Visual C#. NET 实现用户自定义图形编程方法

朱卫新
(连云港电子口岸信息发展有限公司,江苏 连云港 222042)

摘 要:信息系统终端用户可以自己绘制图形,以鼠标拖拽的方式实现已绘制图形的移动、缩放等功能,不仅会极大地方便用户使用,同时可以满足用户现场绘图的需求。对如何在信息管理系统中实现用户自定义图形进行分析和设计,并在Visual C#. NET 编程环境中使用GDI+等技术实现用户自定义图形,使终端用户能够动态创建可以用鼠标拖拽方式移动、缩放的图形,实现直线、矩形、文本的创建、移动、缩放功能,绘制的图形可以保存为XML文件,可以加载已保存XML文件。文中描述了自定义图形编程方法的设计和实现。

关键词:C#;GDI+;自定义图形
中图分类号:TP31 **文献标识码**:A **文章编号**:1673-629X(2012)04-0130-03

Realization of User Defined Graphics Programming Method under Visual C #. NET

ZHU Wei-xin
(Lianyungang E-Port Information Development Co. ,Ltd,Lianyungang 222042,China)

Abstract: Information system terminal user can draw graphics, realize mobile, zoom function of drawings in the way of mouse dragin, not only would be very convenient for user use, at the same time can meet the needs of the user drawing. Analyze and design how to realize the user defined graphics in the information management system, and in Visual C#. NET programming environment use GDI + technology to realize the user defined graphics which make end users could dynamically create the mobile, zoom graphics can be used to drag the way by mouse, and realize the straight line, rectangular, text creating, mobile, zoom function, rendering graphics can be saved for an XML file, can load an XML file has been kept. It describes the design and implementation of the custom graphics programming method.

Key words: C#;GDI+;custom graphics

0 引 言

DOTNET Framework 为操作图形提供了 GDI+ 应用程序编程接口,GDI+ 是 Windows 图形设备接口的高级实现,通过使用 GDI+,可以创建图形、绘制文本以及将图形图像作为对象操作,软件开发人员可以使用 GDI+ 在 Windows 窗体和控件上方便地呈现图形图像[1],但是随着信息系统应用的深入,由软件开发人员来绘制图形已经不能满足信息系统终端用户的需求,很多场景需要终端用户自己绘制图形,并可以对已绘制的图形做相应的修改属性操作。

文中正是描述了在 Visual C#. NET 编程环境中如何开发用户自定义图形的编程方法,使用户可以方便地绘制图形并且能够以鼠标拖拽的方式实现已绘制图形的移动、缩放等功能。

收稿日期:2011-09-06;**修回日期**:2011-12-11
基金项目:江苏省科技基础设施建设计划项目(BM2009835);连云港市科技创新基金项目(CK201039)
作者简介:朱卫新(1981-),男,安徽淮南田家庵人,工程师,硕士研究生,研究方向为港口物流信息化、EDI。

1 设计思路

(1)创建一个基础图形类,把用户要动态绘制的图形都抽象成类[2],并定义其属性和方法,用户绘制图形时实例化该类生成一个图形对象,保存绘制图形的属性;

(2)在窗体上创建一个工具条,并添加四个按钮,用来记录当前是要修改图形还是要绘制新图形;再创建八个 PictureBox,分别赋予不同的颜色,当用户选中已绘制的图形时,在图形的关键位置显示图片[3],用户可以拖动图片来移动或改变图形(在 PictureBox 的相关事件中实现);

(3)在窗体上创建一个动态数组,用来保存绘制的图形对象;

(4)在窗体的鼠标按下事件中判断当前是要绘制新图还是要修改图形,并记录当前鼠标的坐标;

(5)在窗体的鼠标移动事件中判断目前是绘制新

图形还是要选择已绘制的图形，如果是要修改图形，则根据移动鼠标的坐标遍历所有已绘制的图形（动态数组保存的对象）[4]，并判断是否有图形被选中，选中则在图片的关键位置显示图片，以便于改变图形属性；已选中图形或者绘制新图形，则先把新绘制或要修改的图形轨迹用窗体背景色画掉，再把所有窗体上的图形都重新绘制一遍[5]，以防止上个命令会擦除掉其他图形的像素，然后再绘制最新的图形；

（6）在窗体的鼠标释放事件中对于新绘制的图形则实例化一个图形对象[6]，用以保存新图形的属性，并把实例化的对象添加到动态数组中；对于修改的图形，则直接修改图形对象的属性；

（7）在 PictureBox 的鼠标按下、移动、释放事件中实现被选中图形的修改，逻辑依然是先用背景色画掉选中图形的轨迹，再把所有窗体上的图形都重新绘制一遍，以防止上个命令会擦除掉其他图形的像素，然后再绘制最新的图形[7]；

（8）在菜单的保存按钮中实现绘制图形保存成 XML 文件，打开按钮中实现加载已保存的 XML 文件，遍历动态数组保存成 XML 元素或遍历 XML 元素逐一加载到动态数组中[8]。

2　实现方法

（1）建立一个基础图形类，定义直线、矩形、文本等图形的属性和方法，以直线为例，主要代码片段如下：

```
public class Line//建立直线类,定义其属性和方法
{ //鼠标点到直线的水平或垂直距离小于等于指定值时为选中
public bool IsSelect(Point P)
{ if(Math. Abs(P1. X-P2. X)>=Math. Abs(P1. Y-P2. Y))
{if(P. X>Math. Max(P1. X,P2. X)||P. X<Math. Min(P1. X,P2. X))
return false;
int y=XToY(P. X);
if(Math. Abs(P. Y-y)<=m_Diff)
return true;
else
return false;}}
```

（2）在窗体类上定义结构体、枚举、变量和函数，代码片段如下：

```
private ArrayList m_Graph=new ArrayList();//用于存放绘制的图形
private void ReDraw()//根据 ArrayList 里存放的信息重新绘制所有已绘制的图形
{for(int i=m_Graph. Count-1;i>=0;i--)//把 ArrayList 里存放的信息逐一调出来
{objGraph objGh=(objGraph)(m_Graph[i]);
Graph. Line ln=(Graph. Line)(objGh. Graph);
g. DrawLine(pen,ln. P1,ln. P2);//把原来的直线用新的颜色画出来
}}
private int GetSelectIndex(int x,int y) //给一个坐标,判断有没有已绘制的图元被选中,并用不同颜色标识选中的图元
{ for(int i=0;i<m_Graph. Count;i++)
{objGraph objGh=(objGraph)(m_Graph[i])[9];
Graph. Line ln=(Graph. Line)(objGh. Graph); //仅已直线为例,矩形和文本的代码省略
if(ln. IsSelect(new Point(x,y)))
{g. DrawLine(pen,ln. P1,ln. P2);//图形被选中,并用绿色重新绘制
return i;}}}
```

（3）在窗体的鼠标按下事件中判断是要绘制还是修改图形，并给相关变量赋予不同的值，代码片段如下：

```
private void frmGraph_MouseDown(object sender, System. Windows. Forms. MouseEventArgs e)
{if(e. Button==MouseButtons. Left)
{switch(pushBtName)
{case "line":
blSave=false;
blDrawing=true;
break;
case "sel"://选择图形对象
idxSelected=GetSelectIndex(e. X,e. Y);
ReDraw();
if(idxSelected>=0&&idxSelected<m_Graph. Count)
{blMoving=true;
blSave=false;}
break;
default:return;}
curPosX=e. X; //修改变量值,并记录鼠标坐标
curPosY=e. Y;
lastX=e. X;
lastY=e. Y;}}
```

（4）在窗体的鼠标移动事件中绘制图形，代码片段如下：

```
private void frmGraph_MouseMove(object sender, System. Windows. Forms. MouseEventArgs e)
{if(blDrawing) //绘制新图元
{Graphics g=this. CreateGraphics()[10];
Pen pen=new Pen(Color. Black,1);
switch(pushBtName)
{case "line":
if(curPosX!=lastX||curPosY!=lastY)//在鼠标移动过程中把现在画的直线的轨迹用背景色画掉
g. DrawLine(new Pen(this. BackColor,1),curPosX,curPosY,lastX,lastY);
```

```
//为防止清除现画图形的轨迹命令同时也擦掉其他图形,重新画一次以前的所有图形
ReDraw();
g.DrawLine(pen,curPosX,curPosY,e.X,e.Y);//画最新的图形
break;}
lastX=e.X;
lastY=e.Y;}
if(blMoving) //修改已绘制的图元
{if(idxSelected>=0&&idxSelected<m_Graph.Count)
{Graphics g=this.CreateGraphics();
Pen pen=new Pen(Color.Black,1);
objGraph objGh=(objGraph)(m_Graph[idxSelected]);
switch(objGh.gType)
{case objType.LINE: //代码片段以直线为例
Graph.Line ln=(Graph.Line)(objGh.Graph);
if(curPosX!=lastX||curPosY!=lastY)
//在鼠标移动过程中把原图元用背景色画掉
g.DrawLine(new Pen(this.BackColor,1),newPoint(ln.P1.X+lastX-curPosX,ln.P1.Y+lastY-curPosY),new Point(ln.P2.X+lastX-curPosX,ln.P2.Y+lastY-curPosY));
ReDraw();//为防止清除原图元的操作同时也擦掉其他图形上的部分像素,重新画一次以前的所有图形
//画最新的图形
g.DrawLine(pen,new Point(ln.P1.X+e.X-curPosX,ln.P1.Y+e.Y-curPosY),new Point(ln.P2.X+e.X-curPosX,ln.P2.Y+e.Y-curPosY));
break;}
lastX=e.X;
lastY=e.Y;
}}}
```

(5)在窗体的鼠标释放事件中添加新绘制的图元或保存修改过的图元属性到动态数组中,代码片段如下:

```
private void frmGraph_MouseUp(object sender, System.Windows.Forms.MouseEventArgs e)
{if(blDrawing) //保存新绘制的图元
{blDrawing=false;
objGraph objGh;
if(e.X!=curPosX||e.Y!=curPosY)
{switch(pushBtName)
{case "line" ://保存直线的代码,其他图元代码略
Graph.Line ln=new Graph_demo.Graph.Line();
ln.P1=new Point(curPosX,curPosY)[11];
ln.P2=new Point(e.X,e.Y);
objGh=new objGraph();
objGh.gType=objType.LINE;
objGh.Graph=ln;
m_Graph.Add(objGh);//把新画的图形加到 arraylist 中
break;}
ReDraw();}}
if(blMoving) //保存修改的图元属性
{blMoving=false;
if(idxSelected>=0&&idxSelected<m_Graph.Count)
{Graphics g=this.CreateGraphics();
objGraph objGh=(objGraph)(m_Graph[idxSelected]);
switch(objGh.gType)
{case objType.LINE: //保存直线的代码,其他图元代码略
Graph.Line ln=(Graph.Line)(objGh.Graph);
g.DrawLine(new Pen(this.BackColor),ln.P1,ln.P2)[12];//清除原图元
ln.P1=new Point(ln.P1.X+e.X-curPosX,ln.P1.Y+e.Y-curPosY); //更新坐标
ln.P2=new Point(ln.P2.X+e.X-curPosX,ln.P2.Y+e.Y-curPosY);
break;}
ReDraw();//重新绘制所有已绘制的图元
}}}
```

(6)在窗体上创建了八个 PictureBox 控件,当已绘制的图元被选中时显示不同的图片(控件),拖动图片(控件)即可改变选中图元的位置、大小等,主要代码片段如下:

```
//修改直线的代码片段,直线的两端用小图片显示
g.DrawLine(pencls,new Point(ln.P1.X+ddx,ln.P1.Y+ddy),ln.P2);
ReDraw();
g.DrawLine(pen,new Point(ln.P1.X+dx,ln.P1.Y+dy),ln.P2);
```

(7)把绘制的图形保存到 XML 文件,或加载 XML 文件到窗体,主要代码片段如下:

```
//把绘制的图形保存到 XML 文件,保存直线,矩形、文本代码省略
Graph.Line ln=(Graph.Line)(objGh.Graph);
eleMsg.SetAttribute("x1",ln.P1.X.ToString());
nodeMsg.AppendChild(eleMsg);
```

3 结束语

通过演示程序,实现了在 Visual C#.NET 编程环境中如何应用 GDI+使用户可以自定义地动态绘制图形,并且可以使用鼠标拖拽实现图元的移动、缩放等功能,绘制的图形可以保存为 XML 文件,也可以加载已保存 XML 文件到窗体图形。作者已把文中的实现方法应用到集装箱堆场管理信息系统中,用户使用起来形象、直观,得到了用户的一致好评。通过文中示例,希望能增进大家对文中涉及到的相关技术的了解,为以后编程实现中遇到类似的需求提供实践基础。

参考文献:

[1] 刘忠艳,周 波,车向前.一种高效的图像匹配算法[J].计

(下转第 136 页)

在一定程度降低时间复杂度，但一个好的预计算策略更能很好地提高算法的精度。网络状态的时变性和控制信息的滞后性，决定着寻找一个好的预计算策略，再结合并行机制，能够很好地解决网络状态的时变性和控制信息的滞后性。

参考文献：

[1] Dijkstra E W. A note on two problems in connexion with graphs[J]. Numerische Mathematik,1959,1(1):269-271.

[2] Martins E. On a multicriteria shortest path problem[J]. European Journal of Operational Research,1984,16:236-245.

[3] Gandibleux X,Beugnies F,Randriamasy S. Martins' algorithm revisited for multi-objective shortest path problems with a maxmin cost function[J]. 4OR-A Quarterly Journal of Operations Research,2006,4(1):47-59.

[4] Vincke P. Problémes multicritéres[J]. Cahiers du Centre d'Etudes de Recherche Opérationelle,1974,16:425-439.

[5] Skriver A,Andersen K. A label correcting approach for solving bicriterion shortest-path problems[J]. Computers & Operations Research,2000,27(6):507-524.

[6] Raith A,Ehrgott M. A comparison of solution strategies for biobjective shortest path problems[J]. Computers & Operations Research,2009,36(4):1299-1331.

[7] Paixão J M,Santos J L. Labelling methods for the general case of the multi-objective shortest path problem - a computational study[D]. Coimbra:Universidade de Coimbra,2007.

[8] Clímaco J C N,Martins E Q V. A bicriterion shortest path algorithm[J]. European Journal of Operational Research,1982,11(4):399-404.

[9] Martins E Q,Paixão J M,Rosa M S,et al. Ranking multiobjective shortest paths[D]. Coimbra:Universidade de Coimbra,2007.

[10] Paixão J M,Santos J L. A new ranking path algorithm for the multi-objective shortest path problem[D]. Coimbra:Universidade de Coimbra,2008.

[11] Mote J,Murthy I,Olson D L. A parametric approach to solving bicriterion shortest path problems[J]. European Journal of Operational Research,1991,53(1):81-92.

[12] Sahni S. General techniques for combinatorial approximation[J]. Operations Research,1977,25(6):920-936.

[13] Hansen P. Multiple criteria decision making:theory and application[C]//Lecture Notes in Economics and Mathematical Systems. Berlin:Springer,1980:109-127.

[14] Warburton A. Approximation of Pareto optima in multiple-objective,shortest-path problems[J]. Operations Research,1987,35(1):70-79.

[15] Hassin R. Approximation schemes for the restricted shortest path problem[J]. Mathematics of Operations Research,1992,17(1):36-42.

[16] Chen S,Nahrstedt K. On finding multi-constrained paths[C]//IEEE International Conference on Communications(ICC). [s.l.]:[s.n.],1998:874-879.

[17] Lorenz D,Raz D. A simple efficient approximation scheme for the restricted shortest path problem[J]. Operations Research Letters,2001,28(5):213-219.

[18] Yuan X. Heuristic algorithms for multi-constrained quality of service routing[J]. IEEE/ACM Transactions on Networking,2002,10(2):244-256.

[19] Song M,Sahni S. Approximation algorithms for multiconstrained quality-of-service routing[J]. IEEE Transactions on Computers,2006,55(5):603-617.

[20] Xue G,Sen A,Zhang W,et al. Finding a path subject to many additive QoS constraints[J]. IEEE/ACM Transactions on Networking,2007,15(1):201-211.

[21] Tsaggouris G,Zaroliagis C. Multiobjective optimization:improved FPTAS for shortest paths and non-linear objectives with applications[J]. Theory of Computing Systems,2009(1):162-186.

(上接第 132 页)

算机技术与发展,2009,19(4):46-47.

[2] 魏 欣,蒋华伟. 基于纹理和结构的图像修复算法研究[J]. 计算机技术与发展,2010,20(9):91-92.

[3] 高 飞,侯瑞春,周志明. Web 页面缓存技术在业务系统中的应用[J]. 计算机技术与发展,2010,20(1):210-211.

[4] 蔡丽欢,廖英豪,郭东辉. 图像拼接方法及其关键技术研究[J]. 计算机技术与发展,2008,18(3):1-4.

[5] 衣文文,杨彬彬,胡彦磊,等. 一种基于外形区域的图像配准方法与实现[J]. 计算机技术与发展,2008,18(4):1-4.

[6] Nagel C,Evjen B,Glynn J. C#高级编程[M]. 北京:清华大学出版社,2008.

[7] 杨建昌. GDI+高级编程[M]. 北京:清华大学出版社,2009.

[8] 埃斯波西托,萨尔塔列洛. Microsoft . NET 企业级应用架构设计[M]. 陈黎夫译. 北京:人民邮电出版社,2010.

[9] 况 旭,刘 波. XML 的面向对象语言特性[J]. 计算机技术与发展,2010,20(1):55-56.

[10] Segaran T. Programming Collective Intelligence[M]. [s.l.]:O'Reilly,2009.

[11] Reeves W T. Particle systems-a technique for modeling a class of fuzzy objects[J]. Computer Graphics,1983,17(3):359-376.

[12] Luebke D,Reddy M,Cohen J D,et al. Level of Detail for 3D Graphics[M]. USA:Morgan Kaufmann Publishers,2002.