Министерство образования и науки РФ

ФГБОУ ВПО «Тамбовский государственный технический университет»

Кафедра «»

Лабораторная работа №4

по дисциплине «Объектно-ориентированное программирование»

«Векторный редактор»

Выполнил: студент гр. qwinmen

Проверил:.

Тамбов, 20

***Цели и задачи****.*

Разработать простейший векторный графический редактор.

***Решение задачи****.*

Реализация редактора возможна по принципам графического из второй лабороторной работы, однако необходимо предусмотреть отдельный обьект для хранения будущих фигур, обеспечить некий доступ к каждой конкретно выделенной фигуре, а также возможность работать над полями со свойствами фигуры.



***Код***

#region Base element Ele and its derived class

//Базовый элемент

[Serializable]

public abstract class Ele: Object

{

static public Single dpix;

static public Single dpiy;

protected bool IamGroup = false;

protected bool rot; // Can Rotate

//Start point

protected int X;

protected int Y;

//End point

protected int X1;

protected int Y1;

protected OnGroupResize \_onGroupXRes = OnGroupResize.Resize;

protected OnGroupResize \_onGroupX1Res = OnGroupResize.Resize;

protected OnGroupResize \_onGroupYRes = OnGroupResize.Resize;

protected LineCap start;

protected LineCap end;

protected DashStyle dashstyle;

protected int \_rotation;//угол поворота обьекта//для {get;set}

private Color \_penColor;

private float \_penWidth;//1f

private Color \_fillColor;

private bool \_filled;

private bool \_showBorder;

private DashStyle \_dashStyle;

private int \_alpha;

private bool \_useGradientLine = false;

private Color \_endColor=Color.White;

private int \_endalpha = 255;

private int \_gradientLen = 0;

private int \_gradientAngle = 0;

private float \_endColorPos = 1f;

public bool Selected;

public bool Deleted;

public Ele undoEle;

public Ele()

{//Конструктор класса

penColor=Color.Black;

penWidth= 1f;

fillColor=Color.Black;

filled=false;

showBorder = true;

this.dashstyle = DashStyle.Solid;

this.alpha = 255;

}

#region virtual metods

/// <summary>

/// Draw this shape to a graphic ogj.

/// </summary>

public virtual void Draw(Graphics g,int dx,int dy)

{ }

/// <summary>

/// Add this shape to a graphic path.

/// </summary>

public void AddGraphPath(GraphicsPath gp, int dx, int dy)

{

GraphicsPath tmpGp = new GraphicsPath();

AddGp(tmpGp, dx, dy);// AddGp is defined in derived classes

Matrix translateMatrix = new Matrix();

translateMatrix.RotateAt(this.\_rotation, new PointF((this.X + dx + (int)(this.X1 - this.X) / 2) , (this.Y + dy + (int)(this.Y1 - this.Y) / 2) ));

tmpGp.Transform(translateMatrix);

gp.AddPath(tmpGp, true);

}

#endregion

/// <summary>

/// To fill a shape with parallel lines

/// </summary>

protected void FillWithLines(Graphics g, int dx, int dy, float zoom, GraphicsPath myPath, float gridSize, float gridRot)

{

GraphicsState gs = g.Save();//store previos trasformation

g.SetClip(myPath, CombineMode.Intersect);

Matrix mx = g.Transform; // get previous trasformation

PointF p = new PointF(zoom \* (this.X + dx + (this.X1 - this.X) / 2), zoom \* (this.Y + dy + (this.Y1 - this.Y) / 2));

if (this.\_rotation > 0)

mx.RotateAt(this.\_rotation, p, MatrixOrder.Append); //add a trasformation

mx.RotateAt(gridRot, p, MatrixOrder.Append); //add a trasformation

g.Transform = mx;

int max = System.Math.Max(this.Width, this.Height);

System.Drawing.Pen linePen = new System.Drawing.Pen(System.Drawing.Color.Gray);

//linePen.DashStyle = DashStyle.Dash;

int nY = (int)(max \* 3 / (gridSize));

for (int i = 0; i <= nY; i++)

{

g.DrawLine(linePen, (this.X - max + dx) \* zoom, (this.Y - max + dy + i \* gridSize) \* zoom, (this.X + dx + max \* 2) \* zoom, (this.Y - max + dy + i \* gridSize) \* zoom);

}

linePen.Dispose();

g.Restore(gs);

//g.ResetClip();

}

/// <summary>

/// Rotate

/// </summary>

public virtual void Rotate(float x, float y)

{

float tmp = \_rotate(x, y);

this.\_rotation = (int)tmp;

}

/// <summary>

/// Return a point obtained rotating p by RotAng respect 0,0

/// </summary>

protected PointF rotatePoint(PointF p, int RotAng)

{

double RotAngF = RotAng \* Math.PI/180;

double SinVal = Math.Sin(RotAngF);

double CosVal = Math.Cos(RotAngF);

float Nx = (float)(p.X \* CosVal - p.Y \* SinVal);

float Ny = (float)(p.Y \* CosVal + p.X \* SinVal);

return new PointF(Nx, Ny);

}

/// <summary>

/// Gets a rotation angle from a vertical line from the center of the shape and a line

/// from the center to the point (x,y)

/// </summary>

protected float \_rotate(float x, float y)

{

//

Point c = new Point((int)(this.X + (this.X1 - this.X) / 2), (int)(this.Y+(this.Y1 - this.Y) / 2));

float dx = x - c.X;

float dy = y - c.Y;

float b= 0f;

float alpha= 0f;

float f= 0f;

if ((dx>0)&(dy>0))

{

b= 90;

alpha= (float)Math.Abs((Math.Atan((double)(dy/dx)) \* (180/Math.PI)));

}

else

if ((dx<=0)&(dy>=0))

{

b= 180;

if (dy > 0)

{

alpha = (float)Math.Abs((Math.Atan((double)(dx / dy)) \* (180 / Math.PI)));

}

else if (dy == 0)

{

b = 270;

}

}

else

if ((dx<0)&(dy<0))

{

b= 270;

alpha= (float)Math.Abs((Math.Atan((double)(dy/dx)) \* (180/Math.PI)));

}

else

{

b= 0;

alpha= (float)Math.Abs((Math.Atan((double)(dx/dy)) \* (180/Math.PI)));

}

f = (b+ alpha);

return f;

}

private float getDimX()

{

return (float)(System.Math.Sqrt( System.Math.Pow(this.Width, 2) + System.Math.Pow(this.Height, 2)) - this.Width )/2;

}

private float getDimY()

{

return (float)(System.Math.Sqrt(System.Math.Pow(this.Width, 2) + System.Math.Pow(this.Height, 2)) - this.Height) / 2;

}

/// <summary>

/// gets a brush from the properties of the shape

/// </summary>

protected Brush getBrush(int dx,int dy)

{

if (this.filled)

{

if (this.UseGradientLineColor)

{

float wid;

float hei;

if (this.GradientLen > 0)

{

wid = this.GradientLen;

hei = this.GradientLen;

}

else

{

wid = ((this.X1 - this.X) + 2 \* getDimX());

hei = ((this.Y1 - this.Y) + 2 \* getDimY());

}

LinearGradientBrush br=new LinearGradientBrush(

new RectangleF((this.X - getDimX() + dx) , (this.Y - getDimY() + dy), wid, hei)

, this.Trasparency(this.fillColor, this.alpha)

, this.Trasparency(this.EndColor, this.EndAlpha)

, this.GradientAngle

,true);

br.SetBlendTriangularShape(this.EndColorPosition, 0.95f);

br.WrapMode = WrapMode.TileFlipXY;

return br;

}

else

{

return new SolidBrush(this.Trasparency(this.fillColor, this.alpha));

}

}

else

{

return null;

}

}

/// <summary>

/// 2 points distance

/// </summary>

protected int Dist(int x, int y, int x1, int y1)

{

return (int)System.Math.Sqrt(System.Math.Pow((x - x1),2) + System.Math.Pow((y - y1),2) );

}

/// <summary>

/// Make a color darker or lighter

/// </summary>

protected Color dark(Color c, int v, int a)

{

int r = c.R;

r = r - v;

if (r < 0)

r = 0;

if (r >255)

r = 255;

int green = c.G;

green = green - v;

if (green < 0)

green = 0;

if (green > 255)

green = 255;

int b = c.B;

b = b - v;

if (b < 0)

b = 0;

if (b > 255)

b = 255;

if (a > 255)

a = 255;

if (a <0)

a = 0;

return Color.FromArgb(a,r, green, b);

}

/// <summary>

/// Make a color Tresparent/Solid

/// </summary>

protected Color Trasparency(Color c, int v)

{

if (v < 0)

v = 0;

if (v > 255)

v = 255;

return Color.FromArgb(v, c);

}

/// <summary>

/// true if the shape contains the point x,y

/// </summary>

public virtual bool contains(int x, int y)

{

if (sonoUnaLinea)

{

int appo = Dist(x, y, this.X, this.Y) + Dist(x, y, this.X1, this.Y1);

int appo1 = Dist(this.X1, this.Y1, this.X, this.Y) + 7;

return appo < appo1;

}

else

{

return new Rectangle(this.X, this.Y, this.X1 - this.X, this.Y1 - this.Y).Contains(x, y);

}

}

/// <summary>

/// Двинуть обьект на x,y

/// </summary>

public virtual void move(int x, int y)

{

this.X = this.startX - x;

this.Y = this.startY - y;

this.X1 = this.startX1 - x;

this.Y1 = this.startY1 - y;

}

/// <summary>

/// Linea ( estende Ele )

/// </summary>

[Serializable]

public class Linea : Ele

{

private LineCap \_starCap;

private LineCap \_endCap;

public Linea(int x, int y, int x1, int y1)

{

this.X = x;

this.Y = y;

this.X1 = x1;

this.Y1 = y1;

this.sonoUnaLinea = true;

this.Selected = true;

this.endMoveRedim();

this.rot = false; //can rotate?

}

public override void Select()

{

this.undoEle = this.Copy();

}

public override void AddGp(GraphicsPath gp,int dx , int dy )

{

gp.AddLine((this.getX() + dx) , (this.getY() + dy) , (this.getX1() + dx) , (this.getY1() + dy) );

}

public override void Draw(Graphics g,int dx,int dy)

{

System.Drawing.Pen myPen = new System.Drawing.Pen(this.penColor);

myPen.DashStyle = this.dashStyle;

myPen.Color = this.Trasparency(this.penColor, this.alpha);

if (this.Selected)

{

myPen.Color = Color.Red;

myPen.Color = this.Trasparency(myPen.Color, 120);

myPen.Width = myPen.Width + 1;

g.DrawEllipse(myPen,(this.X + dx), (this.Y + dy),3,3);

}

if (this.X==this.X1 && this.Y==this.Y1)

g.DrawEllipse(myPen, (this.X + dx), (this.Y + dy), 3, 3);

else

g.DrawLine(myPen, (this.X + dx) , (this.Y + dy) , (this.X1 + dx), (this.Y1 + dy) );

myPen.Dispose();

}

}

/// <summary>

/// Arc

/// </summary>

[Serializable]

public class Arc : Ele

{

private int \_startAng;

private int \_lenAng;

private LineCap \_starCap;

private LineCap \_endCap;

public Arc(int x, int y, int x1, int y1)

{

this.X = x;

this.Y = y;

this.X1 = x1;

this.Y1 = y1;

this.Selected = true;

this.endMoveRedim();

this.StartAng = 0;

this.LenAng = 90;

this.startCap = LineCap.Custom;

}

[CategoryAttribute("1"), DescriptionAttribute("Rectangle")]

public string ObjectType

{

get

{

return "Arc";

}

}

[CategoryAttribute("Line Appearance"), DescriptionAttribute("Line Start Cap")]

public LineCap startCap

{

get

{

return \_starCap;

}

set

{

\_starCap = value;

}

}

//Начальная точка сектора

public int StartAng

{

get

{

return \_startAng;

}

set

{

\_startAng = value;

}

}

//Отробразить сектор длинной \_lenAng

public int LenAng

{

get

{

return \_lenAng;

}

set

{

\_lenAng = value;

}

}

public override void Select()

{

this.undoEle = this.Copy();

}

public override void AddGp(GraphicsPath gp, int dx, int dy )

{

gp.AddArc((this.X + dx) , (this.Y + dy) , (this.X1 - this.X) , (this.Y1 - this.Y) , this.StartAng, this.LenAng);

}

public override void Draw(Graphics g,int dx,int dy)

{

//System.Drawing.SolidBrush myBrush = new System.Drawing.SolidBrush(this.fillColor);

Brush myBrush = getBrush(dx, dy);

//myBrush.Color = this.Trasparency(this.fillColor, this.alpha);

System.Drawing.Pen myPen = new System.Drawing.Pen(this.penColor);

myPen.DashStyle = this.dashStyle;

myPen.StartCap = this.startCap;

//myBrush.Color = this.Trasparency(this.fillColor, this.alpha);

if (this.Selected)

{

System.Drawing.Pen myPen1 = new System.Drawing.Pen(this.penColor);

myPen1.Width = 0.5f;

myPen1.DashStyle = DashStyle.Dot;

g.DrawEllipse(myPen1, (this.X + dx), (this.Y + dy), (this.X1 - this.X), (this.Y1 - this.Y));

myPen1.Dispose();

//myBrush.Color = this.dark(this.fillColor, 5, this.alpha);

myPen.Color = Color.Red;

myPen.Color = this.Trasparency(myPen.Color, 120);

myPen.Width = myPen.Width + 1;

}

// Create a path and add the object.

GraphicsPath myPath = new GraphicsPath();

myPath.AddArc((this.X + dx), (this.Y + dy), (this.X1 - this.X), (this.Y1 - this.Y), this.StartAng, this.LenAng);

//Matrix translateMatrix = new Matrix();

//translateMatrix.RotateAt(this.Rotation, new Point(this.X + (int)(this.X1 - this.X) / 2, this.Y + (int)(this.Y1 - this.Y) / 2));

//myPath.Transform(translateMatrix);

// Draw the transformed ellipse to the screen.

if (this.filled)

{

g.FillPath(myBrush, myPath);

if (this.showBorder)

g.DrawPath(myPen, myPath);

}

else

g.DrawPath(myPen, myPath);

myPath.Dispose();

myPen.Dispose();

if (myBrush != null)

myBrush.Dispose();

}

}

public override void move(int x, int y)

{

foreach (Ele e in objs)

{

e.move(x, y);

}

this.X = this.startX - x;

this.Y = this.startY - y;

this.X1 = this.startX1 - x;

this.Y1 = this.startY1 - y;

}

public override Ele Copy()

{

//Copy chils

ArrayList l1 = new ArrayList();

foreach (Ele e in this.objs)

{

Ele e1 = e.Copy();

l1.Add(e1);

}

Group newE = new Group(l1);

newE.Rotation = this.Rotation;

newE.\_grapPath = this.\_grapPath;

newE.gprZoomX = this.gprZoomX;

newE.gprZoomY = this.gprZoomY;

newE.IamGroup = this.IamGroup;

newE.\_name = this.Name + "\_" + Group.Ngrp.ToString();

newE.OnGrpXRes = this.OnGrpXRes;

newE.OnGrpX1Res = this.OnGrpX1Res;

newE.OnGrpYRes = this.OnGrpYRes;

newE.OnGrpY1Res = this.OnGrpY1Res;

newE.GroupDisplay = this.GroupDisplay;

if (newE.\_grapPath)

{

newE.penColor = this.penColor;

newE.fillColor = this.fillColor;

newE.filled = this.filled;

newE.dashStyle = this.dashStyle;

newE.alpha = this.alpha;

newE.sonoUnaLinea = this.sonoUnaLinea;

newE.Rotation = this.Rotation;

newE.showBorder = this.showBorder;

newE.UseGradientLineColor = this.UseGradientLineColor;

newE.GradientAngle =this.GradientAngle;

newE.GradientLen = this.GradientLen;

newE.EndAlpha = this.EndAlpha;

newE.EndColor = this.EndColor;

newE.EndColorPosition = this.EndColorPosition;

}

return newE;

}

[Serializable]

public class PointSet : Ele

{// ака btnPen

public ArrayList points;//хранит инфо о точках

private bool \_curved = false;

private bool \_closed = false;

public PointSet(int x, int y, int x1, int y1, ArrayList a)

{

this.X = x;

this.Y = y;

this.X1 = x1;

this.Y1 = y1;

this.Selected = true;

//

this.points = a;

this.setupSize();

//

this.endMoveRedim();

this.Rotation = 0;

this.rot = true; //can rotate?

}

[CategoryAttribute("Polygon"), DescriptionAttribute("Points")]

public ArrayList Points

{

get

{

return this.points;

}

set

{

points = value;

}

}

[CategoryAttribute("1"), DescriptionAttribute("Rectangle")]

public string ObjectType

{

get

{

return "PointSet";

}

}

[DescriptionAttribute("Rotation angle")]

public int Rotation

{

get

{

return \_rotation;

}

set

{

\_rotation = value;

}

}

private void rePos()

{

if (points != null)

{

int minNegativeX = 0;

int minNegativeY = 0;

foreach (PointWr p in points)

{

minNegativeX = p.X;

minNegativeY = p.Y;

break;

}

foreach (PointWr p in points)

{

if (p.X < minNegativeX)

minNegativeX = p.X;

if (p.Y < minNegativeY)

minNegativeY = p.Y;

}

//if (minNegativeX < 0 | minNegativeY < 0)

//{

foreach (PointWr p in points)

{

p.X = p.X - minNegativeX; ;

p.Y = p.Y - minNegativeY; ;

}

//}

this.X = this.X + minNegativeX;

this.Y = this.Y + minNegativeY;

}

}

public ArrayList getRealPosPoints()

{

ArrayList a = new ArrayList();

foreach (PointWr p in points)

{

a.Add(new PointWr(p.X+this.X,p.Y+this.Y));

}

return a;

}

public void setupSize()

{

if (this.points != null)

{

int maxX = 0;

int maxY = 0;

foreach (PointWr p in this.points)

{

if (p.X > maxX)

maxX = p.X;

if (p.Y > maxY)

maxY = p.Y;

}

this.Y1 = this.Y + maxY;

this.X1 = this.X + maxX;

this.rePos();//!

}

}

public override void CommitRotate(float x, float y)

{

//base.CommitRotate(x, y);

//this.Rotation

if (this.Rotation > 0)

{

//CENTER POINT

float midX, midY = 0;

midX = (this.X1 - this.X) / 2;

midY = (this.Y1 - this.Y) / 2;

foreach (PointWr p in points)

{

p.RotateAt(midX, midY, this.Rotation);

}

this.Rotation = 0;

}

}

public override Ele Copy()

{

ArrayList aa = new ArrayList();

if (this.points != null)

{

foreach (PointWr p in points)

{

aa.Add(p.copy());

}

}

PointSet newE = new PointSet(this.X, this.Y, this.X1, this.Y1,aa);

newE.penColor = this.penColor;

newE.fillColor = this.fillColor;

newE.filled = this.filled;

newE.dashStyle = this.dashStyle;

newE.alpha = this.alpha;

newE.sonoUnaLinea = this.sonoUnaLinea;

newE.Rotation = this.Rotation;

newE.showBorder = this.showBorder;

newE.OnGrpXRes = this.OnGrpXRes;

newE.OnGrpX1Res = this.OnGrpX1Res;

newE.OnGrpYRes = this.OnGrpYRes;

newE.OnGrpY1Res = this.OnGrpY1Res;

// newE.copyGradprop(this);

newE.Closed = this.Closed;

newE.Curved = this.Curved;

return newE;

}

public override void CopyFrom(Ele ele)

{

this.copyStdProp(ele, this);

this.Rotation = ((PointSet)ele).Rotation;

this.Curved = ((PointSet)ele).Curved;

this.Closed = ((PointSet)ele).Closed;

}

public override void Select()

{

this.undoEle = this.Copy();

}

public override void AddGp(GraphicsPath gp, int dx, int dy)

{

// To ARRAY

PointF[] myArr = new PointF[this.points.Count];

int i = 0;

foreach (PointWr p in this.points)

{

myArr[i++] = new PointF((p.X + this.X + dx), (p.Y + this.Y + dy));// p.point;

}

if (i < 2)

gp.AddLines(myArr);

else

if (this.Curved)

gp.AddCurve(myArr);

else

gp.AddPolygon(myArr);

}

public override void Draw(Graphics g, int dx, int dy)

{

//System.Drawing.SolidBrush myBrush = new System.Drawing.SolidBrush(this.fillColor);

Brush myBrush = getBrush(dx, dy);

//myBrush.Color = this.Trasparency(this.fillColor, this.alpha);

System.Drawing.Pen myPen = new System.Drawing.Pen(this.penColor);

myPen.DashStyle = this.dashStyle;

if (this.Selected)

{

//myBrush.Color = this.dark(this.fillColor, 5,this.alpha);

myPen.Color = Color.Red;

myPen.Color = this.Trasparency(myPen.Color, 120);

myPen.Width = myPen.Width + 1;

}

// Create a path and add the object.

GraphicsPath myPath = new GraphicsPath();

// To ARRAY

PointF[] myArr = new PointF[this.points.Count];

int i=0;

foreach (PointWr p in this.points)

{

myArr[i++] = new PointF((p.X + this.X +dx) , (p.Y + this.Y+dy) );// p.point;

//if (p.selected)

// g.FillEllipse(new SolidBrush(Color.Green), (p.X + this.X + dx-2) \* zoom, (p.Y + this.Y + dy-2) \* zoom, 5\*zoom, 5\*zoom);

}

if (myArr.Length < 3 | !this.Curved)

{

if (Closed & myArr.Length >= 3)

myPath.AddPolygon(myArr);

else

myPath.AddLines(myArr);

}

else

{

if (Closed)

myPath.AddClosedCurve(myArr);

else

myPath.AddCurve(myArr);

//myPath.AddBeziers(myArr);

}

//myPath.AddRectangle(new RectangleF((this.X + dx) \* zoom, (this.Y + dy) \* zoom, (this.X1 - this.X) \* zoom, (this.Y1 - this.Y) \* zoom));

Matrix translateMatrix = new Matrix();

translateMatrix.RotateAt(this.Rotation, new PointF((this.X + dx + (int)(this.X1 - this.X) / 2) , (this.Y + dy + (int)(this.Y1 - this.Y) / 2) ));

myPath.Transform(translateMatrix);

// Draw the transformed obj to the screen.

if (this.filled)

{

g.FillPath(myBrush, myPath);

if (this.showBorder)

g.DrawPath(myPen, myPath);

}

else

g.DrawPath(myPen, myPath);

myPath.Dispose();

myPen.Dispose();

if (myBrush != null)

myBrush.Dispose();

}

#endregion

}

/// <summary>

/// Rectangle ( extends Ele )

/// </summary>

[Serializable]

public class Rect : Ele

{//ака Прямоугольник

public Rect(int x, int y,int x1, int y1)

{

this.X = x;

this.Y = y;

this.X1 = x1;

this.Y1 = y1;

this.Selected = true;

this.endMoveRedim();

this.Rotation = 0;//Начальный поворот

this.rot = true; //может поворачиваться?

}

public int Rotation

{//поворот прямоугольника

get

{

return \_rotation;

}

set

{

\_rotation = value;

}

}

public override Ele Copy()

{//переназначить метод копирайта для прямоугольников

Rect newE = new Rect(this.X, this.Y, this.X1, this.Y1);

newE.penColor = this.penColor;

newE.fillColor = this.fillColor;

newE.filled = this.filled;

newE.Rotation = this.Rotation;

return newE;

}

public override void Draw(Graphics g,int dx , int dy)

{

Brush myBrush = getBrush(dx,dy);

System.Drawing.Pen myPen = new System.Drawing.Pen(this.penColor);

if (this.Selected)

{//рамка при выделеном обьекте прямоугольник

myPen.Color = Color.Red;

myPen.Color = this.Trasparency(myPen.Color, 120);

myPen.Width = myPen.Width + 1;

}

// создать рамку и добавить

GraphicsPath myPath = new GraphicsPath();//рамка-прямоугольник

myPath.AddRectangle( new RectangleF((this.X + dx) , (this.Y + dy) , (this.X1 - this.X) , (this.Y1 - this.Y) ));

Matrix translateMatrix = new Matrix();//System.Drawing.Drawing2D

//RotateAt(угол поворота, новые X центр, новые Y центра);

translateMatrix.RotateAt(this.Rotation, new PointF((this.X + dx + (int)(this.X1 - this.X) / 2) , (this.Y + dy + (int)(this.Y1 - this.Y) / 2) ));

myPath.Transform(translateMatrix);

// Отобразить на холсте

if (this.filled)

{

g.FillPath(myBrush, myPath);

if (this.showBorder)

g.DrawPath(myPen, myPath);

}

else

g.DrawPath(myPen, myPath);

myPath.Dispose();

myPen.Dispose();

if (myBrush!=null)

myBrush.Dispose();

}

}

/// <summary>

/// Ellipse

/// </summary>

[Serializable]

public class Ell : Ele

{

public Ell(int x, int y, int x1, int y1)

{

this.X = x;

this.Y = y;

this.X1 = x1;

this.Y1 = y1;

this.Selected = true;

this.endMoveRedim();

this.Rotation = 0;

this.rot = true; //can rotate

}

public override Ele Copy()

{

Ell newE = new Ell(this.X, this.Y, this.X1, this.Y1);

newE.penColor = this.penColor;

newE.fillColor = this.fillColor;

newE.filled = this.filled;

newE.Rotation = this.Rotation;

return newE;

}

public int Rotation

{

get

{

return \_rotation;

}

set

{

\_rotation = value;

}

}

[CategoryAttribute("1"), DescriptionAttribute("Ellipse")]

public string ObjectType

{

get

{

return "Ellipse";

}

}

public override void Select()

{

this.undoEle = this.Copy();

}

public override void AddGp(GraphicsPath gp, int dx, int dy )

{

gp.AddEllipse((this.X + dx) , (this.Y + dy) , (this.X1 - this.X) , (this.Y1 - this.Y) );

}

public override void Draw(Graphics g, int dx, int dy )

{

//System.Drawing.SolidBrush myBrush = new System.Drawing.SolidBrush(this.fillColor);

Brush myBrush = getBrush(dx, dy );

//myBrush.Color = this.Trasparency(this.fillColor, this.alpha);

System.Drawing.Pen myPen = new System.Drawing.Pen(this.penColor );

myPen.DashStyle = this.dashStyle;

if (this.Selected)

{

//myBrush.Color = this.dark(this.fillColor, 5, this.alpha);

myPen.Color = Color.Red;

myPen.Color = this.Trasparency(myPen.Color, 120);

myPen.Width = myPen.Width + 1;

}

//test

//myPen = PEN.getPen();

// Create a path and add the object.

GraphicsPath myPath = new GraphicsPath();

myPath.AddEllipse((this.X + dx) , (this.Y + dy) , (this.X1 - this.X) , (this.Y1 - this.Y) );

Matrix translateMatrix = new Matrix();

translateMatrix.RotateAt(this.Rotation, new PointF((this.X + dx + (int)(this.X1 - this.X) / 2) , (this.Y + dy + (int)(this.Y1 - this.Y) / 2) ));

myPath.Transform(translateMatrix);

// Draw the transformed ellipse to the screen.

if (this.filled)

{

g.FillPath(myBrush, myPath);

if (this.showBorder)

g.DrawPath(myPen, myPath);

}

else

g.DrawPath(myPen, myPath);

myPath.Dispose();

myPen.Dispose();

if (myBrush!=null)

myBrush.Dispose();

}

}

/// <summary>

/// Прямоугольник с закругленными углами ( estende Ele )

/// </summary>

///

[DescriptionAttribute("Rounded rectangle")]

[Serializable]

public class RRect : Ele

{

private int \_arcsWidth;

//private int \_rotation;

public RRect(int x, int y, int x1, int y1)

{

this.X = x;

this.Y = y;

this.X1 = x1;

this.Y1 = y1;

this.Selected = true;

this.arcsWidth = 20;

this.\_rotation = 0;

this.endMoveRedim();

this.rot = true; //can rotate?

}

public override void Select()

{

this.undoEle = this.Copy();

}

public override void AddGp(GraphicsPath gp, int dx, int dy )

{

float n = this.arcsWidth;

gp.AddArc(new RectangleF((this.X + dx) , (this.Y + dy) , n , n ), 180, 90);

gp.AddLine((this.X + dx + n / 2) , (this.Y + dy) , (this.X1 + dx - n / 2) , (this.Y + dy) );

gp.AddArc(new RectangleF((this.X1 + dx - n) , (this.Y + dy) , n , n ), 270, 90);

gp.AddLine((this.X1 + dx) , (this.Y + dy + n / 2) , (this.X1 + dx) , (this.Y1 + dy - n / 2) );

gp.AddArc(new RectangleF((this.X1 + dx - n) , (this.Y1 + dy - n) , n , n ), 0, 90);

gp.AddLine((this.X + dx + n / 2) , (this.Y1 + dy) , (this.X1 + dx - n / 2) , (this.Y1 + dy) );

gp.AddArc(new RectangleF((this.X + dx) , (this.Y1 + dy - n) , n , n ), 90, 90);

gp.AddLine((this.X + dx) , (this.Y1 + dy - n / 2) , (this.X + dx) , (this.Y + dy + n / 2) );

}

public override void Draw(Graphics g,int dx, int dy )

{

float n = this.arcsWidth;

//System.Drawing.SolidBrush myBrush = new System.Drawing.SolidBrush(this.fillColor);

Brush myBrush = getBrush(dx, dy );

//myBrush.Color = this.Trasparency(this.fillColor, this.alpha);

System.Drawing.Pen myPen = new System.Drawing.Pen(this.penColor );

myPen.DashStyle = this.dashStyle;

if (this.Selected)

{

//myBrush.Color = this.dark(this.fillColor, 5, this.alpha);

myPen.Color = Color.Red;

myPen.Color = this.Trasparency(myPen.Color, 120);

myPen.Width = myPen.Width + 1;

}

// Отрисовка боковин + округлений

GraphicsPath myPath = new GraphicsPath();

myPath.AddArc(new RectangleF((this.X + dx) , (this.Y + dy) , n , n ), 180, 90);

myPath.AddLine((this.X + dx + n / 2) , (this.Y + dy) , (this.X1 + dx - n / 2) , (this.Y + dy) );

myPath.AddArc(new RectangleF((this.X1 + dx - n) , (this.Y + dy) , n , n ), 270, 90);

myPath.AddLine((this.X1 + dx) , (this.Y + dy + n / 2) , (this.X1 + dx) , (this.Y1 + dy - n / 2) );

myPath.AddArc(new RectangleF((this.X1 + dx - n) , (this.Y1 + dy - n) , n , n ), 0, 90);

myPath.AddLine((this.X + dx + n / 2) , (this.Y1 + dy) , (this.X1 + dx - n / 2) , (this.Y1 + dy) );

myPath.AddArc(new RectangleF((this.X + dx) , (this.Y1 + dy - n) , n , n ), 90, 90);

myPath.AddLine((this.X + dx) , (this.Y1 + dy - n / 2) , (this.X + dx) , (this.Y + dy + n / 2) );

Matrix translateMatrix = new Matrix();

translateMatrix.RotateAt(this.Rotation, new PointF((this.X + dx + (int)(this.X1 - this.X) / 2) , (this.Y + dy + (int)(this.Y1 - this.Y) / 2) ));

myPath.Transform(translateMatrix);

// Draw the transformed ellipse to the screen.

if (this.filled)

{

g.FillPath(myBrush, myPath);

if (this.showBorder)

g.DrawPath(myPen, myPath);

}

else

g.DrawPath(myPen, myPath);

//TEST START

float gridSize = 4;

float gridRot = 45;

//FillWithLines(g, dx, dy, zoom, myPath, gridSize, gridRot);

//TEST END

myPath.Dispose();

myPen.Dispose();

if (myBrush != null)

myBrush.Dispose();

}

}

***Результат***

