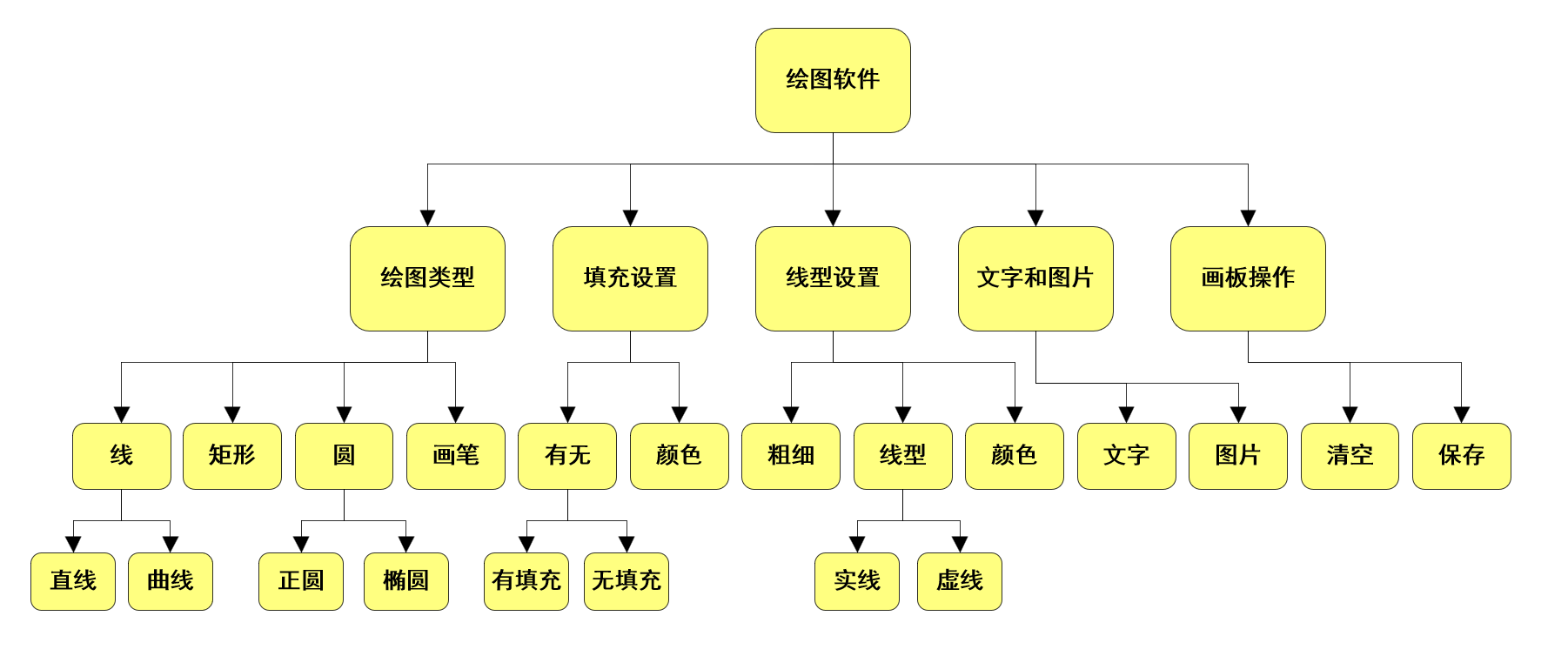
# MFC绘图软件实验报告

## 概述

该绘图程序的设计旨在提供一个功能全面且易于使用的绘图工具，使用户能够通过鼠标完成各种绘图操作。具体的程序设计如下：

1. 提供多种绘图功能：该程序旨在实现直线、曲线、矩形、正圆和椭圆等多种基本图形的绘制功能。用户可以根据自己的需求选择合适的形状，并进行绘制操作。
2. 实现自由绘制功能：除了预定义的图形形状，该程序还提供了自由绘制的功能，用户可以使用鼠标自由绘制任意形状的线条。
3. 支持填充和线型设置：为了增加绘图的灵活性，该程序允许用户选择是否填充绘制的图形，并提供实线和虚线两种线型选择。
4. 提供绘图属性的调整：该程序允许用户根据需要设置线宽、线条颜色和图形填充颜色，以满足个性化绘图需求。
5. 支持添加文字和导入照片：为了丰富绘图内容，该程序还允许用户在绘图窗口中添加文字，并提供了加载照片的功能，使用户能够将自己的照片融入到绘图中。
6. 提供导出功能：为了方便用户保存和分享绘制的图形，该程序支持将当前绘图窗口的内容导出为Bitmap形式的图像文件。



## 设计步骤和流程

1. **用户设置对话框设计**

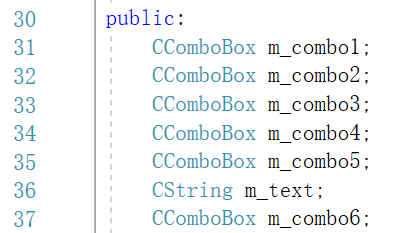
在资源视图中新建一个对话框，用来为用户提供设置操作。在对话框中添加相应的控件（静态文本、复选框、编辑框、按钮），如下图所示：



为新建的对话框添加一个类，类名叫SettingDlg，基类选择CFormView

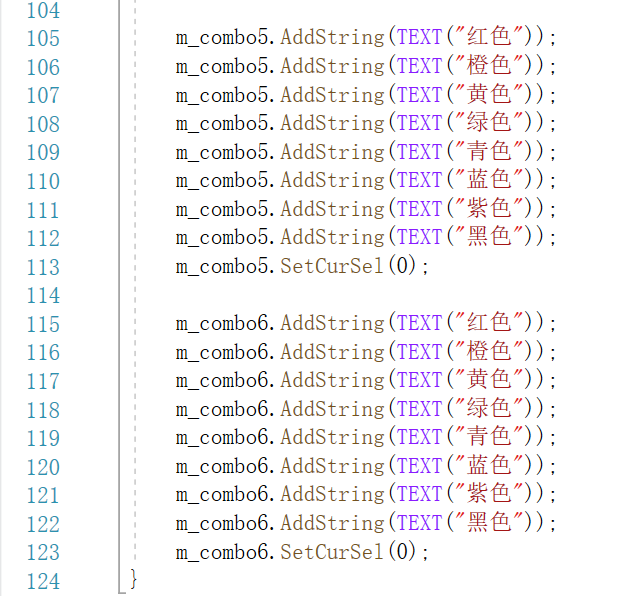


新建类以后，为对话框中的控件添加变量，具体添加情况如下：



重写OnInitialUpdate()函数，在该函数中完成控件的初始化操作：



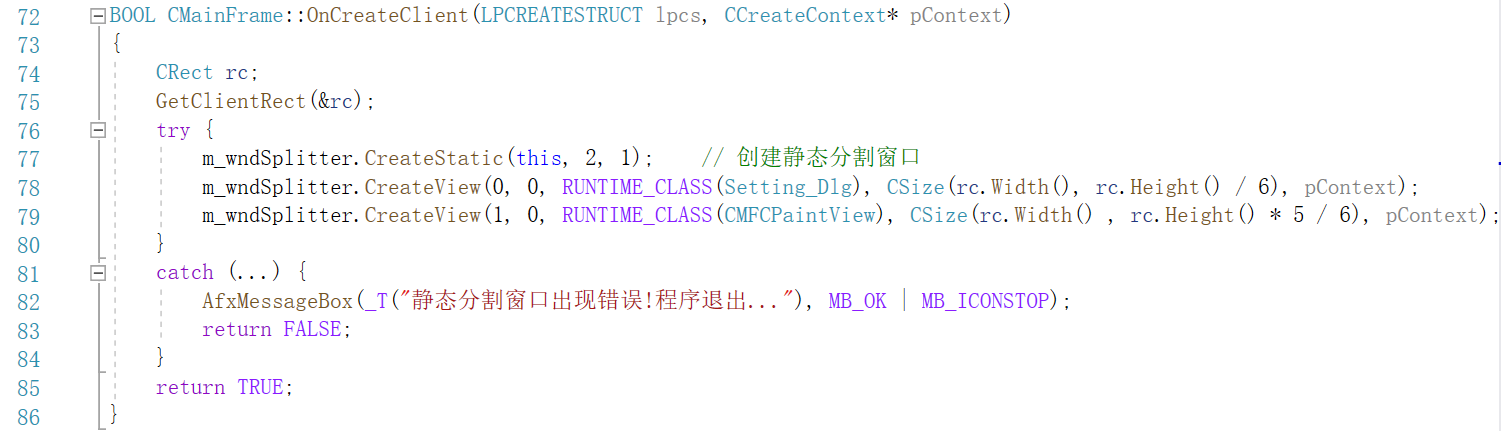


1. **主界面窗口设计**

因为本程序基于单文档视图，可以将窗口一分为二：一个为用户设置区域，一个为绘图区域。在"MainFrm.h"添加对应的分割窗口变量。

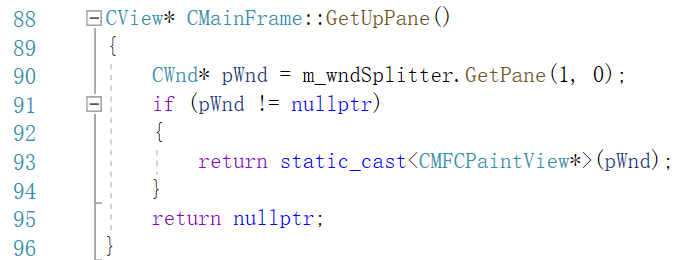


重写OnCreateClient函数，添加分割窗口的代码，窗口一分为二，设置窗口占上方的1/6，绘图窗口占下方的5/6，运行效果如下。



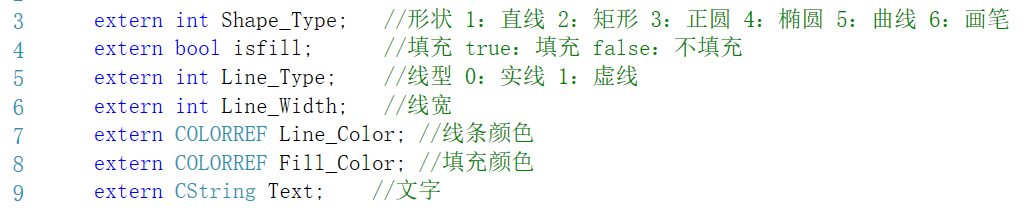


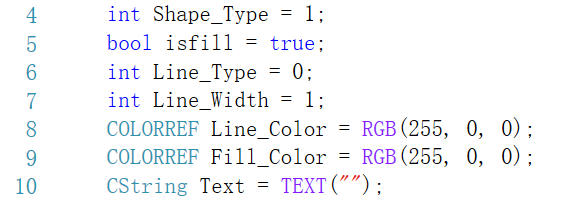
再编写一个接口函数，该函数负责提供View的指针，方便在设置对话框类中调用View中的函数



1. **窗口间数据交互设计**

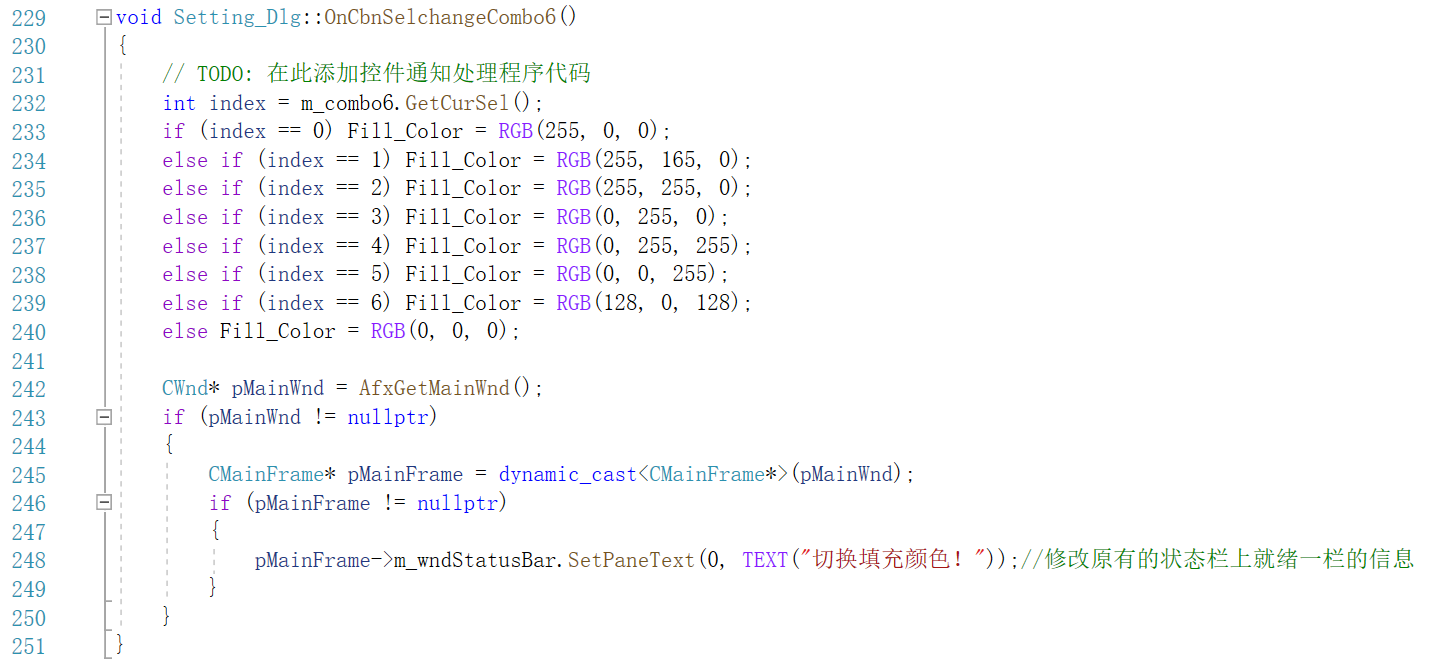
为了方便两个窗口之间的数据交互，这里使用全局变量，用户在设置窗口中进行操作后的信息直接更新到全局变量，绘图窗口通过访问全局变量即可获得当前的绘图状态。





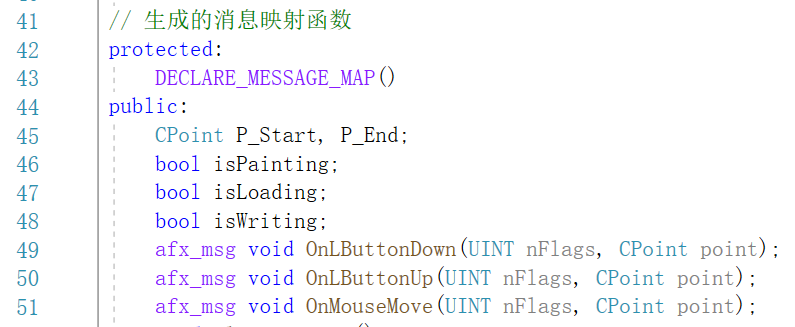
为设置对话框的复选框和按钮添加响应函数，用来更新当前的绘图设置信息。在更新绘图状态后，设置对话框中的函数可以通过View接口更新状态栏中的信息。



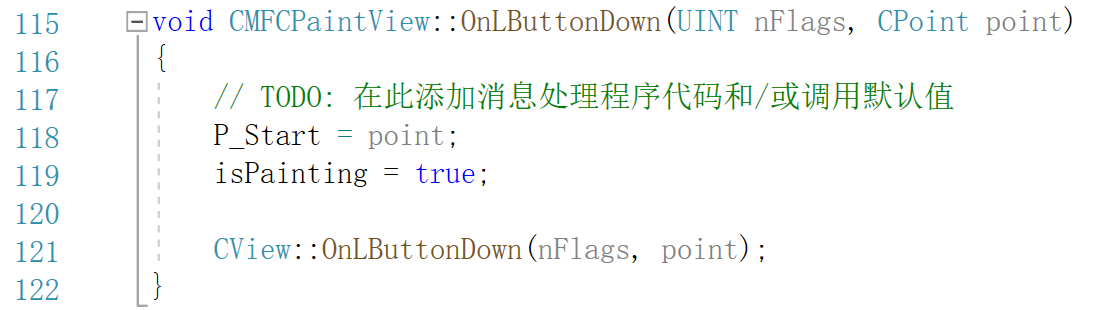
    

1. **鼠标消息处理函数设计**

在View.h中，添加三个标志变量，分别用来判断当前是否在自由绘制、加载图片、绘制文字，添加两个CPoint变量，用来记录鼠标的起始点和终点。最后再添加三个鼠标消息响应函数。



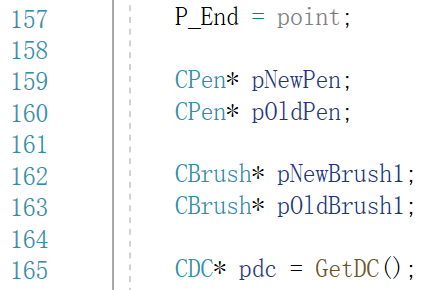
在OnLButtonDown函数中，即鼠标左键按下时，更新起始点的坐标，将自由绘制标志位置为true



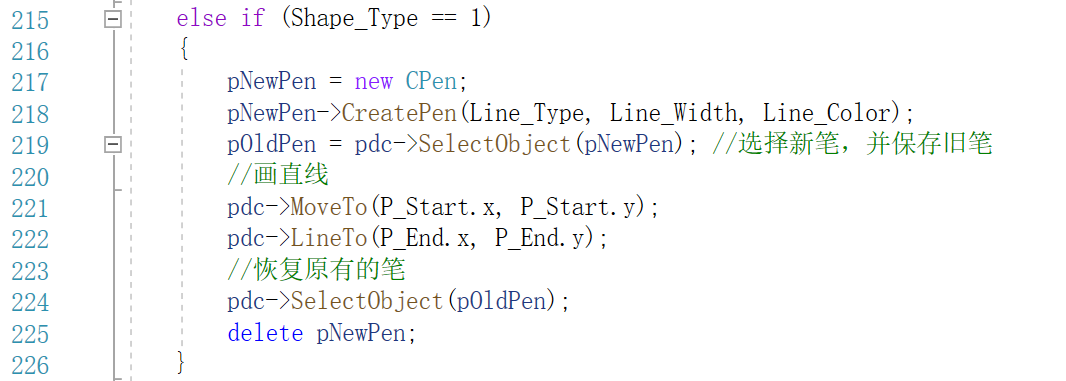
在OnLButtonUp函数中，即鼠标左键抬起时，首先判断是否要绘制文字，当标志位为true时，使用TextOut方法在P\_Start点处绘制文字，效果如下：

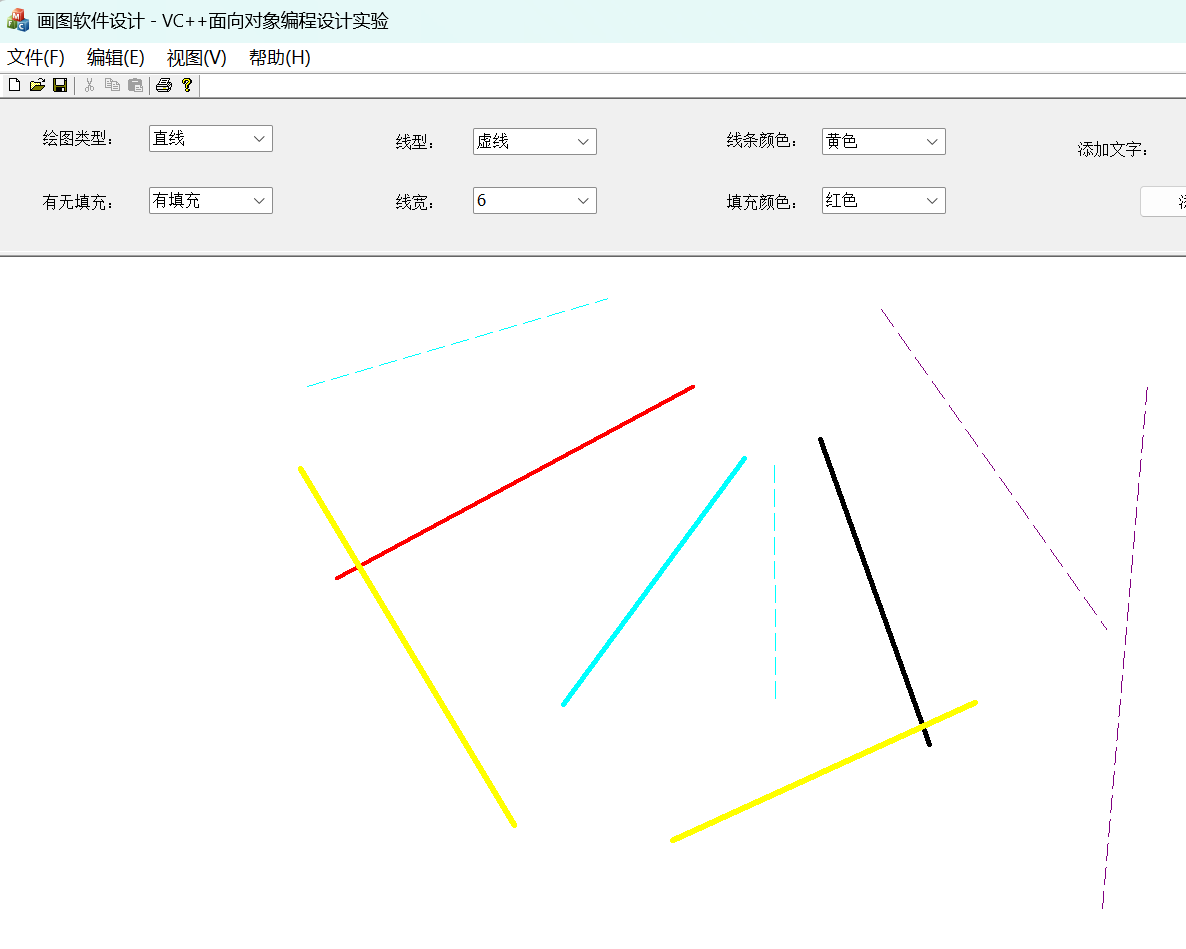
 

更新绘图终点的坐标值，定义新旧画笔与画刷，获得窗口绘图DC

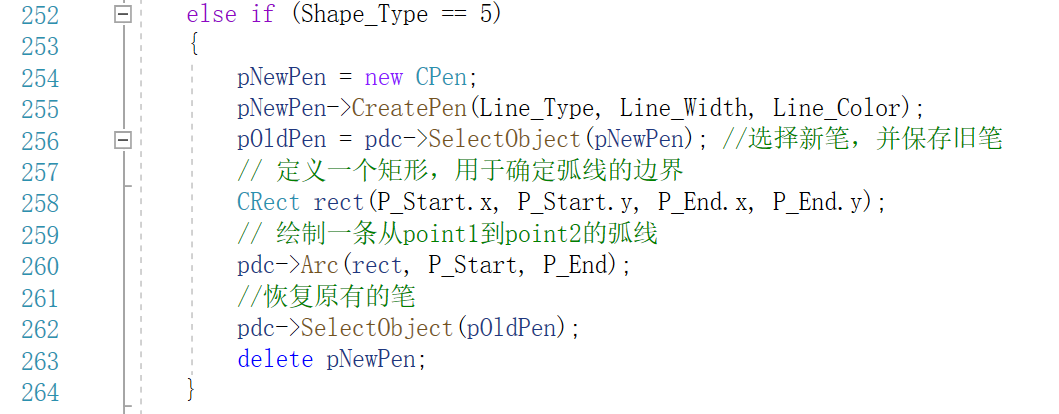


若当前绘制的图形为直线，首先使用CreatePen构造一个当前线型、线宽、颜色的画笔，然后使用MoveTo和LineTo两个函数绘制直线，效果如下：

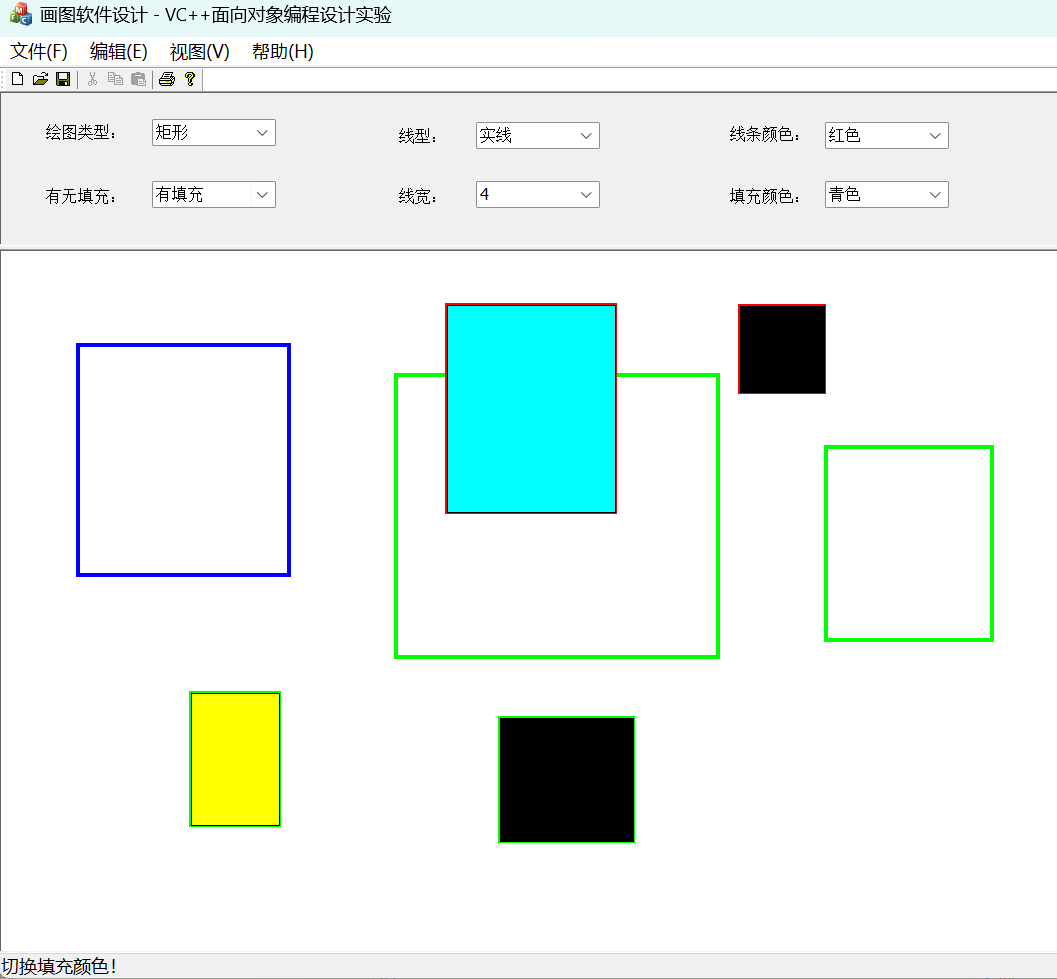




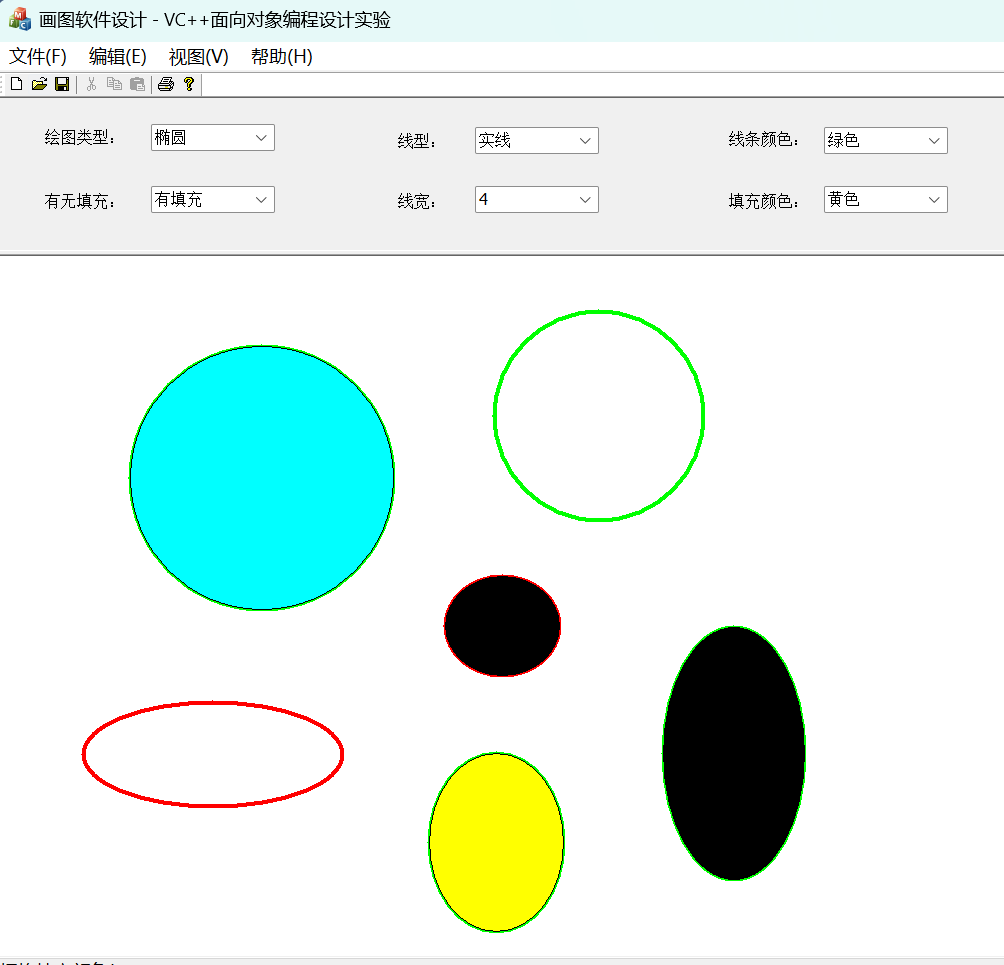
若当前绘制的图形为曲线，首先使用CreatePen构造一个当前线型、线宽、颜色的画笔，然后使用Arc函数绘制曲线，效果如下：



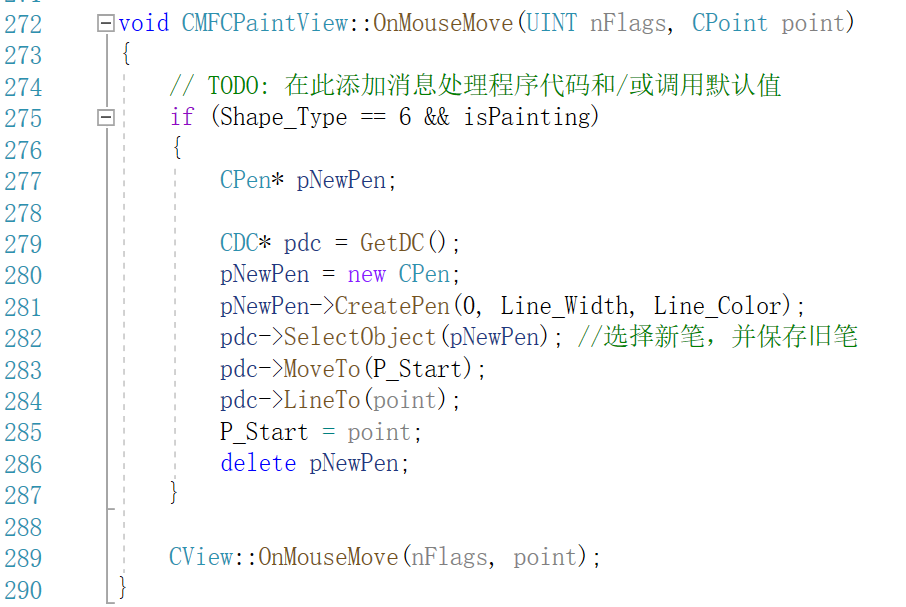
若当前绘制的图形为矩形，首先使用CreatePen构造一个当前线型、线宽、颜色的画笔，用Rectangle函数画出矩形的轮廓。然后判断是否需要填充，若填充，则使用CreateSolidBrush构造一个画刷，用Rectangle函数绘制一个实心矩形，效果如下：

若当前绘制的图形为正圆，首先使用CreatePen构造一个当前线型、线宽、颜色的画笔，画出正圆的轮廓。然后判断是否需要填充，若填充，则使用CreateSolidBrush构造一个画刷，绘制一个实心正圆，效果如下，椭圆同理：

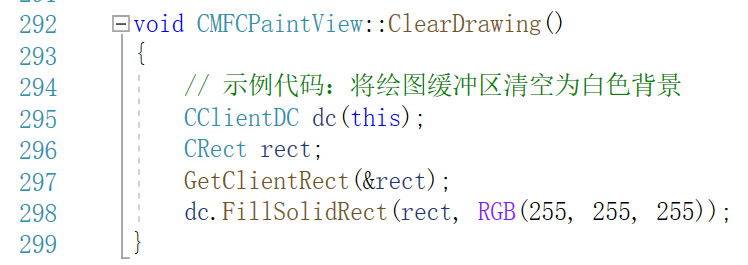


在OnMouseMove中，即当前鼠标正在移动时，首先判断现在是否设置为“画笔“，然后绘制直线，绘制后更新起点和终点的值。

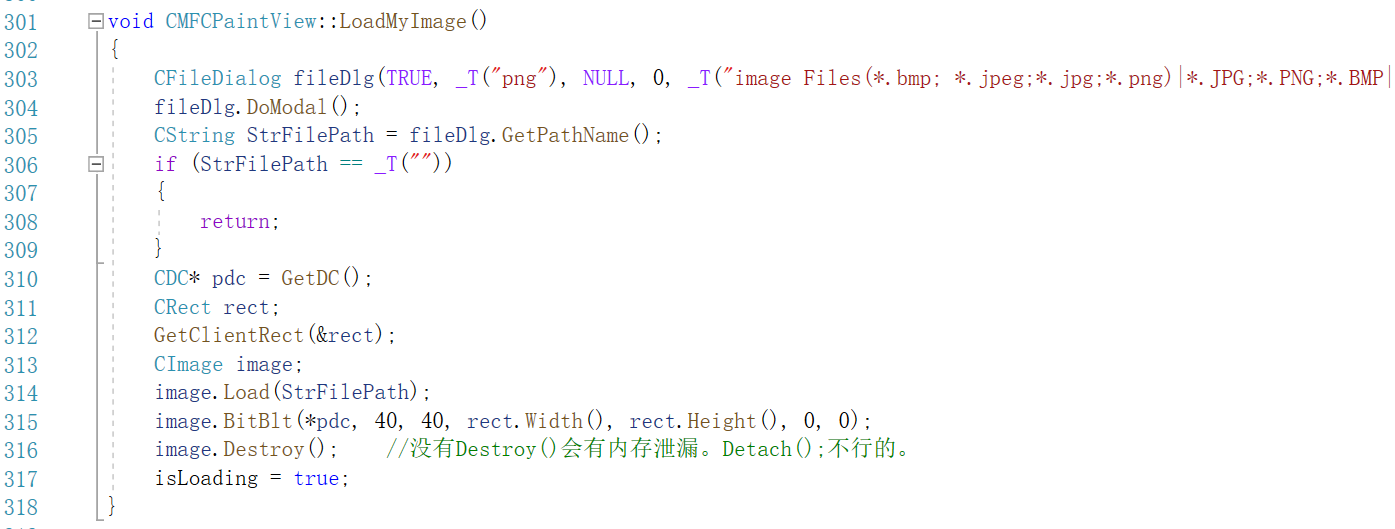
 

1. **加载图片、保存与清空绘图区域设计**

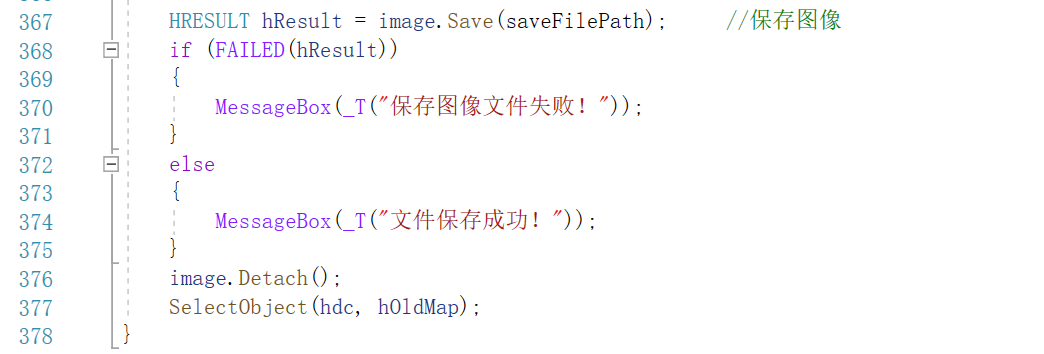
编写ClearDrawing()函数，该函数将绘图区域全部填充为一个白色的矩形，从而达到清空绘图区域的效果。在设置对话框的按钮响应函数中，通过接口函数条用该清空函数即可。

编写LoadMyImage()函数，该函数首先弹出一个文件对话框，可以让用户选择需要显示的图片，用户点下确定后函数记录图片路径，保存在StrFilePath变量中。然后定义一个CImage类加载图片，并把图片显示在窗口上。



编写SavePainting()函数，该函数首先获取当前画布的大小，创建DC并将窗口图片保存在DC中，创建CImage变量，将图片写入指定路径。

## 实验总结

该绘图程序的开发经历了一系列的设计和实现过程，最终实现了一个功能全面且易于使用的绘图工具。以下是对整个项目的总结：

在本项目中，我们成功地利用MFC框架开发了一个基于单文档的绘图程序。该程序具备绘制直线、曲线、矩形、正圆、椭圆等多种形状的功能，用户可以轻松地进行绘图操作。同时，该程序支持画笔的自由绘制，提供填充和线型设置，允许用户调整线宽、线条颜色和图形填充颜色，以及添加文字和加载照片等功能。此外，程序还提供了清空绘图窗口和将绘图导出为Bitmap文件的选项，方便用户进行操作和保存绘图结果。

在开发过程中，我们遇到了一些挑战，例如处理用户的鼠标事件、实现不同形状的绘制算法和提供丰富的绘图选项等。通过充分利用MFC框架提供的类和函数，我们成功地解决了这些问题，并实现了所需的功能。

该绘图程序的用户界面设计简洁明了，易于操作。用户可以通过菜单、工具栏和鼠标交互来完成各种绘图操作。通过对程序进行测试和验证，我们确认了其各项功能的正确性和稳定性。

未来，我们可以考虑进一步改进和扩展该绘图程序。例如，可以增加更多形状的绘制选项，提供更多线型和填充样式的选择，增加图层管理功能等，以满足更多绘图需求。此外，还可以考虑优化程序的性能和界面的美观性，提升用户体验。

总之，通过该绘图程序的开发，我们深入理解了MFC框架的应用和图形绘制的原理，实践了软件开发的各个阶段，获得了宝贵的经验和技能。我们相信这个绘图程序将为用户提供一个强大且易用的工具，帮助他们实现各种创意和绘图需求。