# 第19次实验

# 基于MFC的C++程序设计

**1、实验学时：**2个课时。

**2、实验类型：**设计性实验。

**3、实验目的：**

（1）了解MFC的类库层次、常用的MFC类；

（2）理解MFC类库的继承方式和形成机制；

（3）掌握基于MFC类库设计C++程序、自行构建C++类库的方法。

**4、实验内容：**

**实验题目1：设计画图程序**

（1）程序功能简介

设计一个画图程序，实现以下功能：

* 打开程序可执行exe文件，在绘图下拉菜单中可分别设置绘制的图形形状，如直线、曲线、矩形及椭圆，线宽选项，有1-6可供选择，线型选项有实线、虚线、点线和点段线供设置，还可以设置线色以及填充色，通过弹出的颜色对话框选择需要的颜色，如果不选择线宽、线色以及填充色，则按默认的画笔，画刷来绘制选择的图形。
* 选择好图形后，通过鼠标可以绘制出相应的直线，矩形或椭圆，鼠标的按下确定图形的起点，鼠标的拖动则确定了图形的终点，即通过鼠标的拖动来决定图形的大小，当鼠标弹起，此图形则绘制完毕。

（2）程序设计说明

① 创建单文档工程

创建一个MFC AppWizard[exe]工程，命名为“Draw”，如图1所示，并创建单文档，如图2所示。创建成功后，系统自动生成相应的类，如图3所示。



图1 创建工程

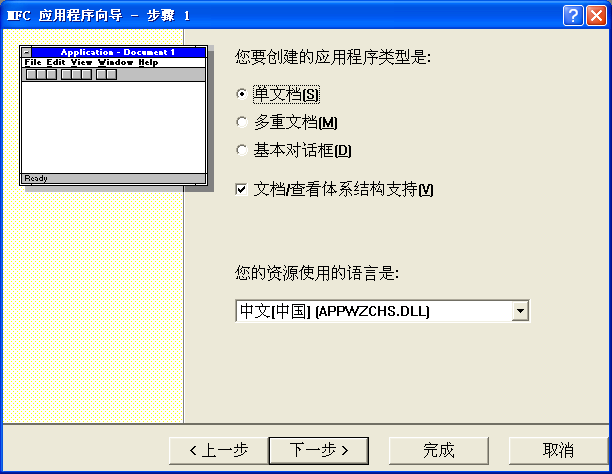


图2 创建单文档

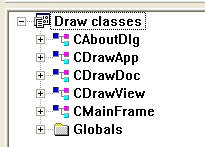


图3 生成类

② 编辑菜单

添加需要的菜单项，如图4所示；并在菜单的属性中设定好所对应的ID，如图5所示，各项菜单对应的ID如表1所示（其中线宽菜单为弹出菜单，只需在菜单项目属性中的弹出选项前打勾即可，分隔线亦只需在菜单项目属性中选中分隔符选项即可）。

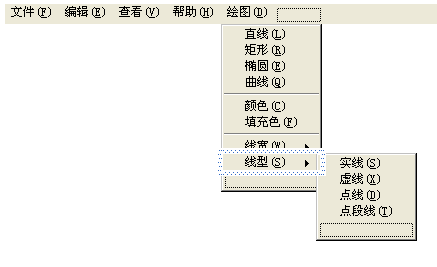
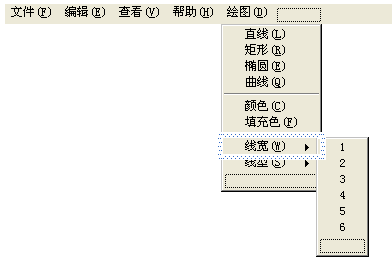


图4 添加菜单项



图5 设置菜单ID

表1 菜单项对应ID

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 菜单名 | ID | 菜单名 | ID | 菜单名 | ID |
| 直线 | ID\_LINE | 线宽1 | ID\_W1 | 实线 | ID\_SOLID |
| 矩形 | ID\_RECT | 线宽2 | ID\_W2 | 虚线 | ID\_DASH |
| 椭圆 | ID\_ELLIPSE | 线宽3 | ID\_W3 | 点线 | ID\_DOT |
| 颜色 | ID\_COLOR | 线宽4 | ID\_W4 | 点段线 | ID\_DASHDOT |
| 填充色 | ID\_FILLCOLOR | 线宽5 | ID\_W5 |  |  |

建立类向导，在视图类CDrawView中，对各菜单项添加对应的COMMAND消息处理函数，部分菜单项还添加对应的UPDATE\_COMMAND\_UI消息函数，如图6所示。

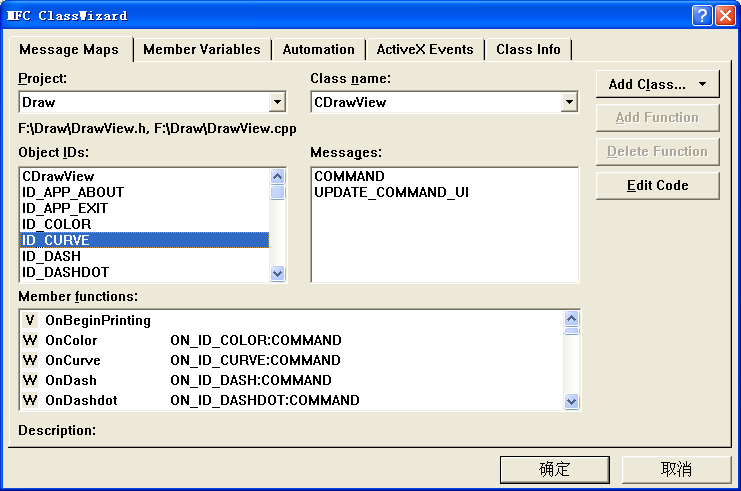


图6 建立类向导

③ 在相应消息函数处添加代码，实现其功能：

建立类向导，在视图类CDrawView中，添加WM\_PAINT消息，以及WM\_LBUTTONDOWN、WM\_LBUTTONUP、WM\_MOUSEMOVE消息，并在其中添加相应的代码。

③ 关键功能实现

* 选择要绘制的图形

在直线、矩形以及椭圆的消息处理函数里为shape分别赋值为1,2,3，在OnPaint()函数里添加3个if语句的代码来这实现这三个图形的绘制：当选择画直线时，shape=1，则执行绘制直线代码，当选择画矩形时，shape =2,则执行绘制矩形的代码，当性选择画椭圆时,shape=3,则执行绘制椭圆的代码。

* 改变线宽

在OnPaint()创建的画笔中，pen.CreatePen(penstyle,w, m\_nColor)，线宽处用w表示，则改变w的值即改变了画笔的粗细。选择菜单绘图→线宽选择划线宽度值，此时所选的线宽值就赋给线宽变量w。若不选择线宽值，则w=1(即默认线宽值为1)。

* 改变线型

在OnPaint()创建的画笔中，pen.CreatePen(penstyle,w, m\_nColor)，线型处用penstyle表示，则改变penstyle的值即改变了画笔的类型。选择菜单绘图→线型选择线型，此时所选的线型就赋给线型变量penstyle。若不选择线型值，则penstyle=ID\_SOLID(即默认线型为实线)。

* 改变线色

在OnPaint()创建的画笔中，pen.CreatePen(penstyle,w, m\_nColor)，线的颜色处用m\_nColor表示，则改变m\_nColor即改变了画笔的颜色。选择菜单绘图→颜色，会弹出通过WM\_COLORREF从系统中引入的颜色选择对话框，选择颜色后，点击确定则相应颜色值就赋给变量m\_nColor，此时画笔的颜色就会发上变化。若不选择颜色，则默认（线）颜色为黑色。

* 改变填充色

在OnPaint()创建的画刷中，brush.CreateSolidBrush(m\_fColor)，颜色就是用m\_fColor表示的，改变m\_fColor即改变了画刷的颜色。选择菜单绘图→填充色，会弹出通过WM\_COLORREF从系统中引入的颜色选择对话框，选择颜色后，点击确定则相应颜色值就赋给变量m\_fColor，此时画刷的颜色就会发上变化。若不选择颜色，则默认填充色为黑色。

* 绘制图形

鼠标左键按下，则捕获鼠标此时的位置坐标为画图的起点，此时在OnLButtonDown()函数里，将捕获的坐标值赋给定义作为起点的变量opoint；按下鼠标左键拖动鼠标时，捕捉鼠标当前位置坐标值为终点坐标值，此时在OnMouseMove()函数里，把捕获的坐标值赋给定义作为终点的变量cpoint，OnPaint()中绘图的代码中的坐标都得到了，进行绘图；鼠标左键弹起时，此时绘图结束，鼠标弹起时所在位置就是画图的最终终点位置，画图结束后，刷新窗口，以便重新画图，此时执行的是OnLButtonUp()函数。

④ 点击编译、链接，运行后的窗口如图7所示。

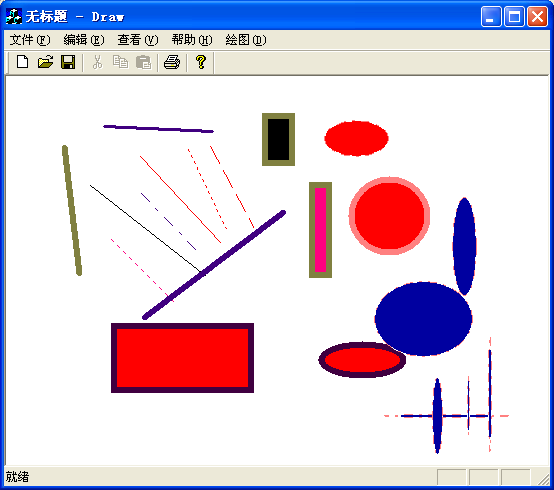


图7 运行结果

（3）程序调试运行

运行程序查看结果，并进行源代码调试和优化。

**参考代码如下：**

① 在视图类构造函数中对各定义变量初始化

CDrawView::CDrawView()

{

w=1;//初始状态时线宽

shape=1;//初始状态时形状

m\_nColor=RGB(0,0,0);//初始状态时线色

m\_fColor=RGB(0,0,0);//初始状态时填充色

penstyle=PS\_SOLID;//初始化线型为实线

}

② 形状函数

void CDrawView::OnLine(){ shape=1; }

void CDrawView::OnRect(){ shape=2; }

void CDrawView::OnEllipse(){ shape=3; }

③ 线宽改变函数

void CDrawView::OnW1(){ w=1; }

void CDrawView::OnW2(){ w=2; }

void CDrawView::OnW3(){ w=3; }

void CDrawView::OnW4(){ w=4; }

void CDrawView::OnW5(){ w=5; }

void CDrawView::OnW6(){ w=6; }

④ 线型选择函数

void CDrawView::OnSolid(){ penstyle=PS\_SOLID; }

void CDrawView::OnDash(){ penstyle=PS\_DASH; }

void CDrawView::OnDot(){ penstyle=PS\_DOT; }

void CDrawView::OnDashdot(){ penstyle=PS\_DASHDOT; }

⑤ 颜色选择函数

void CDrawView::OnColor()

{

CColorDialog Color;//创建颜色对话框

if(Color.DoModal()==IDOK)//如果用户点击OK按钮

{

m\_nColor=Color.GetColor();//更改颜色,把颜色赋到m\_nColor

}

}

⑥ 填充色选择函数

void CDrawView::OnFullcolor()

{

CColorDialog FullColor;//创建颜色对话框

if(FullColor.DoModal()==IDOK)//如果用户点击OK按钮

{

m\_fColor=FullColor.GetColor();//更改颜色,把颜色赋到m\_nColor

}

}

⑦ 单击、移动、绘画函数

void CDrawView::OnLButtonDown(UINT nFlags, CPoint point)

{

m\_Down=TRUE; //鼠标按下

SetCapture(); //获取坐标

opoint=point; //把鼠标所在坐标赋给起点坐标opoint

ReleaseCapture(); //释放坐标

CView::OnLButtonDown(nFlags, point);

}

void CDrawView::OnLButtonUp(UINT nFlags, CPoint point)

{

m\_Down=FALSE; //鼠标不按下，即弹起

Invalidate(0); //刷新

CView::OnLButtonUp(nFlags, point);

}

void CDrawView::OnMouseMove(UINT nFlags, CPoint point)

{

if(m\_Down) //如果鼠标按下

{

cpoint=point; //把鼠标所在点坐标赋给终点坐标cpoint

CClientDC dc(this);

if(4==shape) //画曲线

{

CPen pen(PS\_SOLID,10,RGB(255,0,0));

CPen \*oldpen=dc.SelectObject(&pen);

dc.MoveTo(opoint);

dc.LineTo(cpoint);

dc.SelectObject(oldpen);

opoint=cpoint;

}

}

CView::OnMouseMove(nFlags, point);

}

void CDrawView::OnPaint()

{

CPaintDC dc(this); //device context for painting

CPen pen; //定义画笔

CBrush brush; //定义画刷

pen.CreatePen(penstyle,w, m\_nColor); //创建画笔

brush.CreateSolidBrush(m\_fColor); //创建画刷

dc.SelectObject(&pen); //获取画笔

dc.SelectObject(&brush); //获取画刷

if(1==shape) //画直线

{

dc.MoveTo(opoint.x,opoint.y);

dc.LineTo(cpoint.x,cpoint.y); //绘制直线，从opoint到cpoint连线

}

if(2==shape) //画矩形

{

dc.Rectangle(opoint.x,opoint.y,cpoint.x,cpoint.y); //绘制矩形

}

if(3==shape) //画椭圆

{

dc.Ellipse(opoint.x,opoint.y,cpoint.x,cpoint.y); //绘制椭圆

}

}

⑧ 菜单勾选函数

void CDrawView::OnUpdateLine(CCmdUI\* pCmdUI)

{ pCmdUI->SetCheck(shape==1); //shape为1时，直线选项前打上勾 }

void CDrawView::OnUpdateRect(CCmdUI\* pCmdUI)

{ pCmdUI->SetCheck(shape==2); //shape为2时，直线选项前打上勾 }

void CDrawView::OnUpdateEllipse(CCmdUI\* pCmdUI)

{ pCmdUI->SetCheck(shape==3); //shape为3时，直线选项前打上勾 }

void CDrawView::OnUpdateW1(CCmdUI\* pCmdUI)

{ pCmdUI->SetRadio(w==1); //当线宽为1时，此选项前标有圆点 }

void CDrawView::OnUpdateW2(CCmdUI\* pCmdUI)

{ pCmdUI->SetRadio(w==2); //当线宽为2时，此选项前标有圆点 }

void CDrawView::OnUpdateW3(CCmdUI\* pCmdUI)

{ pCmdUI->SetRadio(w==3); //当线宽为3时，此选项前标有圆点 }

void CDrawView::OnUpdateW4(CCmdUI\* pCmdUI)

{ pCmdUI->SetRadio(w==4); //当线宽为4时，此选项前标有圆点 }

void CDrawView::OnUpdateW5(CCmdUI\* pCmdUI)

{ pCmdUI->SetRadio(w==5); //当线宽为5时，此选项前标有圆点 }

void CDrawView::OnUpdateW6(CCmdUI\* pCmdUI)

{ pCmdUI->SetRadio(w==6); //当线宽为6时，此选项前标有圆点 }

void CDrawView::OnUpdateSolid(CCmdUI\* pCmdUI)

{ pCmdUI->SetRadio(penstyle==PS\_SOLID); //当线型为实线时，此选项前标有圆点}

void CDrawView::OnUpdateDash(CCmdUI\* pCmdUI)

{ pCmdUI->SetRadio(penstyle==PS\_DASH); //当线型为虚线时，此选项前标有圆点 }

void CDrawView::OnUpdateDot(CCmdUI\* pCmdUI)

{ pCmdUI->SetRadio(penstyle==PS\_DOT); //当线型为点线时，此选项前标有圆点 }

void CDrawView::OnUpdateDashdot(CCmdUI\* pCmdUI)

{ pCmdUI->SetRadio(penstyle==PS\_DASHDOT); //当线型为点段线时，此选项前标有圆点 }

**执行结果如下：**

略。

**5、实验结果：**

参考实验报告模板，撰写本次实验的实验报告，并提交实验报