C#实现简易绘图工具

1. 引言

实验目的:通过制作窗体应用程序(C#画图软件),熟悉基本的窗体设计过程以及控件设计,事件处理等,熟悉使用C#的winform窗体进行绘图的基本步骤,对于面向对象编程有更加深刻的体会.

Tutorial任务

设计一个具有基本功能的画图软件

·包括简单的新建文件,保存,重新绘图等功能

·实现一些基本图形的绘制,包括铅笔和基本形状等,学习橡皮工具的创建

·设计一个合理舒适的UI界面

注明:你可能需要先了解一些关于winform窗体应用程序绘图的基本知识,以及关于GDI+类和结构的知识

关于C#图形设计:

1. 实验环境

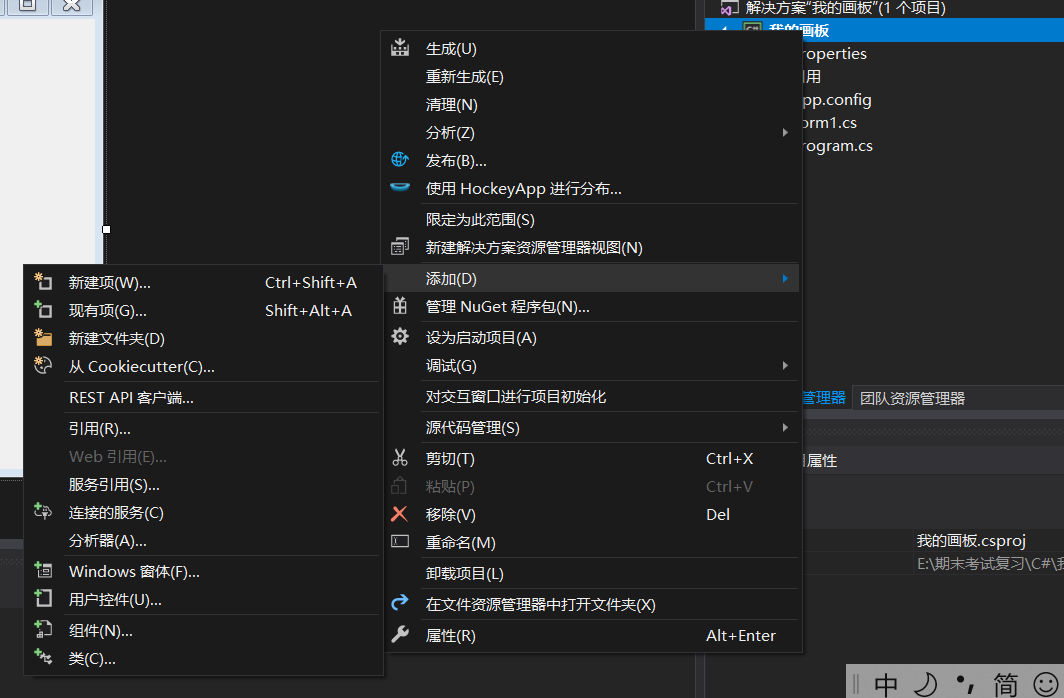
Windows系统下的visual studio 2017,C#窗体应用程序

1. 实验过程

首先,新建我们的项目,取名为”我的画板”

对于画板来说,其中很重要的就是工具,因此,我们需要创建一个类用来存放需要的画图工具,比较常见的功能有绘制直线,矩形,椭圆形,圆形,正方形,以及绘制填充矩形,椭圆形,圆形等,同样,我们还需要橡皮对我们画错的地方进行修改

因此,在项目中创建一个名为drawtools的类,如下图所示:



接下来,我们需要在创建好的类中创建我们的工具:

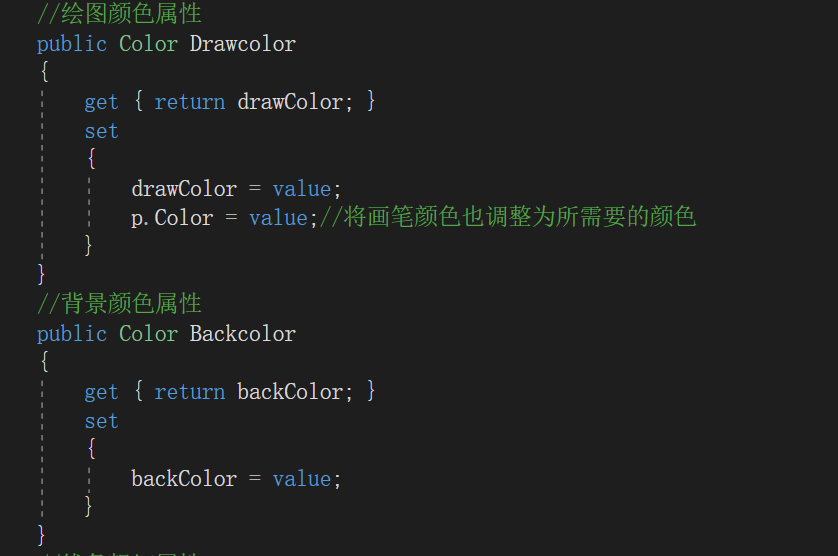
1.声明一些变量:

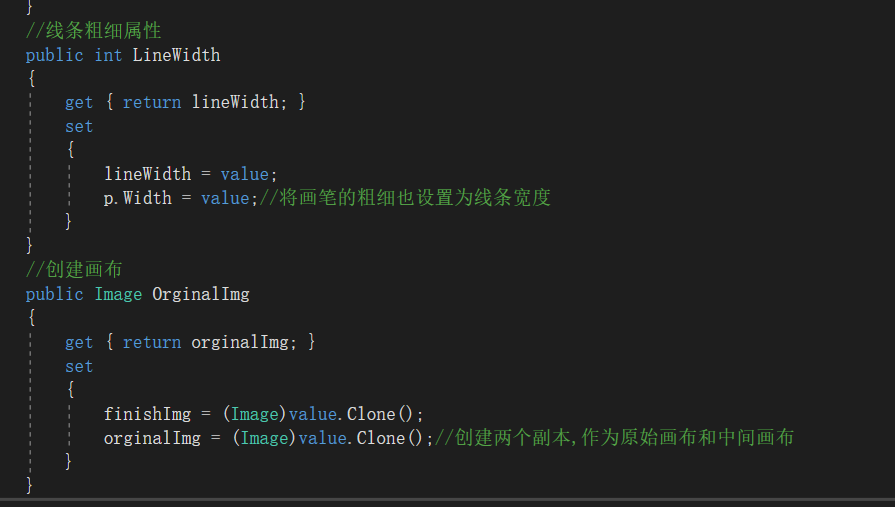


这里需要说明一下,在绘图过程当中有原始画布和中间画布两种画布,是因为在实时绘图过程当中,…………………(还没有搞懂)

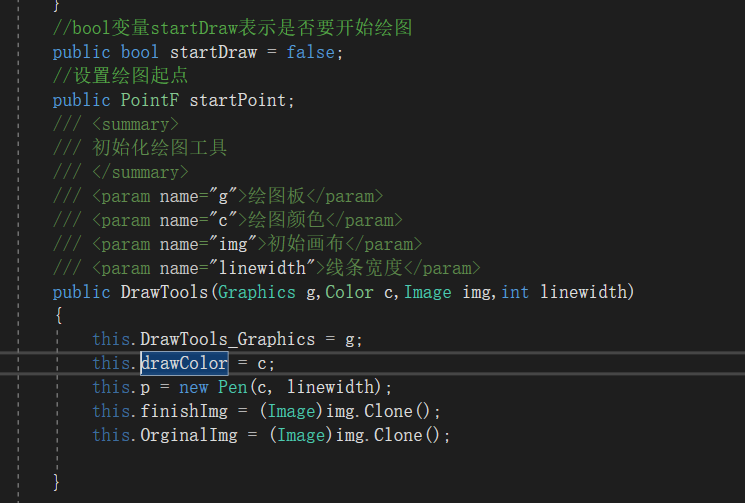
为了更加有效地防止造成图片抖动,从而防止记录不必要的绘图过程中的痕迹,我们的画要先在中间画板上完成,然后再将绘制好的图画一次性导到目标画板上

接下来是属性,通过get和set方法设置绘图颜色,背景颜色,线条粗细以及获得原始画布





接下来,利用构造函数对绘图工具进行初始化



下一步,绘制直线,矩形,圆形,正方形,填充圆形,填充矩形,填充正方形,填充椭圆形

(为了更加有效地防止造成图片抖动,从而防止记录不必要的绘图过程中的痕迹,我们的画要先在中间画板上完成,然后再将绘制好的图画一次性导到目标画板上)

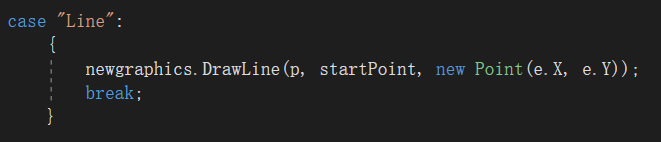
1. 实例化中间画板,此时的画布为上次绘制结束时的副本(当第一次绘制时,画布是初始化的画布副本)
2. 按照选择的绘图样式在中间画板上进行绘制
3. 绘制结束的图画应该画到中间画布上

这里面有一个值得注意的问题,那就是在绘图过程中,用户需要能够实时地看到自己画画的过程(体现在目标画板上),所以在鼠标移动过程中,为了看到效果,绘制的图画依然要显示在目标画板上,而最终绘制结束时的图片应当在鼠标松开时完成,此时才将中间图画最后导出到原始画布上

接下来,进行各种图案的绘制,思路如下:

利用switch和case语句对于各种情况进行分类讨论,即绘制直线,长方形,椭圆等

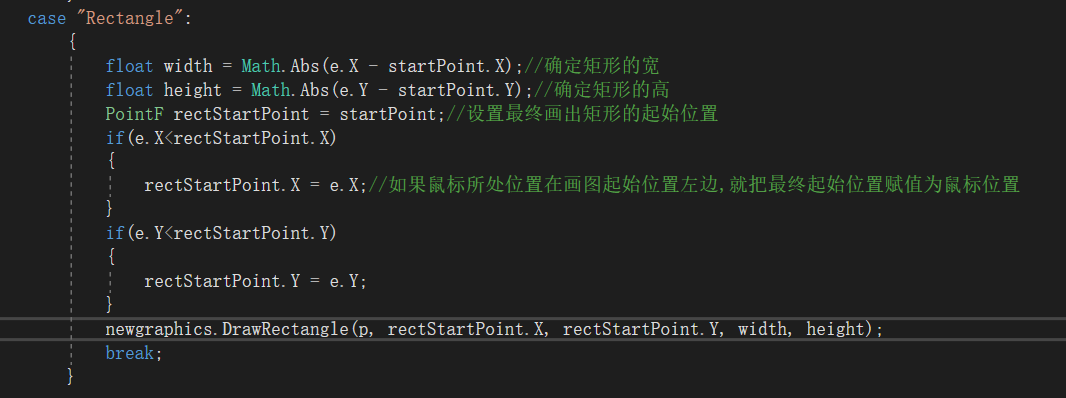
1. 绘制直线,也就是在绘画起始位置与鼠标移动到的位置之间绘制一条直线



1. 绘制一个矩形

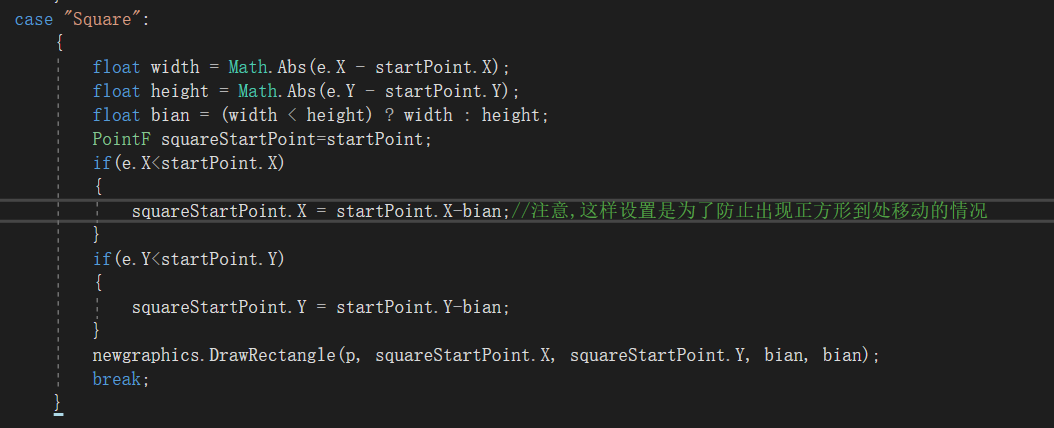
值得注意的是,鼠标移动到的位置与起始位置之间横向距离的绝对值就是矩形的宽,纵向距离的绝对值就是矩形的高,但是由于鼠标移动的位置可能会出现在起始位置的左侧或者上侧的情况,因此要分类进行讨论:

例如,在鼠标移动位置位于起始位置的左侧时,由于矩形的长已经确定,所以不妨将起始位置改为鼠标最后移动到的位置,以免出现当鼠标移动到起始位置左侧时,无法成功绘制矩形.也就是说,我们要让矩形绘制的位置位于起始位置的右下方,讨论方式见以下:



1. 绘制一个正方形

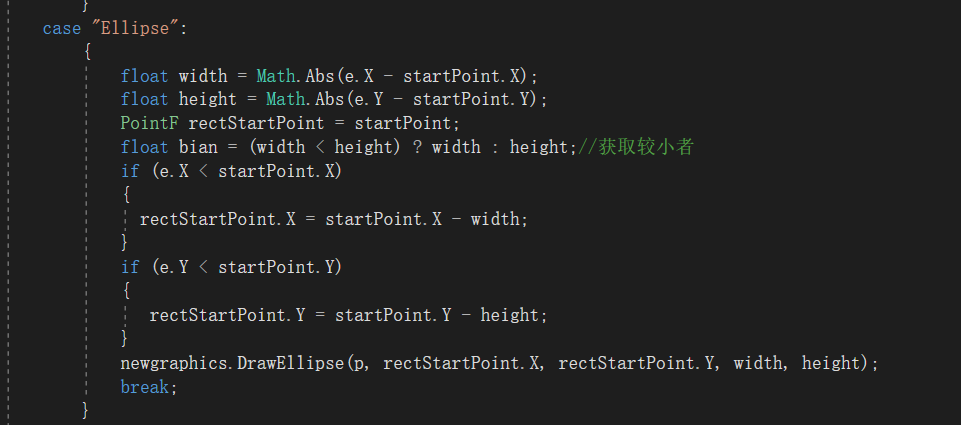
与绘制矩形基本类似,区别在于,正方形的长和宽是等长的,因此,要选择矩形的长和宽中较短的边(或者较长的边)作为正方形的边长,代码如下:



写到这里我们会发现,用户在绘制图形时,如果向左上角进行绘制,表面上看是矩形的左上角向左上方移动,实际上则是从鼠标最后拖拽到的位置开始进行向右下角的绘制(也就是绘制起点是在矩形左上角,进行右下角方向的绘制)

1. 绘制一个椭圆形

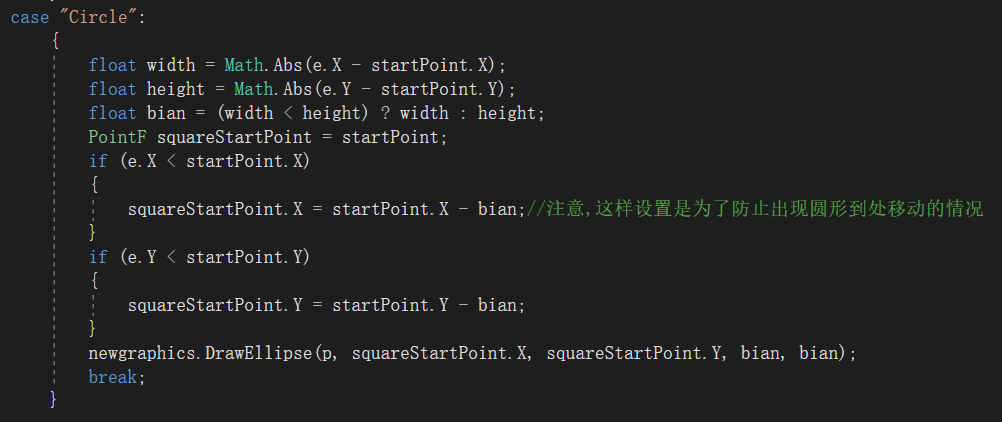
直接使用DrawEllipse的方法对于椭圆形进行绘制



这是一个用矩形边框定义的椭圆,可以理解为长是鼠标移动位置到起始位置x轴方向上的距离.同样,宽是鼠标移动位置到起始位置y轴方向上的距离.完全类比绘制矩形的情况,我们成功解决了绘制椭圆形的问题.

1. 绘制一个圆形

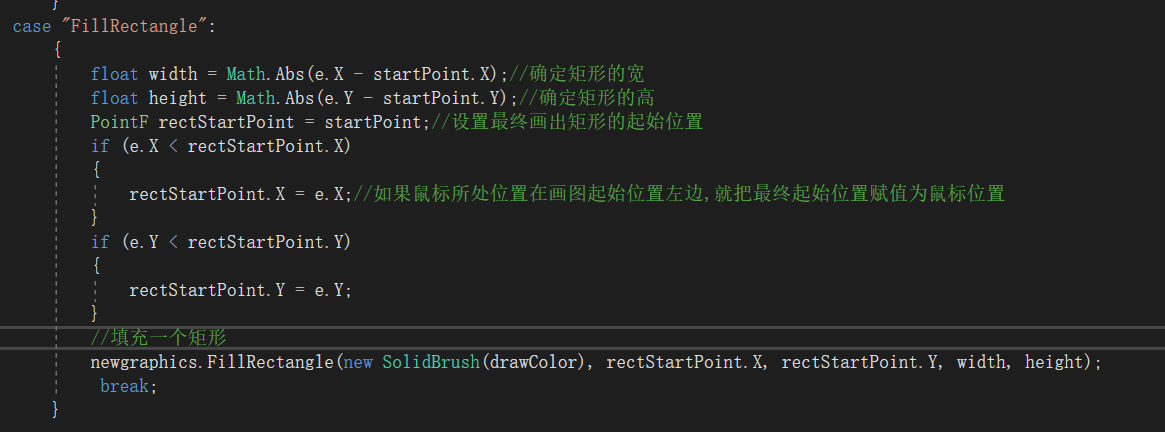
与绘制正方形类似,在这里同样要判断鼠标位置,取长和宽中较短的边作为圆的直径(也就是相当于以一个正方形作为边框将圆形绘制进去)



在画图板中,有的时候我们需要对图形进行颜色填充,这里我们将问题进行简化,不妨设置矩形填充,圆形填充,正方形填充,和椭圆形填充四个部分

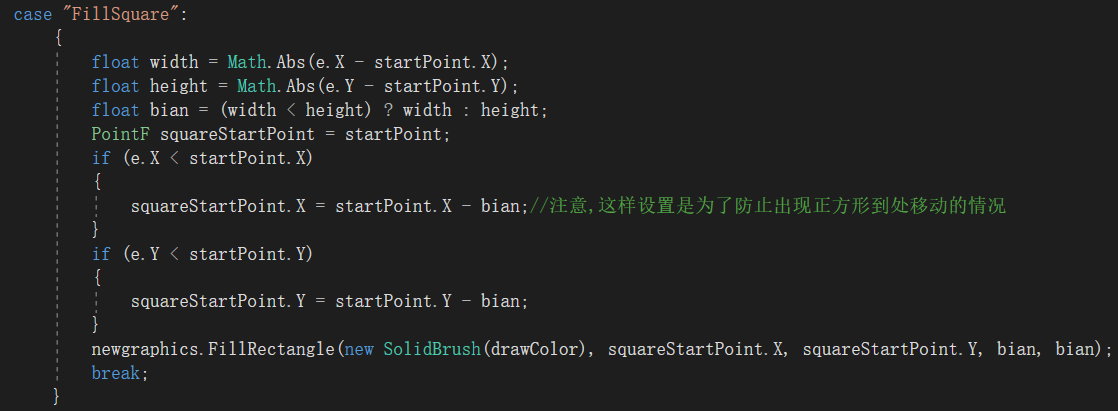
1. 填充一个矩形

和创建矩形相似,唯一的不同是将绘制矩形的方法改为了填充矩形的方法,代码如下:



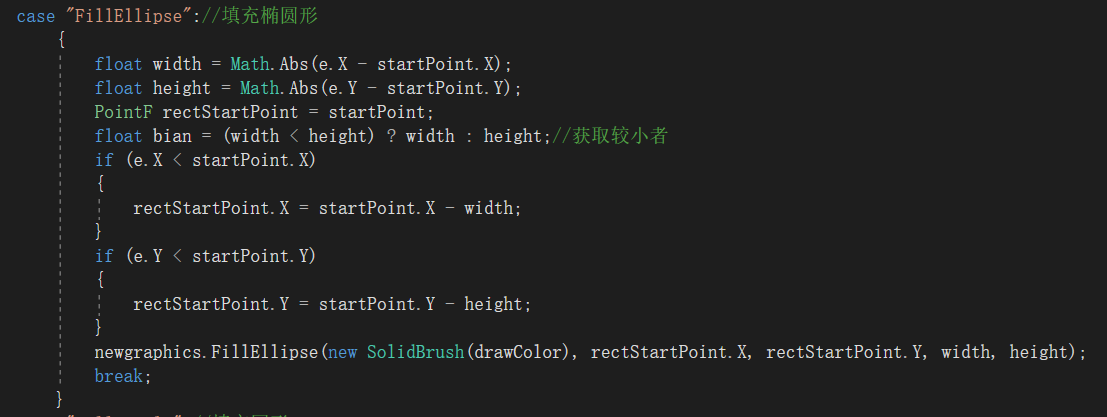
1. 填充一个正方形

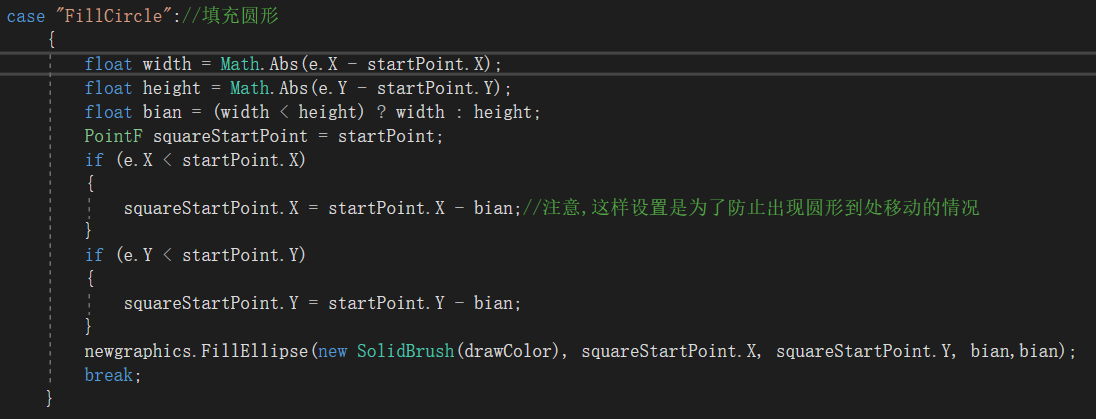
同样,类比创建正方形,代码如下:



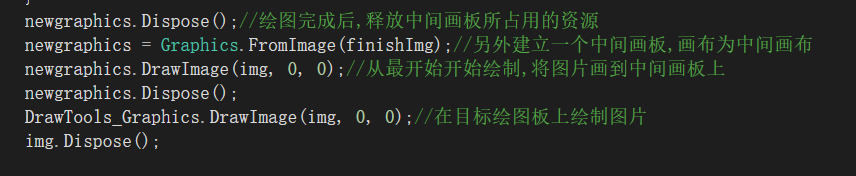
1. (9)填充一个椭圆形以及填充一个圆形

同样,类比创建椭圆形以及创建圆形,可以写出如下的代码来对圆形和椭圆形进行填充





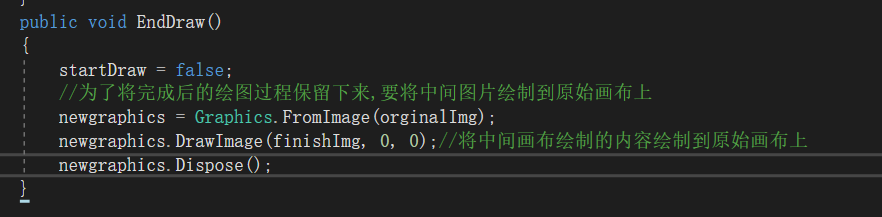
在对于switch分支进行判断之后,需要释放掉中间画板所占用的资源,同时需要重新建立一个以中间图片作为画布的中间画板,并且将图片绘制在中间画板上,并最终绘制到目标画板上



在这里,我们做了如下的几件事

1. 释放掉绘图所在的中间画板所占用的资源
2. 重新建立一个以中间画布为画布的中间画板(举个例子,相当于我们事先在桌子上的一张纸上画了一幅画,现在当画作完成后,我们把桌子移走,(将这幅画进行誊写)再将这幅画放置在了黑板上(也就是最终的目标绘图板))

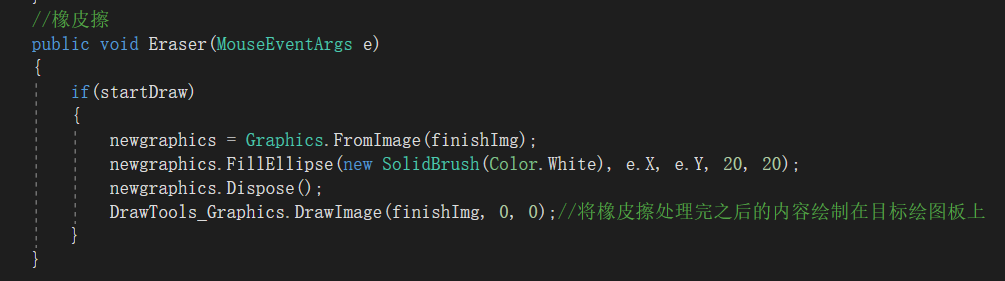
在绘图最终完成后,需要一个函数来把完成后的绘图过程保留下来,也就是将中间图片再绘制到原始画布上,代码如下:



至此,关于绘制各种基本形状的操作基本介绍完毕,接下来,需要在DrawTools类中加入橡皮擦功能以方便用户对于绘制的内容进行修改;

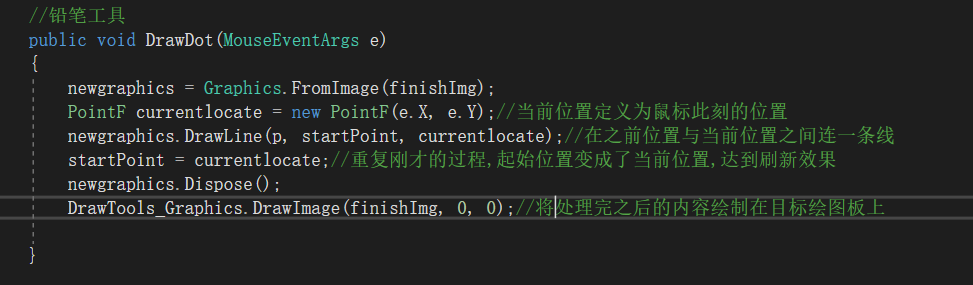
(1)橡皮擦工具的设置思路

橡皮擦的作用是将之前绘制的图案抹除掉,转换一下思路我们可以理解为使用一个与初始绘画区域颜色相同的形状去填充原来绘制的图案(比如如果初始绘画区域为白色,那么橡皮擦可以理解成用白色的矩形/椭圆形/圆形(通常是正方形)去填充之前绘制的图案),想通了这一点之后,我们就可以写出关于橡皮擦的代码:

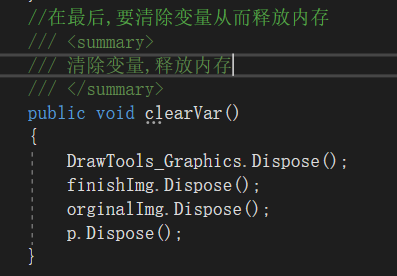


(2)铅笔工具的设置思路

由于用户并非只是绘制简单图形,因此需要设置铅笔工具以让用户绘制自定义形状,而设置铅笔工具的要点就在于如果绘制出看上去可以是”不规则”的形状,这里我们采用的方法是实时更新新的位置来模拟绘制任意曲线的过程,也就是将每次鼠标移动到的位置定义为当前位置,并且在之前位置与当前位置之间连一条直线,由于刷新速度够快,因此看上去就像是在绘制任意曲线,这一部分的代码如下:



在使用完画图工具后,要清除变量以释放内存



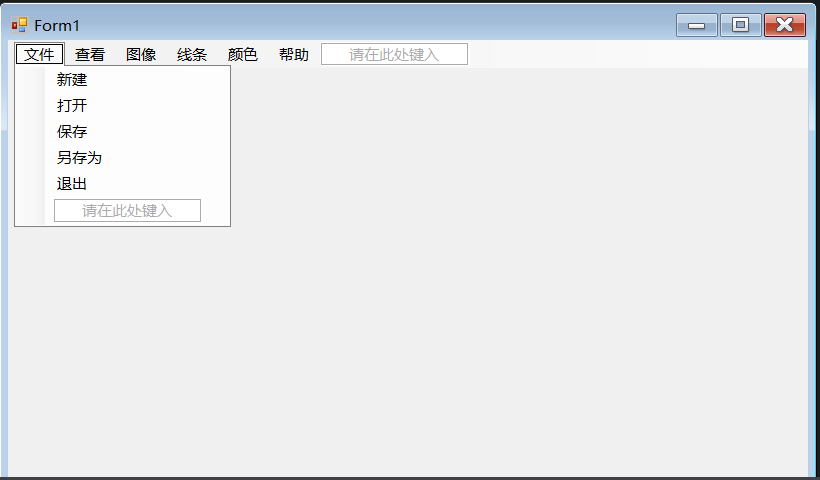
至此,关于DrawTools类的内容已经介绍完毕

二.

接下来,我们需要设计绘图工具的窗口,并且在相应的选项下面添加相应的功能

1. 首先是绘图窗体顶部的菜单栏,需要的较为常见的工具有:
2. 文件:包括新建,打开,保存,另存为和退出五个功能
3. 查看:在这里面可以查看工具箱,颜料库,和状态栏
4. 图像:这里面加入了清除图像的功能
5. 线条:可以选择粗细以及自定义
6. 颜色:这里面可以编辑颜色与背景颜色
7. 帮助:这里面是关于软件的一些基本信息

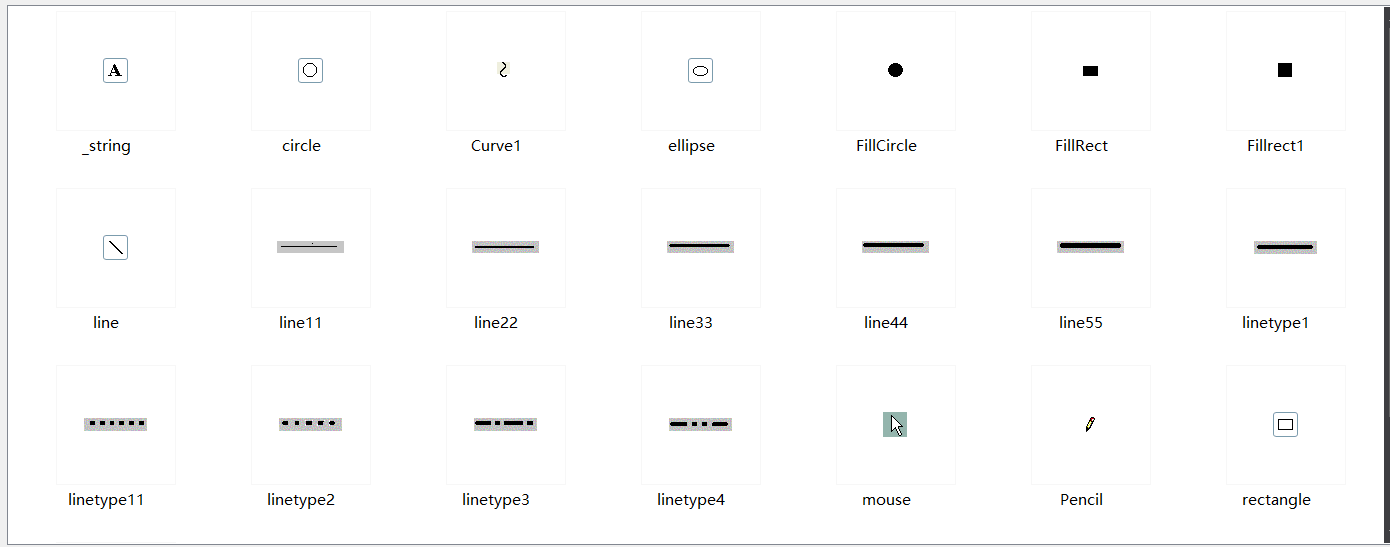
想好了以上工具之后,我们使用菜单—工具栏下面的menustrip来在窗体中设置以上功能,文件部分设置结果如下:



1. 顶部菜单栏设置完成后,我们还需要左侧的工具栏,下方的线条栏,以及右侧的画布部分
2. 新建一个Panel作为承载左侧工具栏的容器,取名为panel1,在默认参数的基础上对某些参数更改如下:
3. 将布局中的Dock更改为”Left”,使得该容器在左侧显示

(b)将行为中的TabIndex(确定此控件将占用的Tab键顺序索引)设置为9

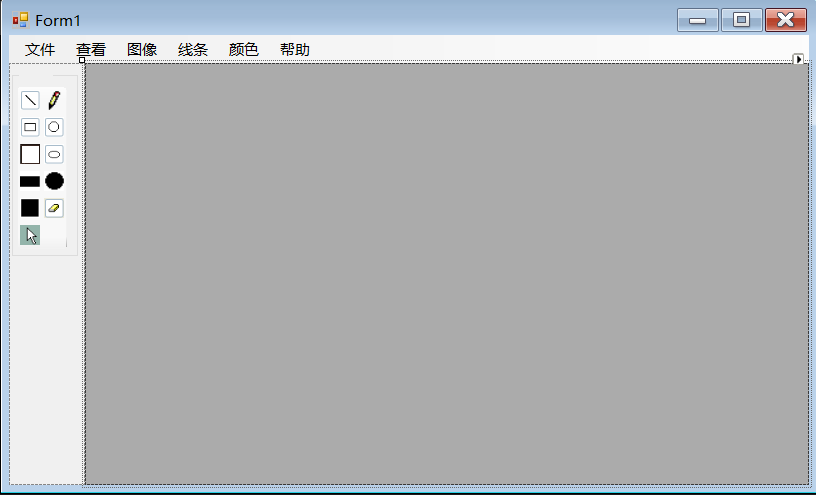
1. 在panel1中放置两个groupbox,其中groupbox1用来存放画板中的工具,groupbox2用来存放线条样式,并新建一个toolstrip设置若干工具(铅笔,橡皮等),具体操作见(3)
2. 首先,将toolstrip1调整为合适大小,使其放置在groupbox1中
3. 然后,我们将所需要的素材导入,选择解决方案---properties---resourses.resx,导入我们所需要的素材,素材大概如下:



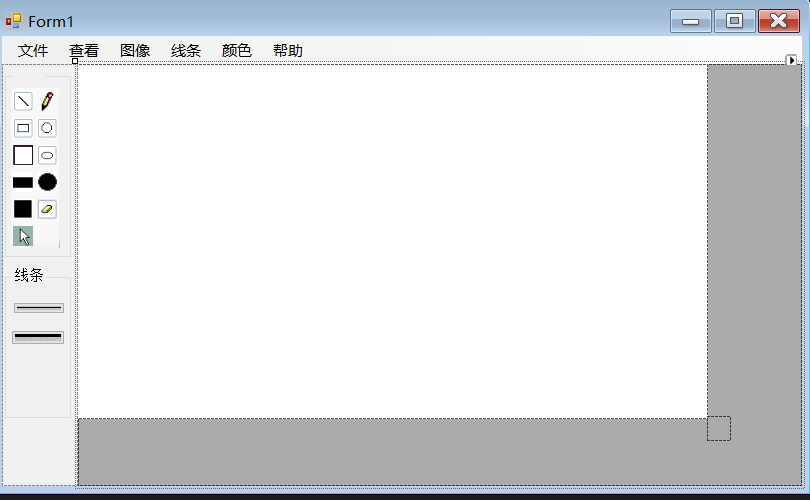
1. 开始逐一向左上角画板部分中添加功能(右键选择toolstrip1—编辑项,将我们所需要的工具的名称,图案等添加进去),分别为绘制直线,铅笔工具,矩形工具,圆形工具,正方形工具,椭圆形工具,矩形填充工具,椭圆形填充工具,橡皮工具,放置结果如下图所示:



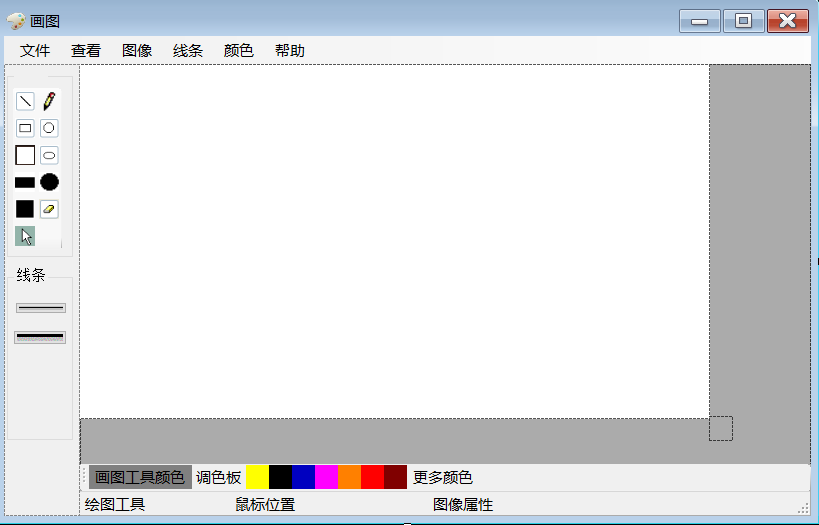
1. 新建一个panel2用来承装画布等,创建完的效果如图所示:



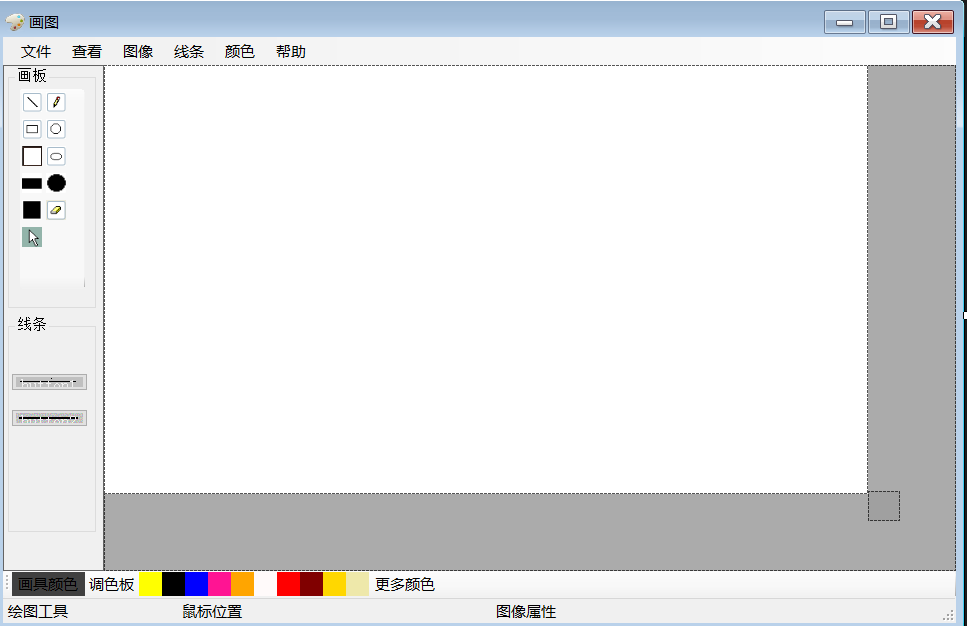
1. 新建一个groupbox2,用来存放各种线条工具,然后将不同粗细的线条以button的形式放置在groupbox2中
2. 新建一个picturebox用来当作画布,取名为pbImg
3. 另外新建一个picturebox取名为reSize,方便后续调整画布的大小,以及确定每次新建文件的位置,以上面板设计完之后效果如下:



1. 接下来,还有比较好用的功能是在窗体的左下角设置一个调色板以方便用户调节颜色,以及设置一个显示鼠标位置的功能方便用户确认绘画位置,最后,再设置整个窗体的图标和文字设置,以上所有窗体设计部分之后的结果如下:

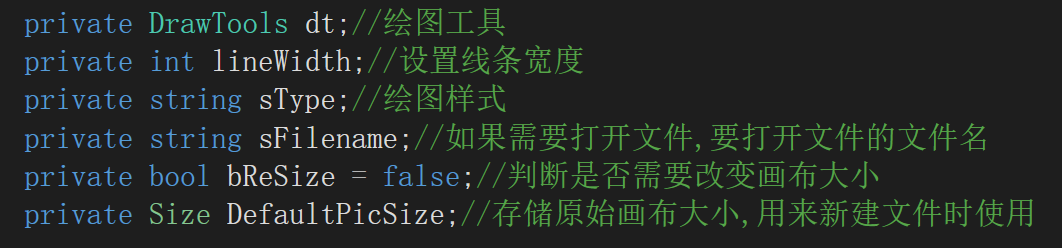


对窗体的布局再进行简单的调整,最终效果如下:



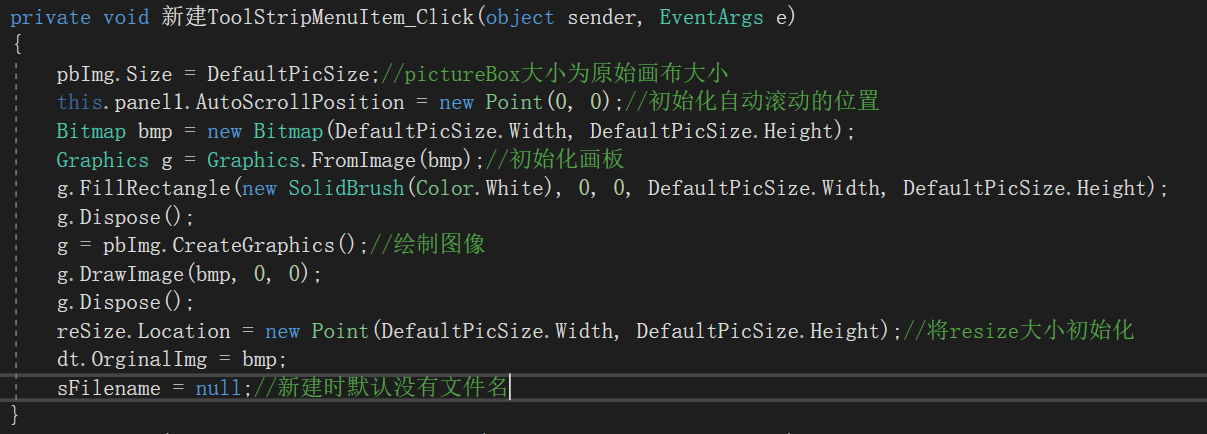
接下来,我们逐一完善各个功能

首先,需要设置一些变量以便于我们后续操作,设置变量如下:



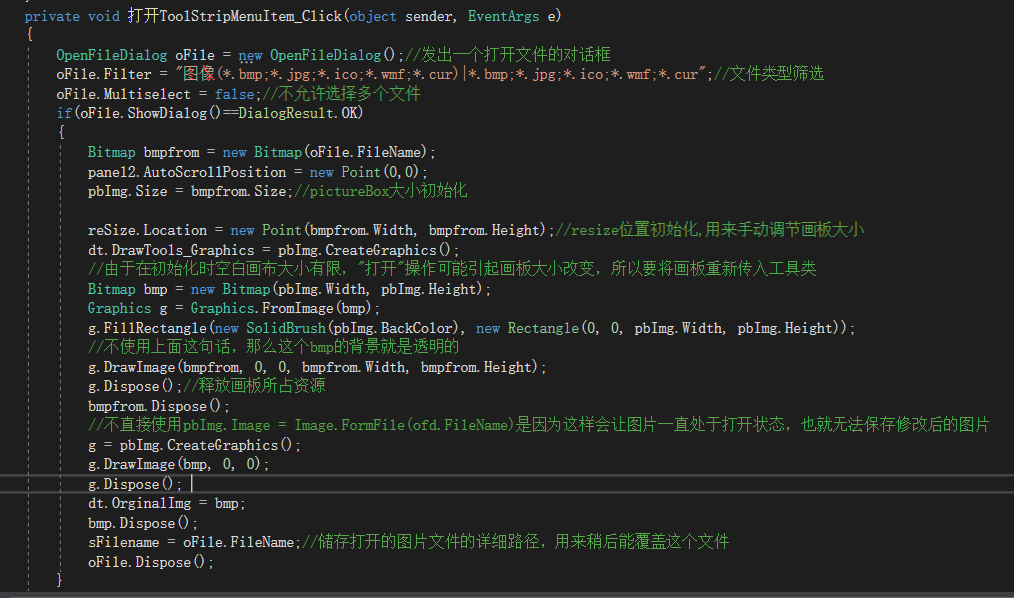
(1a)新建功能(存疑!!!)

这一部分的要点在于,需要对画板和画布进行初始化,定义其初始位置以及resize的位置,具体代码部分如下:



(1b)打开功能(存疑!!!)

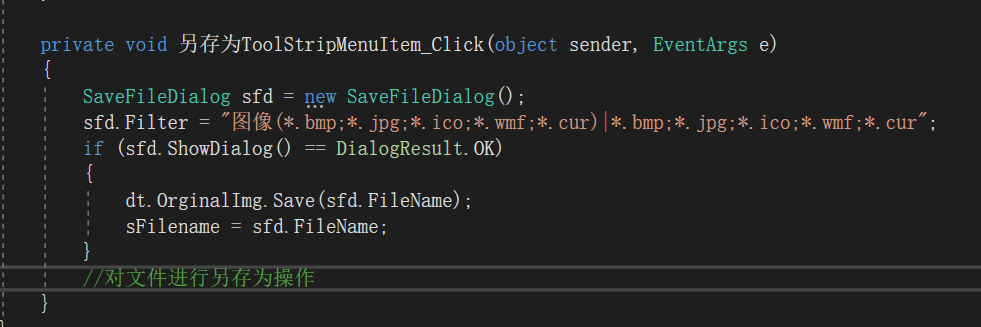
这一部分最重要的就是理解好绘图板初始化的问题,这一部分的代码如下:



(1c)保存

这一部分需要分两种情况进行讨论,一种是用户之前已经命名过这个文件(也就是这个文件有文件名),另一种情况是该文件第一次被命名,以上两种分情况讨论代码如下:

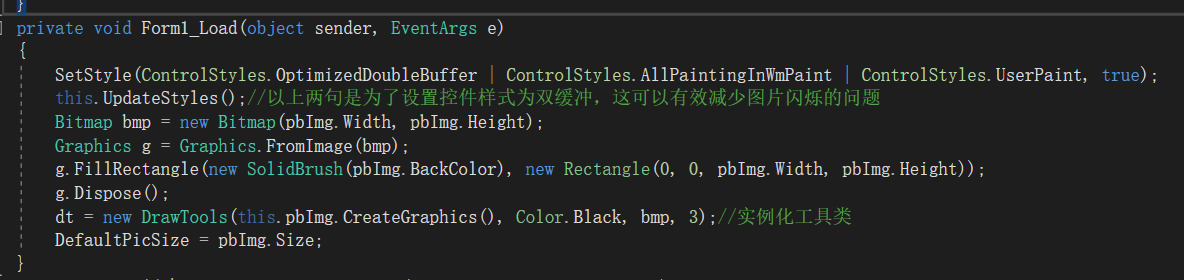
(1d)另存为文件,这里面需要指定路径,代码如下:



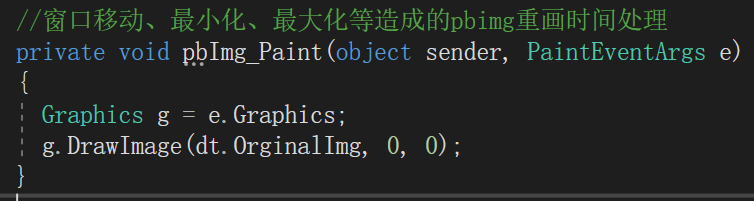
(1e)退出

这一部分需要询问用户是否确定退出,如果是的话就退出窗口

补充:在窗口加载时就需要对我们之前所写的类进行实例化以及对变量进行初始化,在这里我们写了这样一个名为form1\_load的函数

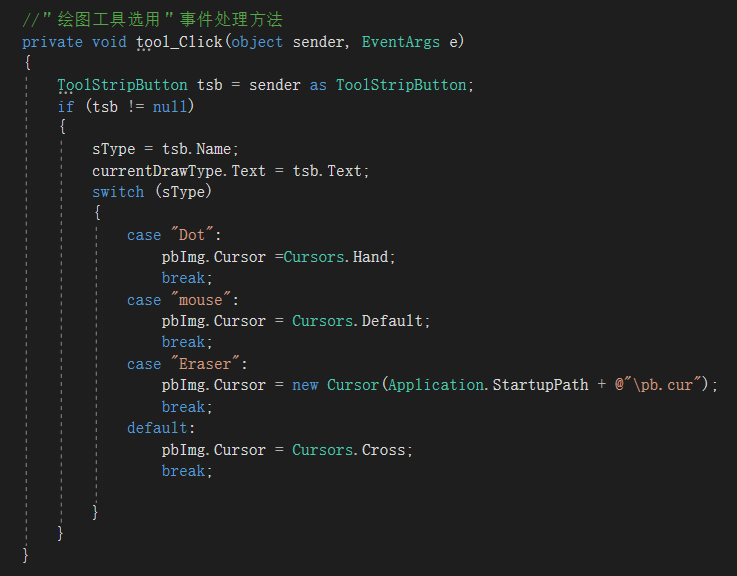


同时,我们还要处理窗口的移动,最小化,最大化所造成的pbImg重画的时间问题,处理代码如下:



1. 接下来,比较重要的一点是要成功地应对”绘图工具事件”,所以接下来是”绘图工具选用”事件处理办法

值得注意的是,对于不用的工具,我们在使用的时候的鼠标光标要有所不同,比如说在选择工具下鼠标光标会呈现指针形状,而在绘制矩形等形状时为了方便用户可以看清绘制大小,采用十字交叉形的光标.代码如下:

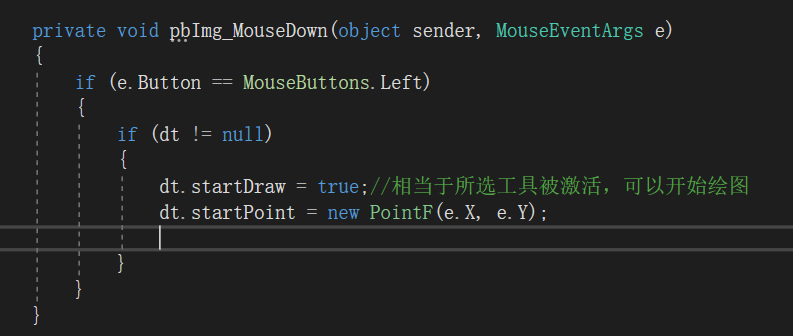


这一部分进行了一步类似于强制类型转换的操作,as在遇到不能强制转换的时候,会返回一个null, sender是事件源,这句的意思是把引发该事件的事件源转成button类型as是用来强制类型转换的,用as转换的好处是,转换失败不会出现异常，这样就比一般的转换更加安全,接下来,对于不同的绘图工具采用不同的光标

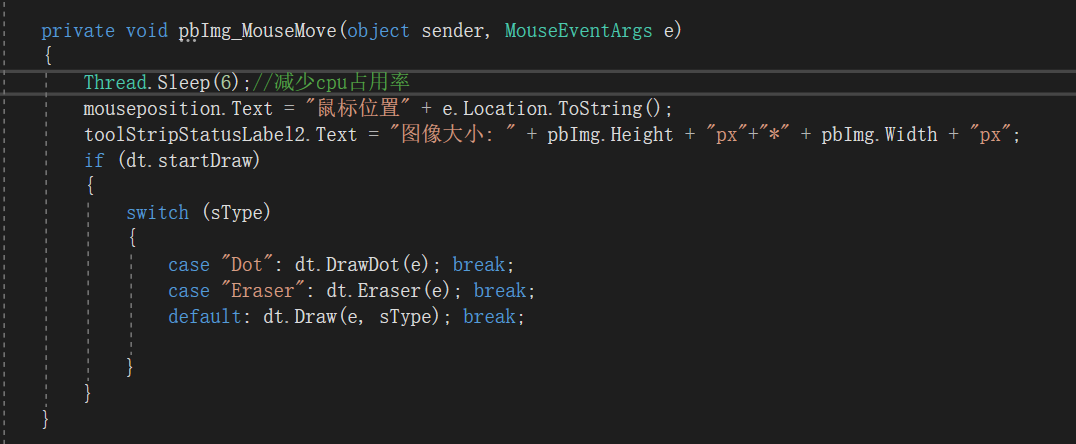
1. 绘图

在绘图时,用户会完成点击鼠标,鼠标移动以及鼠标松开这样三个动作,所以我们要创建三个事件已完成绘图过程

(4a)mousedown事件(当按下鼠标左键时绘图工具会被激活)

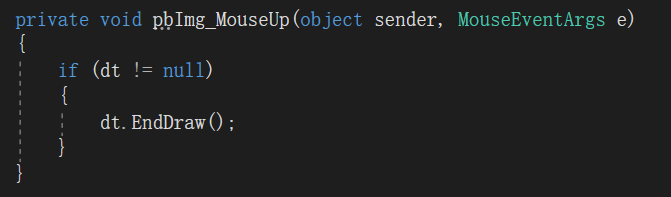


(4b)mousemove事件.在鼠标移动也就是正在绘制图像的过程当中,显示关于鼠标位置以及图像大小的信息



特别需要注意的是,由于铅笔工具和橡皮擦工具需要实时更新鼠标的位置以方便绘制多样化图形,所以我们使用mousemove事件,使得绘画过程随着鼠标移动可以实时进行更新

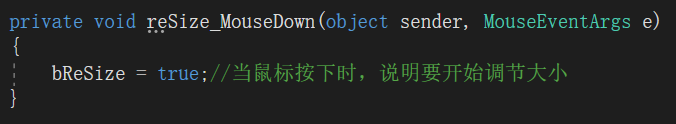
(4c)mouseup事件,在绘画完成之后.鼠标松开.触发结束绘画事件



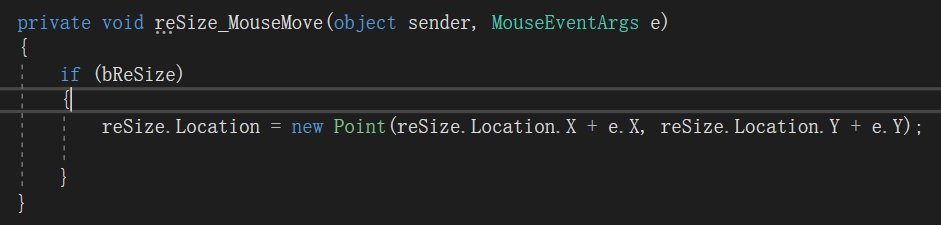
1. 调整画布大小

这一部分与绘制图案在某些方面有相似之处,同样是在鼠标左键按下时开始进行画布大小调节,画布大小会跟随鼠标移动而发生改变,以及在鼠标松开时,大小改变结束,画布大小进行刷新.以下是这三个部分:

(5a)mousedown事件



(5b)mousemove事件,在移动resize块的时候,其位置发生改变,方便后续更新调整新的画布大小

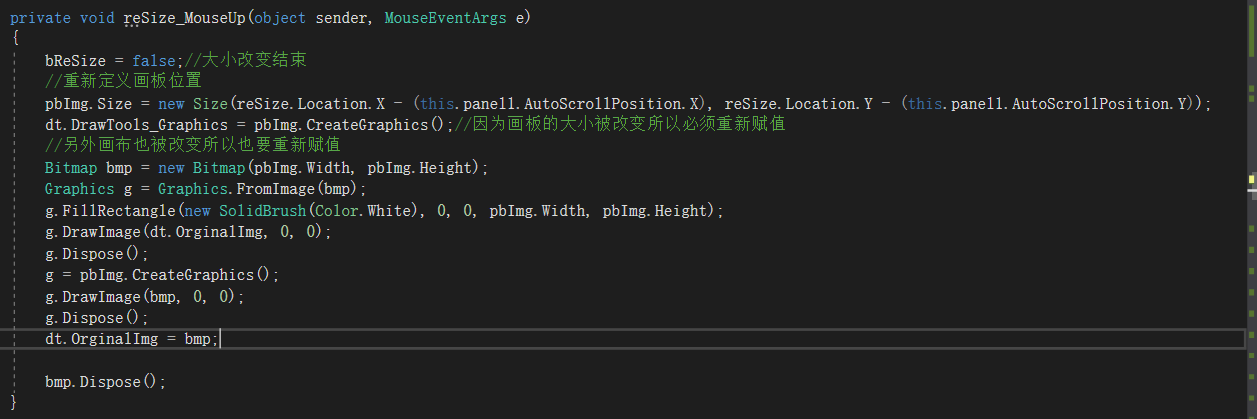


(5c)mouseup事件

在我们调节画板大小的过程中,画板大小可能会超过屏幕区域,所以我们需要实现将autoscroll设置为true;

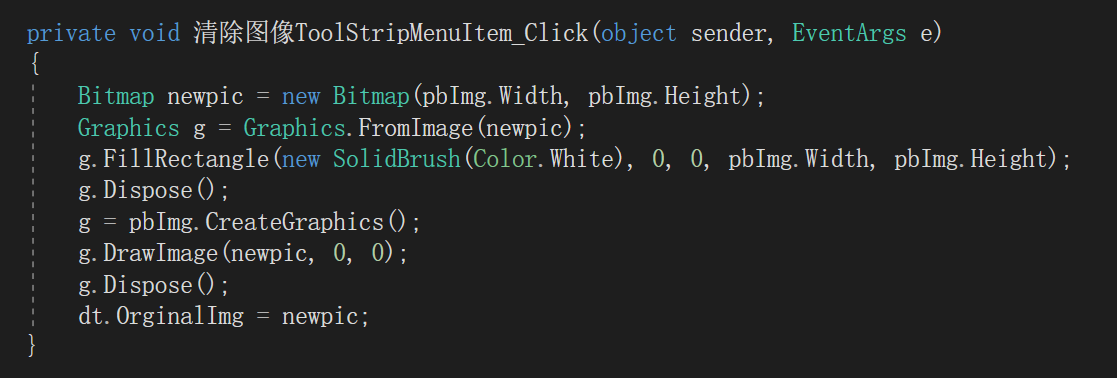
但同时,滚动条的出现会导致其上下移动时并不会自动调整图片的坐标(滚动条上下移动改变的是文档坐标,但是客户区坐标不发生改变(因为location坐标属于客户区坐标,所以在直接计算时就会出现错误))

一种解决这个问题的方案是利用autoscrollposition提供的文档坐标与客户去坐标的偏移量,来确定好最终的坐标位置,代码如下:



1. 清除图像

清除图像的本质在于,用新的画布和画板去覆盖掉原来的画布和画板,代码如下:



1. 在颜色板中调节颜色

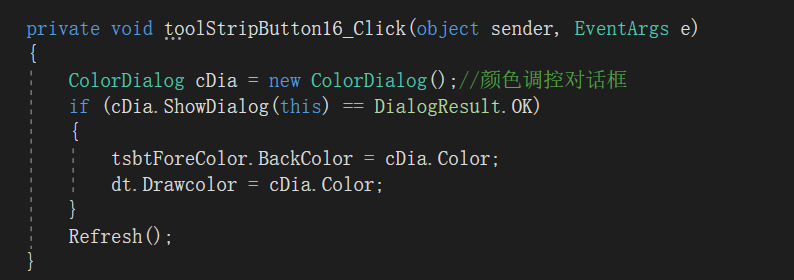
在这一步骤中,我们需要设计方法,当用户下按对应的按钮时可以调整绘图工具的颜色,在主窗体下的颜色有黄色,黑色,蓝色,粉色等颜色,针对这些颜色设计用户左键单击的事件,代码如下(以黄色为例,其他颜色类推)



接下来,按照上面的格式对于绘图工具的其他颜色进行修改

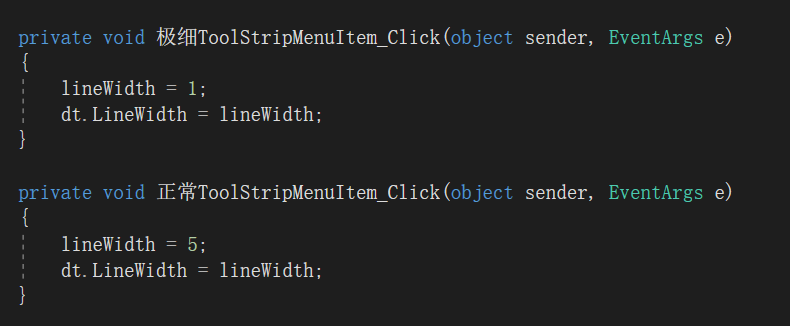
1. 自定义绘图颜色

用的时候,画图软件所提供的颜色并不足够用户使用,因此我们需要设计自定义颜色的模块



1. 一些其他的功能说明:

(9a)上方菜单栏对于线条粗细进行选择,设置极细,正常两个选项,设置内容如下



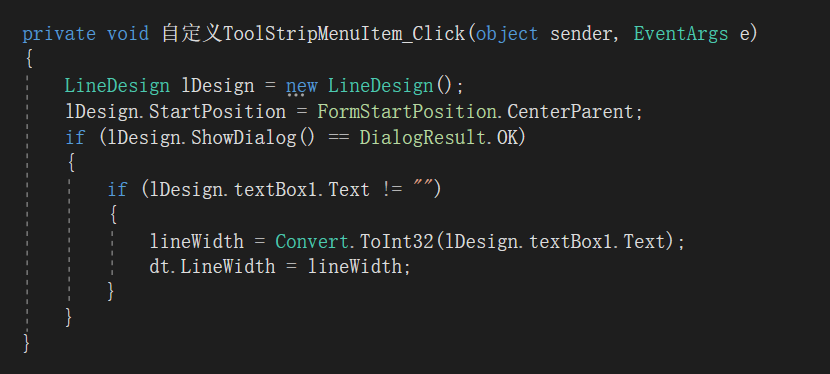
(9b)自定义线条宽度

首先,在用户选择自定义线条宽度之后,需要有另一个窗体弹出并让用户输入可能调节的线条宽度数值,所以在设计这个功能时我们需要额外创建一个窗体,设计如下:

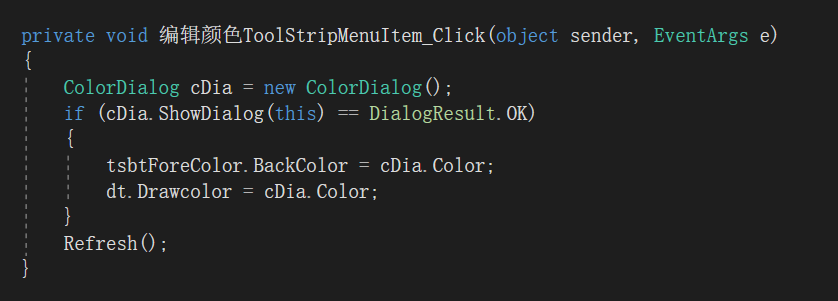
将窗体命名为LineDesign,并设置基本的gon功能,代码如下:



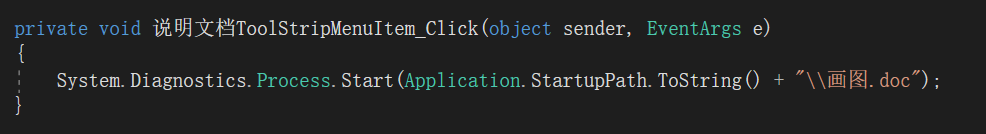
在两个窗体间进行数据传递,代码如下:



(9c)编辑颜色功能

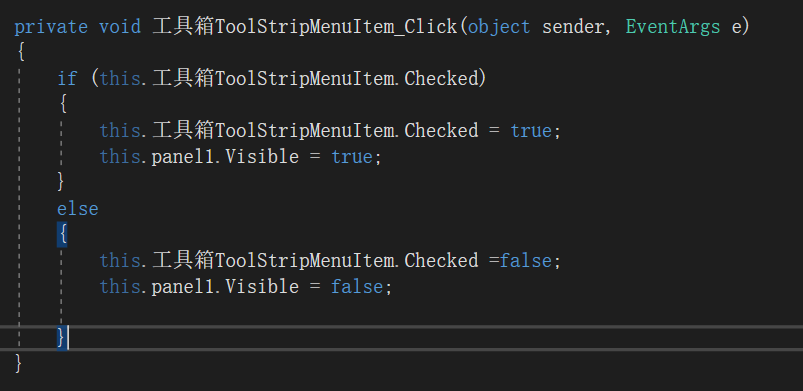


(9d)说明文档功能(即本篇文档)

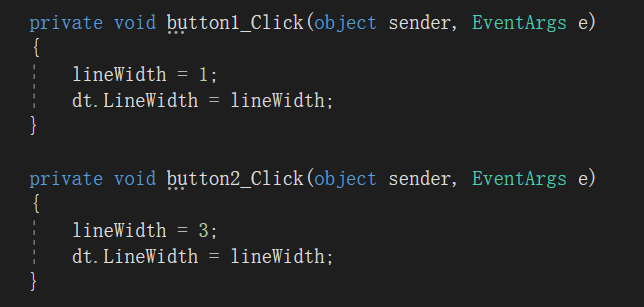


(9e)显示菜单栏可以选择隐藏工具栏,颜料库,状态栏功能

设计要点在于当用户选择隐藏时,将容器设置为不可见,这样就达到了隐藏的目的,以工具栏为例,代码如下:



(9f)设置左下角对于线条宽度的调节



至此,C#画图工具的整体规划和代码实现就写到这里,后续如果还有补充功能的话会进行更新,希望大家可以多多关注.