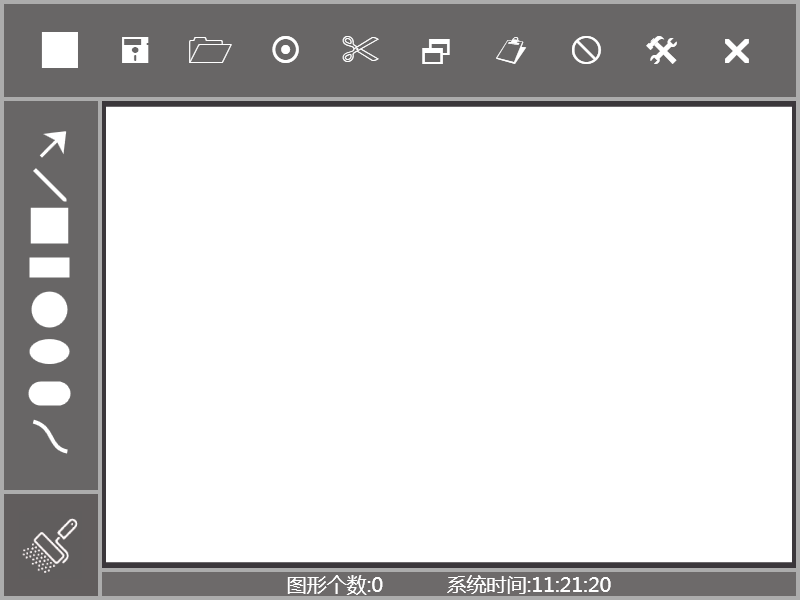
## 1.1总体功能描述

当今图像处理越发普及，人们对于图像处理的需求也各不相同。而一些绘图软件存在过于复杂（如PS）或是只具备基础功能（如windows自带画图）的问题，因此我们开发一个基于Windows Presentation Foundation（WPF）的简单绘图工具。

以下为我们程序的启动界面。



以下为我们程序的工作界面。



## 1.2功能点说明

（本部分是大作业考核和评分的主要依据）

大作业功能点说明

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **功能类型** | **功能点名称** | **按键（使用键盘或鼠标处理函数实现，参见小组成员邵键准的报告）** | **实现方式** | **功能点描述**  **（操作方法参见附录——使用手册）**  **（以下“调用类接口\*\*实现”均为自己实现的代码，参见小组成员董渊的报告）** |
| 基本功能 | 主流类型图片载入 | Ctrl+O | 自己编写C#代码 | 支持.png\.jpg\.gif\.bmp\.eps等主流图片类型的读入和加载。使用C#标准OpenFileDialog对象获得文件路径并且使用Uri读入之后转为BitmapImage，用Image控件显示。 |
| 图片保存 | Ctrl+S | 自己编写C#代码 | 支持bmp（位图）和eps（矢量图）两种格式。使用SaveFileDialog对象获得文件路径之后用filestream输出。 |
| 鼠标手绘图形 | 鼠标拖框 | 自己编写C#代码 | 包括矩形、圆角矩形、圆、椭圆、直线、贝塞尔曲线等。 |
| 单击选择图形 | 鼠标单击 | 自己编写C#代码 | 调用类接口SelectPoint来实现。 |
| 复杂功能 | 框选图形 | 鼠标拖框 | 自己编写C#代码 | 递归调用类接口SelectRect来实现。 |
| 全选图形 | Ctrl+A | 自己编写C#代码 | 调用类接口SelectAll来实现。 |
| 多次选中图形 | Shift+鼠标拖框 | 自己编写C#代码 | 调用类接口MergeComposite来实现。 |
| 被选择图形的闪烁 |  | 自己编写C#代码 | 通过来回设置选择的CompositeGraphic的isVisible属性来达到闪烁的目的。 |
| 删除图形 | Delete | 自己编写C#代码 | 调用类接口Clear来实现。 |
| 图形状态变更 |  | 自己编写C#代码 | 设置类属性DrawMode来实现。可以设置图形的边框粗细、颜色，以及内部颜色。 |
| 调色板 |  | 使用第三方库 | 支持通过ARGB属性或是在图形中直接选色的方式给图形的边框和填充分别选色。 |
| 拖动图形 | 鼠标按住拖动 | 自己编写C#代码 | 调用类接口Move来实现。 |
| 键盘移动 | Up,Down,  Left,Right | 自己编写C#代码 | 通过长按可快速移动。 |
| 剪切选中图形 | Ctrl+X | 自己编写C#代码 | 将选中的图形加入一个List，并从画布上删除选中的图形。 |
| 复制选中图形 | Ctrl+C | 自己编写C#代码 | 将选中的图形加入一个List，但不从画布上删除选中的图形。 |
| 粘贴图形 | Ctrl+V | 自己编写C#代码 | 对上述List中的所有成员递归调用接口  ICloneable.Clone。长按Ctrl+V可以连续粘贴。 |
| 统计功能 | 图形个数统计 |  | 自己编写C#代码 | 递归调用类属性Count并将它和Label绑定。 |
| 用户体验优化 | 打开和新建图片、关闭程序时的友情提醒。 |  | 自己编写C#代码 | 防止误删未完成画布中的内容。 |
| 显示系统时间 |  | 自己编写C#代码 | 使用DateTime.Now.ToString方法 |
| 错误处理 | 载入图像时对不支持图像及无法处理的图像的抛出。 |  | 自己编写C#代码 | 若捕获异常，会弹出对话框。 |

# 

# 2系统总体结构

## 2.1概要设计

按照“面向接口编程，而不是面向实现编程”的面向对象基本原则，在建立解决方案的时候我的队友董渊就将解决方案分成了Ccao-big-homework（UI，WPF工程项目，由我实现，调用者）和Ccao-big-homework-core（实现，C#类库，由我的队友董渊实现，被调用者）。双方互不干涉，独立调试。

由于采用了多种科学合理的设计模式（详见我的队友董渊的报告），本程序不曾进行过任何强制类型转换，不曾进行过任何运行时类型判定，所有的多态都靠重载函数来实现，充分体现了面向对象的思想。

## 2.2小组成员分工

小组成员分工说明

|  |  |  |  |
| --- | --- | --- | --- |
| 小组成员姓名 | 小组成员班级 | 小组成员学号 | 小组成员分工 |
| 邵键准 | 自44 | 2014011517 | （GUI设计）  程序界面绘制；  启动窗口动画制作；  菜单栏特效实现；  程序图标添加；  选中图形闪烁的实现；  显示系统时间标签的实现  （人机交互功能实现）  菜单栏按钮功能的编写:包括新建、保存、打开、全选、复制、剪切、粘贴、删除、样式选择、退出等功能；  图形选项卡的实现；  图形计数标签的实现；  鼠标复杂事件的实现，包括鼠标的单击、拖动等。  键盘功能的实现；  各种图形绘制的实现；  调色板的实现； |
| 董渊 | 自43 | 2014011493 | （逻辑层编写）  类库各类层次和接口的设计；  基本图形（矩形、圆角矩形、圆、椭圆、直线、贝塞尔曲线等）绘制的实现；  图形在界面上的显示；  画布内图形的遍历枚举；  图形及其组合的复制、删除与回收；  图形的单击选择、拖框选择，全部选择与组合选择；  图形组合的拖动；  图形的组合与解除组合；  图形可见性的调整；  图形状态的设置；  图形个数的统计 |

# 3本人工作内容

## 3.1 GUI的设计

GUI的设计采用了较为简洁的风格，设计完成后曾请求周边同学进行体验并对细节进行改进，力求用户体验较好。主体采用XAML语言，实现设计和功能的分离，并配之以C#的事件处理函数。程序取消了不怎么美观的窗口边框，并采取点住程序窗口任何一个位置均可拖动的方法。

### 3.1.1 程序启动和关闭动画设计

程序的主要窗口在开始和结束时都是通过淡入和淡出来呈现和销毁窗口，实现此效果使用了一个计时器，并让窗口的透明度随计时器而改变。下图为启动过程截图，可看见窗体还是半透明状态(请无视背景的代码)。



### 3.1.2 程序启动界面的设计

程序启动界面如下。



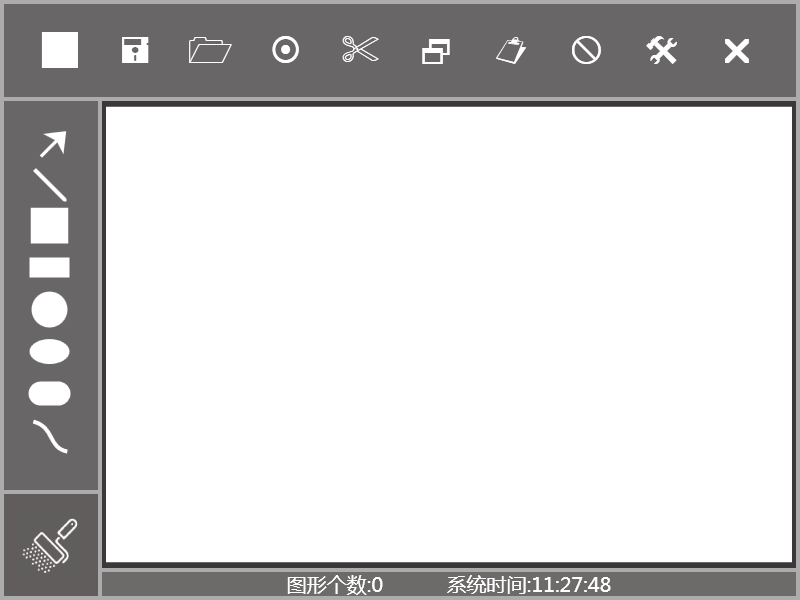
程序的启动界面曾经如下，感觉略丑，然后删了。



界面包括版本号与开发人员，“新建绘图”按钮在鼠标移上后会有高亮，点击进入绘图页面，点击“离开”按钮则直接调用Application.Current.Shutdown()函数销毁窗口。

### 3.1.3 程序主体绘图界面设计

主体绘图界面如下。



上部为菜单栏，具体按钮功能见使用手册，实现了鼠标移到某个按键上时该按钮闪烁一次并且放大，同时调整整个工作框的布局。效果如下。



左部为绘图框，选择后可使用鼠标绘制不同的图形，包括直线、圆、椭圆、正方形、长方形、贝塞尔曲线。其中贝塞尔曲线限于WPF提供的贝塞尔曲线构造函数的局限性，其必定从画布的左上角开始绘制。

左下角为一个图标，无实际作用。

### 3.1.4 程序图标的添加

我们的程序可是有图标的哦！



下面是调色板的图标。



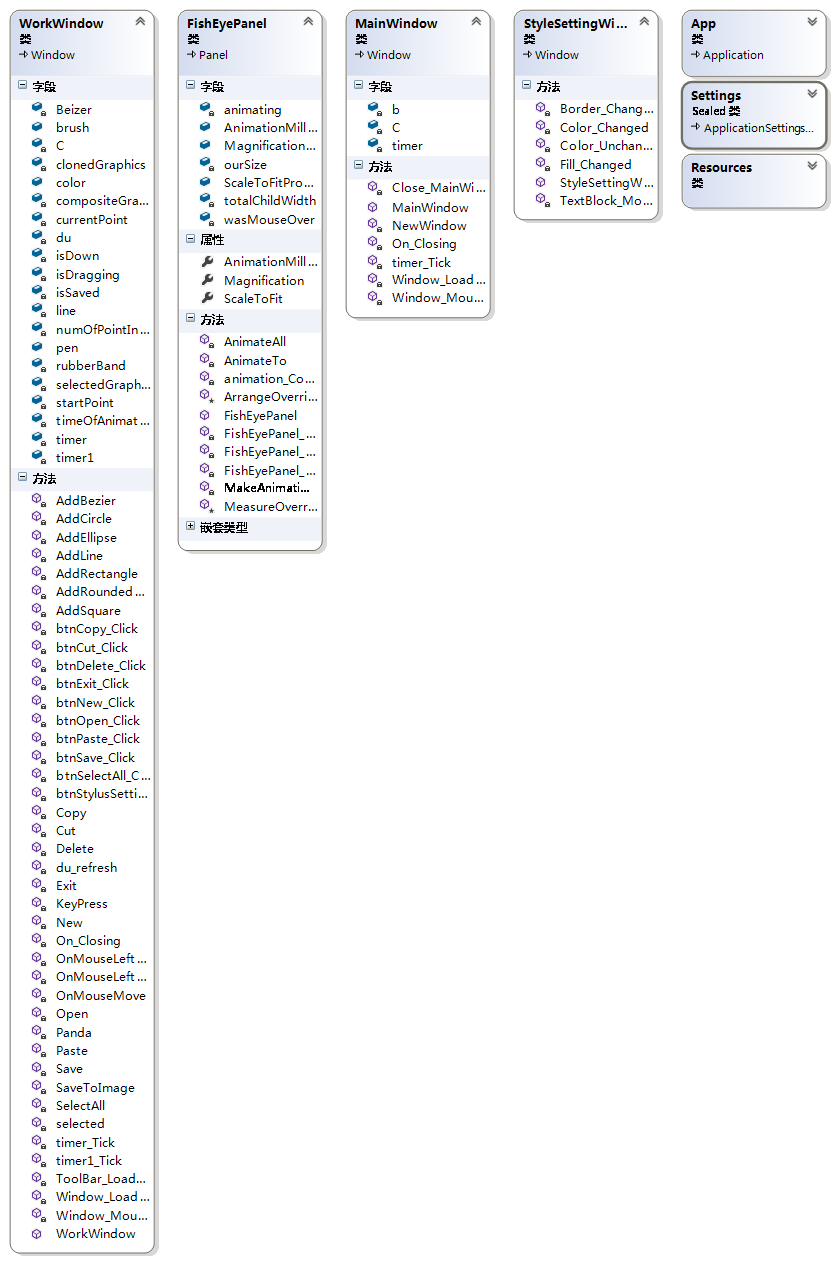
## 3.2 逻辑层

主要实现解决方案Ccao-big-homework。

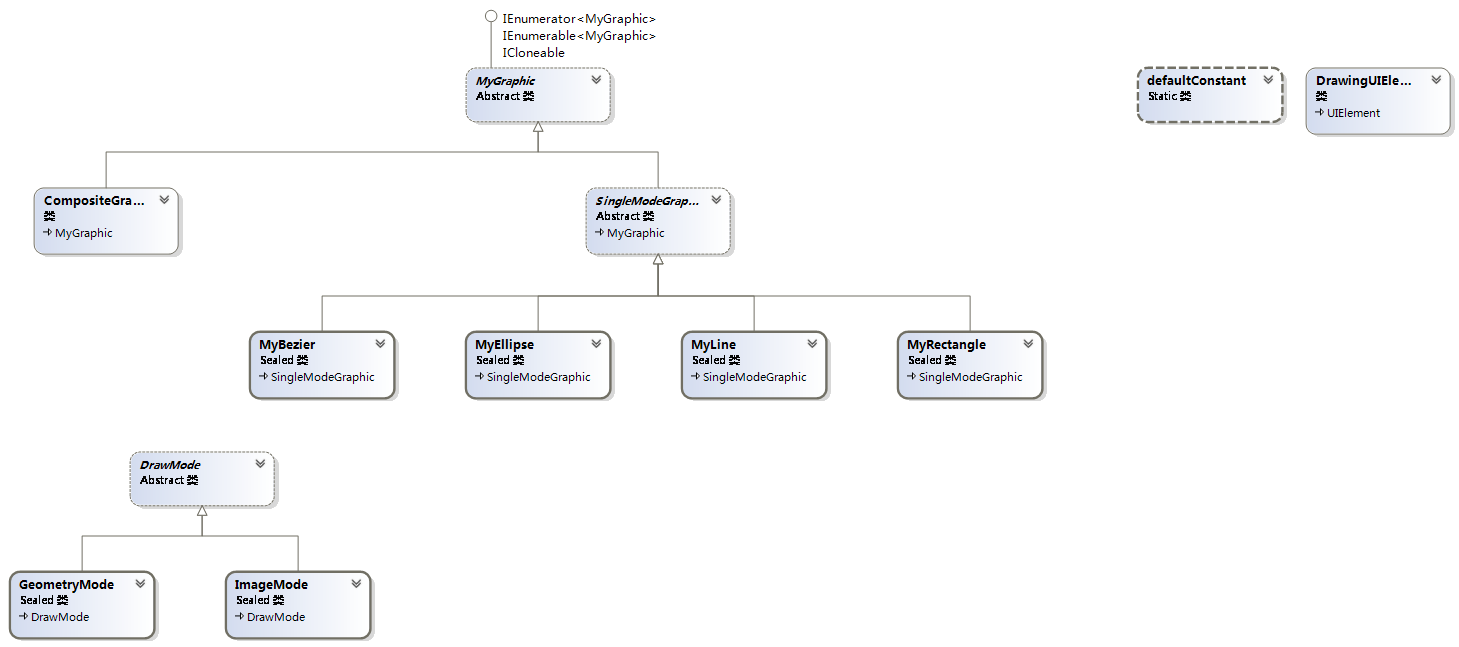
### 3.2.1 工作窗口WorkWindow部分

#### 3.2.1.1 概述

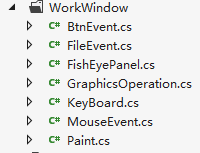
主要实现人与程序的交互，包括鼠标事件和键盘事件等。由于绘图软件基本是在一个窗口进行操作，一般使用组合而非继承的方式，因此类的结构比较扁平。类图如下。



（好吧呀就是没有继承，这个软件的继承关系主要在于绘图的各种图形与画布，主要由我的队友董渊实现，类图如下。）



由于没有继承，全部实现在一个窗口里代码显得非常臃肿，因此我根据功能的不同将同一个窗口类分成了如下几个文件。



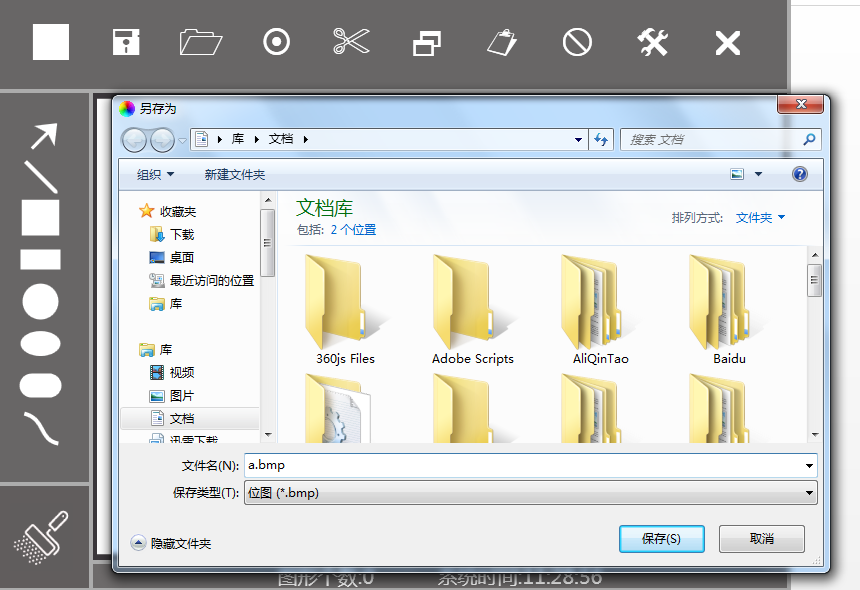
下面逐步介绍各个文件实现的方法。

#### 3.2.1.2 BtnEvent

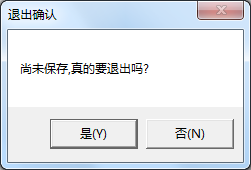
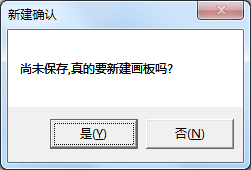
实现菜单栏所有按钮功能，包括新建、打开、保存、退出、全选、复制、剪切、粘贴、样式选择按钮。实现方法是调用其他文件里的私有函数。

#### 3.2.1.3 FileEvent

实现图片读写的相关方法。我们的程序支持使用文件夹视图来把文件保存到计算机的任意位置或是从计算机任意位置载入图片。同时还在现有图片未保存时弹出对话框提示用户要保存，防止了图片的误删，优化用户体验。



实现画布的新建与刷新，并实现程序的退出功能。



#### 3.2.1.4 FishEyePanel

实现上部菜单栏中当鼠标移到某个按键上时该按钮闪烁一次并且放大，同时调整整个工作框的布局。此处采用了组合的形式，FishEyePanel是一个新的类，在WorkWindow类中创建该类的对象，并布局到主窗口上。

#### 3.2.1.5 GraphicsOperation

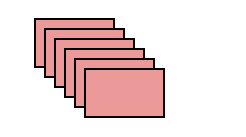
实现图像整体操作，包括：

（1）全选：调用队友提供的SelectRect接口，并把范围设置成整个画布大小的矩形，从而得到一个类型为List<CompositeGraphic>的对象，添加到selectedGraphics里。

（2）复制：调用队友提供的Clone()函数，往clonedGraphics这个对象里添加对象。

（3）剪切：复制的同时清空selectedGraphics。

（4）粘贴：将clonedGraphics里的所有成员添加到总画布compositeGraphic的Children()成员中，从而实现绘制，并向右下角移动(10,10)，从而和复制的原图区分开，然后清空clonedGraphics()，并再次调用复制函数()，从而实现粘贴的连续性，即复制一次可连续粘贴。下图为按下Ctrl+C后连续按下Ctrl+V的效果。



#### 3.2.1.6 KeyBoard

实现键盘的按键监控，包括：

（1）Ctrl+A：全选图像。

（2）Ctrl+C：复制选中图像。

（3）Ctrl+V：粘贴选中图像。

（4）Ctrl+X：剪切选中图像。

（5）Ctrl+W：关闭程序。

（6）Ctrl+O：打开图片。

（7）Ctrl+N：新建画布。

（8）Ctrl+S：保存图片。

（9）Delete：删除选中图像。

（10）Key Up、Key Down、Key Left、Key Right：选中图像的上下左右移动。

（11）Shift：按住时可多次增加选择已选中的图形

#### 3.2.1.7 MouseEvent

我的工作中最难的部分，而且肩负调试队友代码的使命（队友的代码属于core，比较抽象，自己难以调试）。

（1）Window\_MouseLeftButtonDown事件：实现鼠标不在画布上时窗口根据鼠标的移动而拖动。

（2）OnMouseLeftButtonDown事件：鼠标左键按下处理事件。主要记录鼠标开始移动的点startPoint，让画布捕捉到鼠标，并标记左侧的ToolBar是否选中rbSelect选择按钮。

（3）OnMouseMove事件：鼠标移动时的处理事件。当鼠标移动且画布捕捉到鼠标的时候，若是发现rbSelect选择按钮被选中且当前有图形被选中且Shift键未被按下，那么说明这个鼠标事件需要的是移动图形，因而用虚直线实时绘制鼠标指示的移动路径；若是其他绘图按钮被选中，则说明要绘制图形，则用虚长方形或是虚直线实时绘制图形。

（4）OnMouseLeftButtonUp事件：鼠标左键抬起的处理事件。首先判断鼠标抬起时和按下时位置是否相同。若是相同，则说明用户只是按了一下，那么不管左侧ToolBar是选中的什么，说明用户都是想选中一个图像，因此将左侧的工具条调整到rbSelect按钮，并且调用总画布compositrGraphic的SelectPoint方法，得到这个点选中的图像，将其加入selectGraphics，在这个List里面的对象，每隔0.5秒更改一次可见性，从外观看来，选中的图形会闪烁。若是鼠标移动了，且选项选中的是图形选项卡，则绘制对应的图形，并且刷新画布，并把之前实时跟踪鼠标的虚线图形删去。接下来判断选项卡是否为贝塞尔曲线选项，若是则把该点增加到贝塞尔曲线的List里，当List里的成员个数增加到4时，绘制一条贝塞尔曲线并清空该List。最后是最为复杂的rbSelect按钮，如果当时选中的图形为0，即selectGraphics为空，则说明用户想要选中他框选的图形，则调用SelectRect得到选中的所有图形，并把其加入selectGraphics中使其闪烁。若selectGraphics不为空，则说明用户想要移动选中的图形，则调用move方法移动选中的图形，并把selectGraphics清空。同时，如果整个过程中Shift键被按下且rbSelect被选中，说明用户想要增加选择图形，于是将选中的图形增加到selectGraphics里面。

以上方法还各自特判了贝塞尔曲线绘制时的点四个点的情况。

#### 3.2.1.8 Paint

各种图形的绘制，包括：

（1）直线

（2）长方形

（3）正方形

（4）圆

（5）椭圆

（6）圆角矩形

（7）贝塞尔曲线

其中（1）~（6）的绘制方式基本相同，都是新建一个该图形的对象，然后根据传来的两个点确定图形的长宽和位置，将这个对象添加至总画布compositeGraphic，然后刷新画布。

对于（7），绘制方式是在画布上点四个点，则出现贝塞尔曲线。

#### 3.2.1.9 其他零碎功能

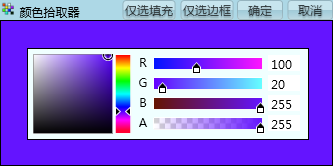
主要包括一些动画效果。启动时工具条的移入运用了ThicknessAnimation控件。为了美观我特意写了一个函数隐藏了工具条尾部的小箭头。选中图形闪烁的功能则使用了一个DispatcherTimer计时器，每过500ms就把selectGraphics的成员的可见性改变一次。图形个数统计调用总画布compositeGraphic的count属性，每隔0.5秒刷新一次。系统时间标签则使用DateTime.Now.ToString方法获取。

### 3.2.2 启动窗口MainWindow部分

这个窗口很简单，只实现了弹出WorkWindow和关闭窗口退出的功能。并贴了一张图，写了版本号。

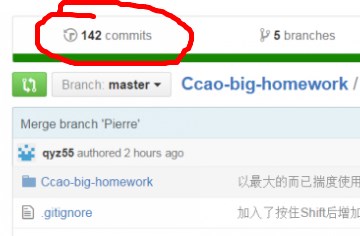
### 3.2.3 颜色拾取窗口StyleSettingWindow部分

颜色拾取窗口我并没有花太多时间自己写，本来以为WPF自带调色板控件，结果发现没有，于是在网上找了一个扩展控件，并组合到WorkWindow类中，在按下上部菜单栏中的“样式选择”按钮时弹出。

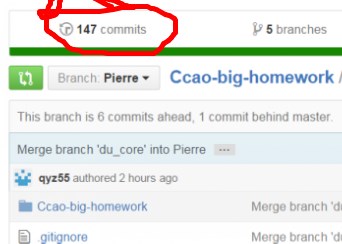


## 3.3 代码版本控制

本程序采用GitHub实行版本控制，代码仓库由我的队友董渊建立，Git上的地址为：<https://github.com/hzsydy/Ccao-big-homework>。截至9月17日，master主分支已有142次提交记录。

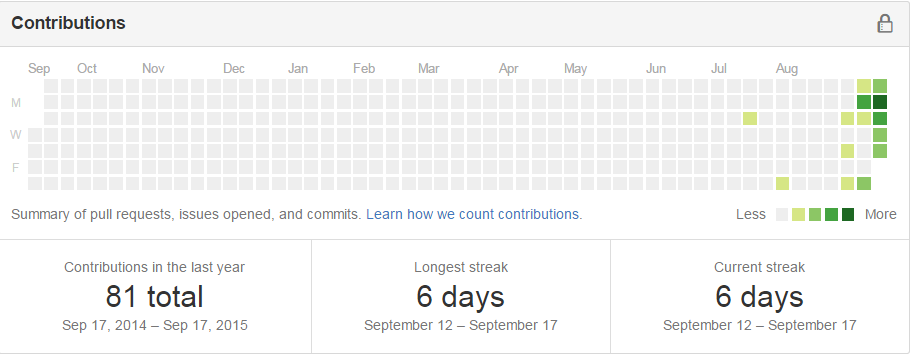


我的工作主要在Pierre分支里完成，队友董渊的工作主要在du\_core里完成。然后大家分别把分支merge到主分支里。截至9月17日下午16:24，我的分支Pierre已有147次提交记录。

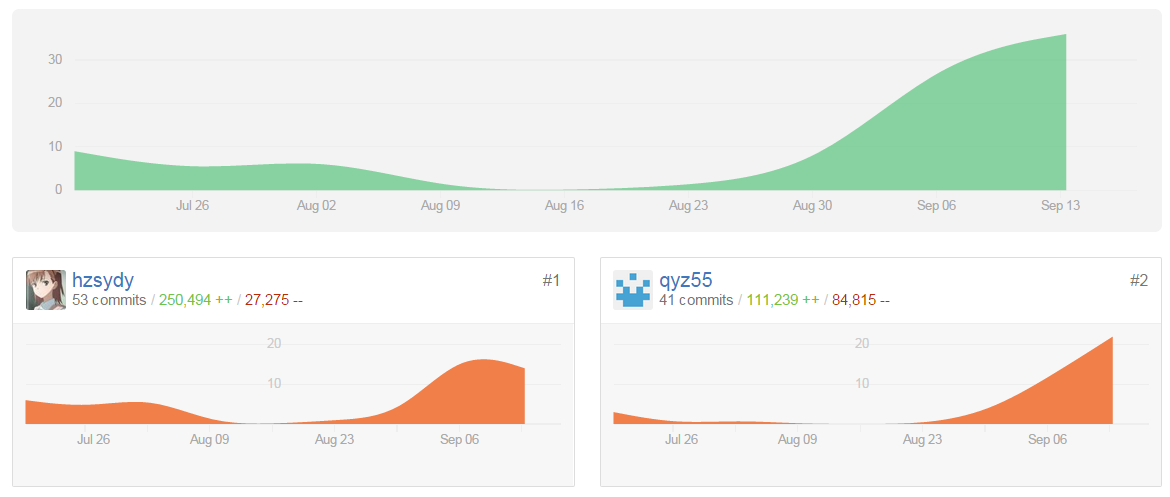


值得一提的是，前期由于我对GitHub的不熟悉，导致了3次代码的conflict，有两次都污染了master主分支，最后使用du\_core来恢复主分支的。同时，我在开发前期的分支Pierre’s-UI也被删除。经过这些教训，我对代码的版本管理的重要性有了更深刻的认识，对Git的使用越发熟练，也让我在程序主体开发过程中和队友合作愉快，没有出现无法解决的大问题。

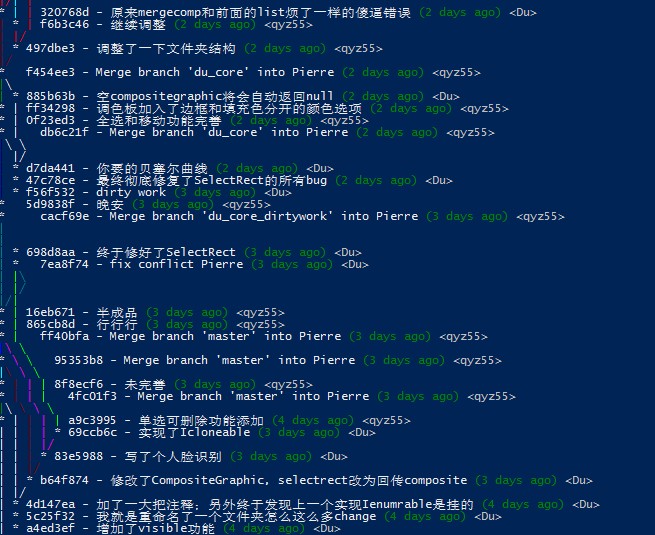
以下是我在Git上的数据统计（注意到这一整条代表一年，所以绿色主要集中在后部）。



以下还是数据统计。其中用户名qyz55是我。代码量如此之大主要是用了第三方库的原因。



下面是我们部分提交记录的截图。



## 3.4 使用手册的编写

由于我负责设计界面和具体功能的实现，对软件的各项功能比较熟悉，所以由我负责撰写使用手册。

# 4项目总结

开发一个程序让我学到了非常多的东西，开发一个程序也让我遇到了前所未有的困难。我们是以产品级的程序设计理念要求自己，而不仅仅是一个大作业，为此我们花在测试和调试上的时间几乎等同于代码开发的时间。暑假里拿着一本《C#入门经典》自学C#，从当时什么也不会到现在已经能够熟练调试C#的代码，殊为不易。下面是我的几点总结与想法。

## 4.1 OOP模式的优越性

接口和实现的分离极大提高了我们的开发效率。只需要知道给接口提供什么参数，能得到什么结果就可以了，不需要知道它是如何被实现的，这样的黑箱极大地提高了我们的开发效率。

## 4.2 版本控制的重要性

版本控制不仅让我们能够清晰地了解自己的开发进度，而且在开发出现重大bug的时候，能够迅速地回滚到以前的版本。这样不会出现自己写了什么挂了都不知道的，结果只好全部删除重来的情况。况且，每一次在Git上的Commit都意味着我实现了新的功能，内心总能感到满足。

## 4.3 自主解决困难的能力

为了写大作业我估计百度了不下300次，基本开发的状态就是：想实现某功能-->发现自己不会-->百度google大法好-->会了-->实现该功能-->又想实现某功能。

开发过程中我遇到过很多困难，举个例子：关闭窗口时我写了个淡出动画，结果发现动画刚开始运行就程序就被退出，表现为动画无法播放。百度了一个小时，终于找到解决办法：在给程序的OnClosing事件中，把close命令取消，然后坐等0.6秒动画放完后再退出程序。

真正到开发程序才发现以前学的都是纸上谈兵，经过一份大作业的洗礼我收获了极大提升的编程技能。

## 4.4 团队合作的技能

两个人的小团队也需要大量的合作，以前我一直以为所谓的负责整合代码的人都是打酱油的，但只有自己写了才发现代码耦合的困难，尽管我们已经分工如此明确。

困难一：为了完全达到设计与后台分离的目的，我们使用了WPF，然而队友董渊一开始是用System.Drawing，实现了所有的绘图抽象类，直到耦合了才发现这是基于WinForm的，WPF使用的是System.Windows.Media类来实现绘图功能，最后只好全部推倒重写，这体现了前期调研的重要性，没有清楚地明白自己要干什么就直接开工最后很容易一事无成（下图的所有commit直接无效）。



困难二：队友实现的抽象类，本身并不能调试，所以要在我的界面做完以后通过效果来检验他写的是否正确。鉴于盲打代码的不靠谱性，我的实现真是bug不断，比如在实现过程中发现提供的接口不能完全实现想要的功能，只好再找队友要接口，比如当初考虑时没想到选中的图像需要闪烁的问题，于是临时加上了isVisible属性。比如框选图形的函数一开始是写挂了的，而且还不止一个bug，最后是队友逐行调试逐步修正调出来的，而在此期间我就是不停地运行-->画图-->框选-->选不中-->看自己的逻辑有没有挂-->报给队友说他负责的那部分挂了-->逐行调bug。

诸多困难，就不一一赘述了。

# 5相关问题的说明

## 5.1 开发、测试环境

开发环境： Visual Studio 2012 Premium

运行框架：.net framework 4.5

测试环境：Windows 8.1、Windows 7

## 5.2 开发语言

C# & XAML

## 5.3 使用方式

参见《使用手册》