Draw Rectangle

```
/*
GDI+ Programming in C# and VB .NET
by Nick Symmonds

Publisher: Apress
ISBN: 159059035X
*/
using System;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

namespace AllCornerRect
{
public class AllCornerRect : System.Windows.Forms.Form
{

#region Class Local Variables

RealRect MyRect;

#endregion

private System.ComponentModel.Container components = null;

public AllCornerRect() {
InitializeComponent();

this.SetStyle( ControlStyles.AllPaintingInWmPaint, true);
this.SetStyle( ControlStyles.DoubleBuffer, true);

this.MouseDown += new MouseEventHandler(this.M_down);
```

```
this.MouseUp += new MouseEventHandler(this.M_up);
this.MouseMove += new MouseEventHandler(this.M_move);

MyRect = new RealRect();
}

protected override void Dispose( bool disposing )
{
if( disposing )
{
if (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()
{
//
// AllCornerRect
//
this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
this.ClientSize = new System.Drawing.Size(320, 301);
this.MaximizeBox = false;
this.MinimizeBox = false;
this.Name = “AllCornerRect”;
this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen;
this.Text = “AllCornerRect”;
this.Load += new System.EventHandler(this.AllCornerRect_Load);
```

```
}
#endregion

[STAThread]
static void Main()
{
Application.Run(new AllCornerRect());
}

private void AllCornerRect_Load(object sender,
System.EventArgs e)
{
}

protected override void OnPaint(PaintEventArgs e)
{
base.OnPaint(e);

Graphics G = e.Graphics;

G.DrawRectangle(Pens.Red, MyRect.Rect);
}

#region Squeek

private void M_up(object sender, MouseEventArgs e)
{
Invalidate();
}

private void M_down(object sender, MouseEventArgs e)
{
if (e.Button != MouseButtons.Left)
return;

MyRect = new RealRect(e.X, e.Y);
}

private void M_move(object sender, MouseEventArgs e)
{
```

```
if (e.Button != MouseButtons.Left)
return;

MyRect.EndX = e.X;
MyRect.EndY = e.Y;
Invalidate();
}

#endregion
}

public class RealRect
{

#region Class Local Variables

private Point mStart;
private Point mEnd;
private Point mRealStart;
private Point mRealEnd;
private Size mRealSize;
private Rectangle mRect;

#endregion

public RealRect(int X, int Y)
{
mStart = Point.Empty;
mEnd = Point.Empty;
mRealStart = Point.Empty;
mRealEnd = Point.Empty;
mRealSize = Size.Empty;

mStart.X = X;
mStart.Y = Y;
mRealStart.X = X;
mRealStart.Y = Y;

mRect = Rectangle.Empty;
```

```
}

public RealRect()
{
mStart = Point.Empty;
mEnd = Point.Empty;
mRealStart = Point.Empty;
mRealEnd = Point.Empty;
mRealSize = Size.Empty;

mStart.X = 0;
mStart.Y = 0;
mRealStart.X = 0;
mRealStart.Y = 0;

mRect = Rectangle.Empty;
}

///

/// Ending X Value of rectangle
///
public int EndX
{
set{ mEnd.X = value; }
}

///

/// Ending Y Value of rectangle
///
public int EndY
{
set{ mEnd.Y = value; }
}

///

/// Get the corrected rectangle
///
```

```
public Rectangle Rect
{
get
{
MakeReal();
mRect.Location = mRealStart;
mRect.Size = mRealSize;
return mRect;
}
}

private void MakeReal()
{
//Started top left, ended bottom right
if (mEnd.X > mStart.X && mEnd.Y > mStart.Y)
{
mRealStart = mStart;
mRealEnd = mEnd;
mRealSize = new Size(mRealEnd.X-mRealStart.X, mRealEnd.Y-
mRealStart.Y);
return;
}

//Started bottom right, ended top left
if (mEnd.X < mStart.X && mEnd.Y < mStart.Y) { mRealEnd =
mStart; mRealStart = mEnd; mRealSize = new Size(mRealEnd.X-
mRealStart.X, mRealEnd.Y-mRealStart.Y); return; } //Started
top right left, ended bottom left if (mEnd.X < mStart.X &&
mEnd.Y > mStart.Y)
{
mRealStart.X = mEnd.X;
mRealStart.Y = mStart.Y;
mRealEnd.X = mStart.X;
mRealEnd.Y = mEnd.Y;
mRealSize = new Size(mRealEnd.X-mRealStart.X, mRealEnd.Y-
mRealStart.Y);
return;
```

```
}

//Started bottom left, ended top right
if (mEnd.X > mStart.X && mEnd.Y < mStart.Y) { mRealStart.X = mStart.X; mRealStart.Y = mEnd.Y; mRealEnd.X = mEnd.X; mRealEnd.Y = mStart.Y; mRealSize = new Size(mRealEnd.X-mRealStart.X, mRealEnd.Y-mRealStart.Y); return; } } } }
```
[/csharp]