# CrectTracker类的使用

CrectTracker俗称橡皮筋类，本例演示了它的简单的使用方法。本实例模拟了Windows中这样一个操作：用鼠标选择文件夹的时候，系统会实时的根据鼠标的位置绘制一个虚线框矩形，在与该矩形有交集的文件夹会变成蓝色，表示被选中，而没有交集的文件夹保持原来的状态。实例还模拟了在绘图工具中用鼠标拖动以改变图形大小比例的形为：

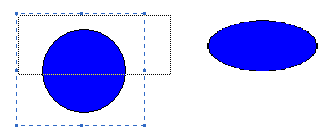
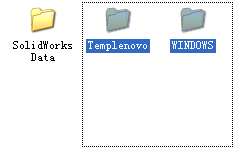


图1 选择文件夹 图2 改变图形大小

一、建立一个单文档工程，在Doc类中添加一个CrectTracker对象成员：

public:

CRectTracker m\_rectTracker;

二、在Doc的构造函数中初始化m\_rectTracker对象，包括大小和样式：

m\_rectTracker.m\_rect.SetRect(0,0,100,100);

m\_rectTracker.m\_nStyle=CRectTracker::resizeInside|CRectTracker::dottedLine;

三、为View类添加一个布尔形变量bDraw，初始化为FALSE，用来标识是否绘制蓝色矩形，蓝色表示被选中，黄色表示没有被选中。

四、设置鼠标样式：

BOOL CCRectTrackerDemoView::OnSetCursor(CWnd\* pWnd, UINT nHitTest, UINT message)

{

CCRectTrackerDemoDoc\* pDoc=GetDocument();

if (pWnd==this&&pDoc->m\_rectTracker.SetCursor(this,nHitTest))

{

return TRUE;

}

return CView::OnSetCursor(pWnd, nHitTest, message);

}

五、编写OnDraw函数：

void CCRectTrackerDemoView::OnDraw(CDC\* pDC)

{

CCRectTrackerDemoDoc\* pDoc = GetDocument();

ASSERT\_VALID(pDoc);

if (!pDoc)

return;

CBrush brush(RGB(255,255,0));

CBrush \*pOldBrush=pDC->SelectObject(&brush);

CRect rect;

pDoc->m\_rectTracker.GetTrueRect(&rect);

if (bDraw)

{

CBrush tempBrush(RGB(0,0,255));

pDC->SelectObject(&tempBrush);

pDC->Ellipse(rect);

pDC->SelectObject(&brush);

}

else

{

pDC->Ellipse(rect);

}

pDoc->m\_rectTracker.Draw(pDC);

pDC->SelectObject(pOldBrush);

}

注意：这里定义了一个CRect对象rect，m\_rectTracker.GetTrueRect(&rect)是将m\_rectTracker当前表示的矩形赋给rect，m\_rectTracker.Draw(pDC)函数是将橡皮筋的虚线框矩形绘制出来。

六、编写鼠标的OnLButtonDown函数：

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

{

CRectTracker temp;

int nInt;

nInt=GetDocument()->m\_rectTracker.HitTest(point);

if (nInt<0)

{

CRectTracker temp;

TRACE("Before TrackRubberBand\n");

temp.TrackRubberBand(this,point,TRUE);

TRACE("After TrackRubberBand\n");

temp.m\_rect.NormalizeRect();

if (temp.m\_rect.IntersectRect(temp.m\_rect,GetDocument()->m\_rectTracker.m\_rect))

bDraw=TRUE;

else

bDraw=FALSE;

Invalidate();

}

else

{

TRACE("Before Tracking...\n");

CRect tempRect;

GetDocument()->m\_rectTracker.GetTrueRect(&tempRect);

TRACE("m\_rectTracker:%d,%d,%d,%d",

tempRect.top,tempRect.bottom,tempRect.left,tempRect.right);

GetDocument()->m\_rectTracker.Track(this,point,TRUE);

TRACE("After Tracking...\n");

GetDocument()->m\_rectTracker.GetTrueRect(&tempRect);

TRACE("m\_rectTracker:%d,%d,%d,%d",

tempRect.top,tempRect.bottom,tempRect.left,tempRect.right);

Invalidate();

}

CView::OnLButtonDown(nFlags, point);

}

这里要解释几点：

1、为了测试鼠标点击的位置，可使用以下代码：

nInt=GetDocument()->m\_rectTracker.HitTest(point);

该代码使用m\_rectTracker的HitTest函数来测试鼠标位置，测试结果返回由nInt接收，如果点在矩形的外部，返回值为-1，在内部则为大于等于0的数，具体返回值的意义可参见MSDN。

2、当鼠标点击在矩形的外部时，定义一个CRectTracker temp对象来实时的显示绘制的虚线框矩形，这个绘制过程并不需要调用OnDraw函数，实际上，绘制虚线框是以下代码完成的：

temp.TrackRubberBand(this,point,TRUE);

第一个参数为窗口指针，第二个参数为起始点位置，第三个参数如果是TRUE则表示可以允许从反向绘制，为FALSE时表示不允许反向绘制。

当程序运行到该句代码时，会一直阻塞在这里，直到鼠标抬起，这可以在调试时观察TRACE语句的输出得知。

3、temp.m\_rect.NormalizeRect()使得绘制的temp矩形规范化。

4、判断临时的橡皮筋的矩形与图形的矩形是否有交集，如果有交集，表示图形被选中，被选中图形显示蓝色，若无交集表示没有被选中，图形显示黄色。

if (temp.m\_rect.IntersectRect(temp.m\_rect,GetDocument()->m\_rectTracker.m\_rect))

bDraw=TRUE;

else

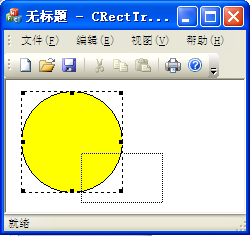
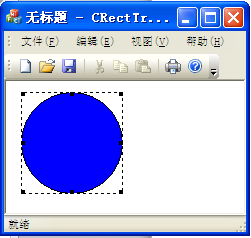
bDraw=FALSE;

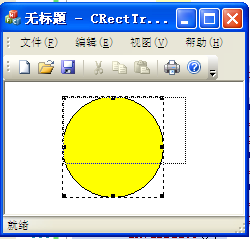
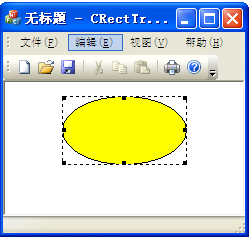
5、如果鼠标点击在图形内部，则用以下代码来实时跟踪m\_rectTracker所表示的矩形大小和位置：

GetDocument()->m\_rectTracker.Track(this,point,TRUE);

同样，程序运行到这时会被阻塞，直到鼠标按钮被释放，这也可以在调试时观察TRACE的输出语句可知。

6、这里并没有编写鼠标的OnMouseMove和OnLButtonUp函数，m\_rectTracker的实时跟踪矩形大小和位置，以及绘制虚线框的功能是由CRectTracker类自己完成的。

** **