Image Paint Simple Demo

```
/*
Code revised from chapter 6

GDI+ Custom Controls with Visual C# 2005
By Iulian Serban, Dragos Brezoi, Tiberiu Radu, Adam Ward

Language English
Paperback 272 pages [191mm x 235mm]
Release date July 2006
ISBN 1904811604

Sample chapter
http://international.us.server12.fileserver.kutayzorlu.com/files/download/2017/01/1604_CustomControls_SampleChapter_uuid-c42b9aba-e49d-40de-b471-e86e75ebe5f0_crc-0.pdf

For More info on GDI+ Custom Control with Microsoft Visual C# book
visit website www.packtpub.com

*/
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Text;
using System.Windows.Forms;
using System.Drawing;
using System.Drawing.Drawing2D;

namespace WarpControlApp1
{
public partial class Form1 : Form
{
```

```
Image img = null;
public Form1()
{
InitializeComponent();
}
private Image CreatePicture()
{
// Create a new Bitmap object, 50 x 50 pixels in size
Image canvas = new Bitmap(50, 50);
// create an object that will do the drawing operations
Graphics artist = Graphics.FromImage(canvas);
// draw a few shapes on the canvas picture
artist.Clear(Color.Lime);
artist.FillEllipse(Brushes.Red, 3, 30, 30, 30);
artist.DrawBezier(new Pen(Color.Blue, 3), 0, 0, 40, 15, 10,
35, 50, 50);
// now the drawing is done, we can discard the artist object
artist.Dispose();
//return the picture
return canvas;
}

private void Form1_Load(object sender, EventArgs e)
{
img = CreatePicture();
}

private void Form1_MouseUp(object sender, MouseEventArgs e)
{
Random rand = new Random(); // randomises our drawing
parameters
// set up all our parameters first before calling
DrawWarpedPicture.
Graphics target = this.CreateGraphics(); // draw onto the
form's surface
PointF pivotOnImage = new PointF(img.Width / 2, img.Height /
2);
```

```
PointF pivotOnTarget = new PointF((Single)e.X, (Single)e.Y);
double rotate = rand.NextDouble() * 360;
double scaleFactor = 0.2 + (rand.NextDouble() * 2);
SizeF skewing = new SizeF(rand.Next(-20, 21), rand.Next(-20,
21));
// draw it!
ImageWarper warper = new ImageWarper();
warper.DrawWarpedPicture(target,   img,   pivotOnImage,
pivotOnTarget, rotate, scaleFactor, skewing);

}

private System.ComponentModel.IContainer components = null;

///

/// Clean up any resources being used.
///
/// true if managed resources should be disposed; otherwise,
false. protected override void Dispose(bool disposing)
{
if (disposing && (components != null))
{
components.Dispose();
}
base.Dispose(disposing);
}

#region Windows Form Designer generated code

///

/// Required method for Designer support – do not modify
/// the contents of this method with the code editor.
///
private void InitializeComponent()
{
this.SuspendLayout();
//
// Form1
```

```
//
this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(292, 266);
this.Name = “”;
this.Text = “Click the form to draw”;
this.MouseUp += new
System.Windows.Forms.MouseEventHandler(this.Form1_MouseUp);
this.Load += new System.EventHandler(this.Form1_Load);
this.ResumeLayout(false);

}

#endregion

[STAThread]
static void Main()
{
Application.EnableVisualStyles();
Application.SetCompatibleTextRenderingDefault(false);
Application.Run(new Form1());
}

}
public class ImageWarper : Component
{
public void DrawWarpedPicture(
Graphics surface, // the surface to draw on
Image img, // the image to draw
PointF sourceAxle, // pivot point passing through image.
PointF destAxle, // pivot point's position on destination
surface
double degrees, // degrees through which the image is rotated
clockwise
double scale, // size multiplier
SizeF skew // the slanting effect size, applies BEFORE scaling
or rotation
)
```

```
{
// give this array temporary coords that will be overwritten
in the loop below
// the skewing is done here orthogonally, before any
trigonometry is applied
PointF[] temp = new PointF[3] {
new PointF(skew.Width, -skew.Height),
new PointF((img.Width – 1) + skew.Width, skew.Height),
new PointF(-skew.Width,(img.Height – 1) – skew.Height) };
double ang, dist;
double radians = degrees * (Math.PI / 180);
// convert the images corner points into scaled, rotated,
skewed and translated points
for (int i = 0; i < 3; i++) { // measure the angle to the
image's corner and then add the rotation value to it ang =
GetBearingRadians(sourceAxle, temp[i], out dist) + radians;
dist *= scale; // scale temp[i] = new
PointF((Single)((Math.Cos(ang) * dist) + destAxle.X),
(Single)((Math.Sin(ang) * dist) + destAxle.Y)); }
surface.DrawImage(img, temp); } private double
GetBearingRadians(PointF reference, PointF target, out double
distance) { double dx = target.X - reference.X; double dy =
target.Y - reference.Y; double result = Math.Atan2(dy, dx);
distance = Math.Sqrt((dx * dx) + (dy * dy)); if (result < 0)
result += (Math.PI * 2); // add the negative number to 360
degrees to correct the atan2 value return result; } } }
[/csharp]
```