Text Format

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

namespace TextFormat_c
{
public class TextFormat : System.Windows.Forms.Form
{

#region Class Local Variables
Rectangle UserRect = Rectangle.Empty;
bool RectStarted = false;
bool RectFinished = false;
bool Typing = false;
bool InsideRect = false;
Cursor ClientCursor = Cursors.Default;
string RectText = string.Empty;
StringFormat BoxFormat = StringFormat.GenericDefault;
#endregion

private System.Windows.Forms.Button cmdNew;
```

```
private System.Windows.Forms.Button cmdLeft;
private System.Windows.Forms.Button cmdTop;

private System.ComponentModel.Container components = null;

public TextFormat()
{
InitializeComponent();

//All the mouse move events use the same delegate
this.cmdNew.MouseMove                  +=                  new
MouseEventHandler(this.Button_MouseMove);
this.cmdLeft.MouseMove                  +=                  new
MouseEventHandler(this.Button_MouseMove);
this.cmdTop.MouseMove                  +=                  new
MouseEventHandler(this.Button_MouseMove);

this.KeyPress += new KeyPressEventHandler(this.FormKeyPress);

BoxFormat.Alignment = StringAlignment.Center;
BoxFormat.LineAlignment = StringAlignment.Center;
}

protected override void Dispose( bool disposing )
{
if( disposing )
{
if (components != null)
{
components.Dispose();
}
}
ClientCursor.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.cmdNew = new System.Windows.Forms.Button();
this.cmdLeft = new System.Windows.Forms.Button();
this.cmdTop = new System.Windows.Forms.Button();
this.SuspendLayout();
//
// cmdNew
//
this.cmdNew.Location = new System.Drawing.Point(16, 328);
this.cmdNew.Name = “cmdNew”;
this.cmdNew.Size = new System.Drawing.Size(56, 24);
this.cmdNew.TabIndex = 0;
this.cmdNew.Text = “&New”;
this.cmdNew.Click                  +=                  new
System.EventHandler(this.cmdNew_Click);
//
// cmdLeft
//
this.cmdLeft.Location = new System.Drawing.Point(104, 328);
this.cmdLeft.Name = “cmdLeft”;
this.cmdLeft.Size = new System.Drawing.Size(104, 24);
this.cmdLeft.TabIndex = 1;
this.cmdLeft.Text = “Center Left Align”;
this.cmdLeft.Click                  +=                  new
System.EventHandler(this.cmdLeft_Click);
//
// cmdTop
//
this.cmdTop.Location = new System.Drawing.Point(248, 328);
this.cmdTop.Name = “cmdTop”;
this.cmdTop.Size = new System.Drawing.Size(104, 24);
this.cmdTop.TabIndex = 2;
this.cmdTop.Text = “Top Left Align”;
```

```
this.cmdTop.Click += new
System.EventHandler(this.cmdTop_Click);
//
// TextFormat
//
this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
this.ClientSize = new System.Drawing.Size(392, 373);
this.Controls.AddRange(new System.Windows.Forms.Control[] {
this.cmdTop,
this.cmdLeft,
this.cmdNew});
this.MaximizeBox = false;
this.MinimizeBox = false;
this.Name = “TextFormat”;
this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen;
this.Text = “TextFormat”;
this.Load += new System.EventHandler(this.TextFormat_Load);
this.ResumeLayout(false);

}
#endregion

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

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

protected override void OnPaint(PaintEventArgs e)
{
Graphics G = e.Graphics;
using(Pen P = new Pen(Brushes.Black, 1))
```

```
{
if ( !RectFinished )
P.DashStyle = DashStyle.Dash;

G.DrawRectangle(P,UserRect);
}

if (RectFinished && ClientCursor == Cursors.IBeam)
G.DrawString(RectText, new Font("Arial", 16), Brushes.Black,
UserRect, BoxFormat);

base.OnPaint(e);
}

protected override void OnMouseDown(MouseEventArgs e)
{
//If left button then start the rectangle
if ( e.Button == MouseButtons.Left )
{
if (UserRect == Rectangle.Empty)
{
UserRect.X = e.X;
UserRect.Y = e.Y;
RectStarted = true;
RectFinished = false;
}
}
//If right button then start the edit
else if ( e.Button == MouseButtons.Right )
{
if ( UserRect != Rectangle.Empty )
{
ClientCursor = Cursors.IBeam;
this.Cursor = ClientCursor;
Point pos = new Point(UserRect.X+UserRect.Width/2,
UserRect.Y+UserRect.Height/2);
//Translate cursor screen postion to position on form
int Offset = this.Height-this.ClientRectangle.Height;
```

```
pos += new Size(this.Location.X, this.Location.Y+Offset);
Cursor.Position = pos;
Typing = true;
this.KeyPreview = true;
}
}
base.OnMouseDown(e);
}
protected override void OnMouseUp(MouseEventArgs e)
{
base.OnMouseUp(e);

// A negative rectangle is not allowed.
// Mouse_down then Mouse_up without Mouse_move is not allowed
if (UserRect.Width <= 0 || UserRect.Height <=0 ) UserRect = Rectangle.Empty; //Rectangle has ended RectStarted = false; RectFinished = true; Invalidate(); } protected override void OnMouseMove(MouseEventArgs e) { base.OnMove(e); //Let program know if cursor is inside user rectangle InsideRect = false; if (UserRect != Rectangle.Empty) if (UserRect.Contains(new Point(e.X, e.Y))) InsideRect = true; this.Cursor = ClientCursor; //Increase the size of the rectangle each time the mouse moves. if (RectStarted) { Size s = new Size(e.X-UserRect.X, e.Y-UserRect.Y); UserRect.Size=s; Invalidate(); } } private void cmdNew_Click(object sender, System.EventArgs e) { if ( Typing && InsideRect ) return; //Start a new blank form UserRect = Rectangle.Empty; ClientCursor = Cursors.Default; RectStarted = false; RectFinished = false; RectText = string.Empty; this.KeyPreview=false; Invalidate(); } private void Button_MouseMove(object sender, MouseEventArgs e) { this.Cursor = Cursors.Default; } void FormKeyPress(object sender, KeyPressEventArgs e) { //Handle backspace key if (e.KeyChar == (char)8) { if ( RectText != string.Empty ) RectText = RectText.Remove(RectText.Length-1, 1); } else RectText += e.KeyChar; Invalidate(); } private void cmdLeft_Click(object sender, System.EventArgs e) { //Change horizontal alignement and redraw BoxFormat.Alignment =
```

StringAlignment.Near; Invalidate(); } private void cmdTop_Click(object sender, System.EventArgs e) { //Chnage vertical alignment and redraw BoxFormat.LineAlignment = StringAlignment.Near; Invalidate(); } } } [/csharp]