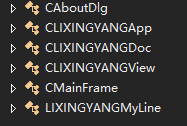
可视化程序设计题目要求:

在画线程序的基础上, 实现绘制矩形, 然后进行显示和保存的功能.

整个项目的类图:



其中, Doc表示对数据的操作, View表示页面的显示, 而MyLine表示由线段构成的矩形.

1. 实现绘制矩形的思路和代码

一个矩形由四条线段组成, 所以, 绘制四条线段就可以实现矩形的绘制.

代码示例:

void CMyLine::OnDraw(CDC \*pDC)

{

// 一个矩形的四边分解为四条线段

pDC->MoveTo(m\_pointStart.x,m\_pointStart.y);

pDC->LineTo(m\_pointStart.x,m\_pointEnd.y);//左竖

pDC->MoveTo(m\_pointStart.x,m\_pointEnd.y);

pDC->LineTo(m\_pointEnd.x,m\_pointEnd.y);//下横

pDC->MoveTo(m\_pointStart.x,m\_pointStart.y);

pDC->LineTo(m\_pointEnd.x,m\_pointStart.y);//上横

pDC->MoveTo(m\_pointEnd.x,m\_pointStart.y);

pDC->LineTo(m\_pointEnd.x,m\_pointEnd.y);//右竖

}

View类中核心的三个函数:

OnMouseMove(),OnLButtonUp(),OnLButtonDown()

// CMyDrawLineView 消息处理程序

// 按住鼠标左键移动

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

{

if (m\_bDraw) {

CClientDC dc(this);

dc.SetROP2(R2\_NOT);

CBrush \*pBrush=CBrush::FromHandle((HBRUSH)GetStockObject(NULL\_BRUSH));

CBrush \*OldBrush=dc.SelectObject(pBrush);

dc.MoveTo(m\_pointOrigin);

//擦除

dc.Rectangle(CRect(m\_pointOrigin,m\_pointCur));

dc.MoveTo(m\_pointOrigin);

//绘制

dc.Rectangle(CRect(m\_pointOrigin,point));

m\_pointCur = point;

OldBrush=pBrush;

}

// 鼠标只要移动就调用

CView::OnMouseMove(nFlags, point);

}

// 抬起鼠标左键

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

{

// TODO: 在此添加消息处理程序代码和/或调用默认值

if (m\_bDraw) {

m\_bDraw = FALSE;

ReleaseCapture();

ClipCursor(NULL);//释放鼠标使用权限（即鼠标可以全屏移动）

CMyDrawLineDoc\* pDoc = GetDocument();

pDoc->AddLine(m\_pointOrigin,point);

pDoc->ClearArray2Lines();

}

CView::OnLButtonUp(nFlags, point);

}

// 按住鼠标左键

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

{

// TODO: 在此添加消息处理程序代码和/或调用默认值

m\_pointOrigin = point;

m\_pointCur = point;

SetCapture();

m\_bDraw = TRUE;

//定义矩形区域

RECT rect;

GetClientRect(&rect);//取得区域rect

ClientToScreen(&rect);

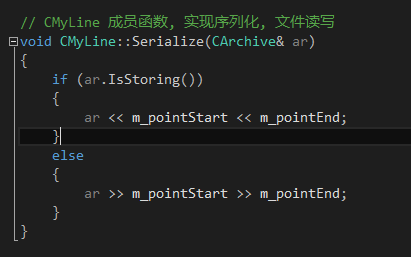
ClipCursor(&rect);

CView::OnLButtonDown(nFlags, point);

}

1. 文件保存的实现

即对象序列化的过程



以及Doc里面的函数

