智能系统的设计与仿真

作业三

姓名：石若川 学号：2111381 专业：智能科学与技术

1 实验内容

完成对话框设计。可以选择矩形、圆形和直线的绘制；给定左上角和右下角参数；点击计算，如果形状选择为矩形和圆形，则给出其面积和周长；如果形状选择为直线，则面积编辑框消失，只给出线长。点击重置，则面积和周长重置为零；可以选择实线、虚线和点划线的选择；可以实现颜色的选择。点击确定，在视图输出相应线型、颜色的图形，并在其下方标注形状和颜色 RGB 数值。

2 代码分析

2.1 对话框设计

实验中新建了一个对话框，添加所需要的不同控件，最终设计的对话框如图2.1所示。

图 2.1: 对话框设计

对每个控件添加一个 ID 进行操作，系统添加至 Resorce.h 中。部分 ID 如下所示。

```
#define ID_Reset                4
#define IDD_ABOUTBOX            100
#define IDP_OLE_INIT_FAILED     100
#define IDR_MAINFRAME           128
#define IDR_MYTESTTYPE          130
#define IDD_DIALOG1             310
#define IDC_RATIO_RECT          1001
#define IDC_RADIO_ROUND         1002
#define IDC_RADIO_LINE          1003
#define IDC_RADIO_SOLID         1004
```

```
#define IDC_RADIO_DASH              1005
#define IDC_RADIO_DASHDOT           1006
#define IDC_EDIT_Y1                 1007
#define IDC_EDIT_X1                 1008
#define IDC_EDIT_X2                 1009
#define IDC_EDIT_Y2                 1010
#define IDC_EDIT2                   1011
#define IDC_CALC                    1011
#define IDC_EDIT3                   1012
#define IDC_CALS                    1012
#define IDC_BUTTON_SETCOLOR         1014
#define ID_CAL                      1015
#define IDC_STATIC_C                1017
#define IDC_STATIC_S                1018
#define IDC_STATIC_Y2               1019
#define IDC_STATIC_X1               1020
#define IDC_STATIC_Y1               1021
#define IDC_STATIC_X2               1022
#define ID_DLG                      32772
```

2.2 控制不同形状下对话框的变化

由于不同形状的对话框有所区别，代码中对编辑框和固定文本进行设置，代码如下所示。

```
void CMyDlg::OnBnClickedRatioRect()
{
   // TODO: 在此添加控件通知处理程序代码
   GetDlgItem(IDC_CALC)->ShowWindow(SW_NORMAL);
   GetDlgItem(IDC_CALS)->ShowWindow(SW_NORMAL);
   GetDlgItem(IDC_EDIT_Y2)->ShowWindow(SW_NORMAL);
   Static_C.SetWindowTextW(L"周长");
   Static_S.SetWindowTextW(L"面积");
   Static_X1.SetWindowTextW(L"X1");
   Static_X2.SetWindowTextW(L"X2");
   Static_Y1.SetWindowTextW(L"Y1");
   Static_Y2.SetWindowTextW(L"Y2");
}


void CMyDlg::OnBnClickedRadioRound()
{
   // TODO: 在此添加控件通知处理程序代码
```

```
    GetDlgItem(IDC_CALC)->ShowWindow(SW_NORMAL);
    GetDlgItem(IDC_CALS)->ShowWindow(SW_NORMAL);
    GetDlgItem(IDC_EDIT_Y2)->ShowWindow(SW_HIDE);
    Static_C.SetWindowTextW(L"周长");
    Static_S.SetWindowTextW(L"面积");
    Static_X1.SetWindowTextW(L"x");
    Static_X2.SetWindowTextW(L"r");
    Static_Y1.SetWindowTextW(L"y");
    Static_Y2.SetWindowTextW(L"");
}


void CMyDlg::OnBnClickedRadioLine()
{
    // TODO: 在此添加控件通知处理程序代码
    GetDlgItem(IDC_CALC)->ShowWindow(SW_NORMAL);
    GetDlgItem(IDC_CALS)->ShowWindow(SW_HIDE);
    GetDlgItem(IDC_EDIT_Y2)->ShowWindow(SW_NORMAL);
    Static_C.SetWindowTextW(L"线长");
    Static_S.SetWindowTextW(L"");
    Static_X1.SetWindowTextW(L"X1");
    Static_X2.SetWindowTextW(L"X2");
    Static_Y1.SetWindowTextW(L"Y1");
    Static_Y2.SetWindowTextW(L"Y2");
}
```

2.3 计算与重置

在 CCmyDlg 类中，对“计算”和“重置”按钮的控件编写如下代码，利用 m_Shape 进行分类，分别对周长和面积进行计算。在直线的计算中，由于线长显示在原周长的编辑框中，所以将线长赋值给周长。

```
void CMyDlg::OnBnClickedCal()
{
    // TODO: 在此添加控件通知处理程序代码
    UpdateData(true);
    switch (m_Shape) {
    case 0:
        m_C = 2 * abs(m_x2 - m_x1 + m_y2 - m_y1);
        m_S = abs(m_x2 - m_x1) * abs(m_y2 - m_y1);

        break;
```

```
    case 1:
        m_C = int(2 * 3.14 * m_x2);
        m_S = int(3.14 * m_x2 * m_x2);
        break;
    case 2:
        m_LineLength = int(sqrt(pow(m_x2 - m_x1, 2) + pow(m_y2 - m_y1, 2)));
        m_C = m_LineLength;
        break;
    }
    UpdateData(false);
}


void CMyDlg::OnBnClickedReset()
{
    // TODO: 在此添加控件通知处理程序代码
    m_C = 0;
    m_S = 0;
    m_LineLength = 0;
    UpdateData(false);
}
```

2.4 创建模态对话框

在 CMYTESTView 类中添加响应消息，当点击菜单栏中的“对话框”时创建模态对话框。另外当点击对话框中的“确认”按钮时，将对话框中所设置的形状、线型、位置赋值给 CMYTESTView 类中的成员变量。由于圆形的位置通过圆心和半径进行设置，所以需要进行一定的变换，转换为 CRect 的形式，方便后续绘制。

```
void CMYTESTView::OnDlg()
{
    // TODO: 在此添加命令处理程序代码
    CMyDlg Dlg;
    //Dlg.DoModal();

    if (Dlg.DoModal() == IDOK) {
        m_isDraw = true;
        m_Shape = Dlg.m_Shape; // 绘制形状
        m_LineType = Dlg.m_LineType; // 线的类型

        switch (m_Shape) {
        case 0:
```

```
        rect.left = Dlg.m_x1;
        rect.right = Dlg.m_x2;
        rect.top = Dlg.m_y1;
        rect.bottom = Dlg.m_y2;
        break;
      case 1:
        rect.left = Dlg.m_x1 - Dlg.m_x2; // 圆心-半径
        rect.right = Dlg.m_x1 + Dlg.m_x2; // 圆心+半径
        rect.top = Dlg.m_y1 - Dlg.m_x2;
        rect.bottom = Dlg.m_y1 + Dlg.m_x2;
        break;
      case 2:
        rect.left = Dlg.m_x1;
        rect.right = Dlg.m_x2;
        rect.top = Dlg.m_y1;
        rect.bottom = Dlg.m_y2;
        break;
      }

      color = Dlg.color;
      Invalidate();
    }
}
```

2.5 绘制形状

在 CMYTESTView 类的 OnDraw 函数中编写绘制图像的代码，利用 m_LineType 和 m_Shape 变量进行分类，分别设置不同的线型和形状，并在绘制的图像下方显示所绘制的图像和 RGB 值。

```
void CMYTESTView::OnDraw(CDC* pDC)
{
  CMYTESTDoc* pDoc = GetDocument();
  ASSERT_VALID(pDoc);
  if (!pDoc)
    return;

  // TODO: 在此处为本机数据添加绘制代码

  CPen newPen; CPen * pOldPen;
  switch(m_LineType) {
    case 0: newPen.CreatePen(PS_SOLID, 1, color); break;
    case 1: newPen.CreatePen(PS_DASH, 1, color); break;
```

```
14      case 2: newPen.CreatePen(PS_DASHDOT, 1, color); break;
15    }
16    pOldPen = pDC->SelectObject(&newPen);
17    CString str;
18    switch(m_Shape) {
19      case 0:
20        pDC->Rectangle(rect);
21        pDC->SetTextColor(RGB(0, 0, 0));
22        if (m_isDraw)
23        {
24          str.Format(_T(" (形状: 矩形, R: %d,G: %d,B: %d) "), GetRValue(color),
              GetGValue(color), GetBValue(color));
25          pDC->TextOut(rect.left, rect.bottom, str);
26        }
27        break;
28      case 1:
29        pDC->Ellipse(rect);
30        pDC->SetTextColor(RGB(0, 0, 0));
31        if (m_isDraw)
32        {
33          str.Format(_T(" (形状: 圆形, R: %d,G: %d,B: %d) "), GetRValue(color),
              GetGValue(color), GetBValue(color));
34          pDC->TextOut(rect.left, rect.bottom, str);
35        }
36        break;
37      case 2:
38        pDC->MoveTo(CPoint(rect.left, rect.top));
39        pDC->LineTo(CPoint(rect.right, rect.bottom));
40        if (m_isDraw)
41        {
42          str.Format(_T(" (形状: 直线, R: %d,G: %d,B: %d) "), GetRValue(color),
              GetGValue(color), GetBValue(color));
43          pDC->TextOut(rect.left, rect.bottom, str);
44        }
45        break;
46    }
47    pDC->SelectObject(pOldPen);
48  }
```

3 结果展示

实验中在菜单栏中创建了“操作”菜单项，并在下拉菜单中添加了“对话框”菜单项，如图3.2所示。

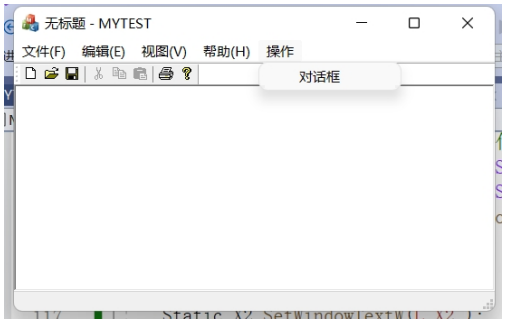


图 3.2: 菜单栏

点击“对话框”选项，弹出默认对话框如图3.3所示，默认形状为矩形，默认线型为实线，默认颜色为红色。

图 3.3: 默认对话框

选择对话框中的不同形状，下方绘制位置和统计的输入项会有所改变，如图3.4。矩形下，绘制位置按照给定的左上角 XY 坐标和右下角 XY 坐标进行绘制，统计结果输出面积和周长；圆形下，绘制位置按照给定的圆心 XY 坐标和半径进行绘制，统计结果输出面积和周长；直线下，绘制位置按照给定的左上角 XY 坐标和右下角 XY 坐标进行绘制，统计结果线长。

(a) 矩形 (b) 圆形 (c) 直线

图 3.4: 选择不同形状

分别在不同形状中输入绘制位置，点击“计算”按钮，结果如图3.5所示。矩形和圆心输出了周长

和面积的计算结果，直线输出了线长的计算结果。点击“重置”按钮，周长、面积和线长中的值会被清零。

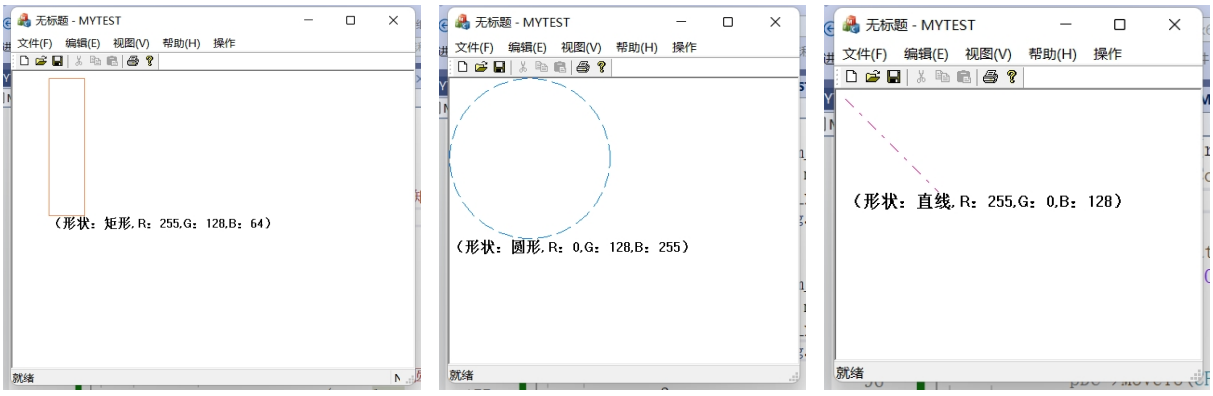
(a) 矩形 (b) 圆形 (c) 直线

图 3.5: 不同形状的统计结果

点击“颜色设置”按钮，会弹出颜色对话框，如图3.6所示。

图 3.6: 颜色对话框

选择不同的颜色和线型进行绘制，得到图3.7的结果。绘图的下方会显示绘制的形状和 RGB 结果。



(a) 矩形 (b) 圆形 (c) 直线

图 3.7: 不同颜色和线型的绘图结果

4 实验总结

通过本次实验，我熟练掌握了 MFC 对话框编程和绘图的基本技能，了解了如何通过用户交互来控制图形的绘制和属性设置。在实现过程中，我遇到了几个挑战，例如如何在视图中正确绘制不同线型和颜色的图形，如何计算并显示图形的面积和周长。这些问题都在逐步调试中得以解决。总体来说，本次实验不仅巩固了我对 MFC 编程的理解，也提高了我处理复杂用户界面和绘图需求的能力，为今后的图形编程打下了坚实的基础。