Double Buffering

```
/*
User Interfaces in C#: Windows Forms and Custom Controls
by Matthew MacDonald

Publisher: Apress
ISBN: 1590590457
*/
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;

namespace GDI_Basics
{
///

/// Summary description for DoubleBuffering.
///
public class DoubleBuffering : System.Windows.Forms.Form
{
internal System.Windows.Forms.CheckBox chkDoubleBuffer;
internal System.Windows.Forms.Timer tmrRefresh;
private System.ComponentModel.IContainer components;

public DoubleBuffering()
{
//
// Required for Windows Form Designer support
//
InitializeComponent();

//
// TODO: Add any constructor code after InitializeComponent
```

```
call
//
}

///

/// Clean up any resources being used.
///
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()
{
this.components = new System.ComponentModel.Container();
this.chkDoubleBuffer = new System.Windows.Forms.CheckBox();
this.tmrRefresh                    =                    new
System.Windows.Forms.Timer(this.components);
this.SuspendLayout();
//
// chkDoubleBuffer
//
this.chkDoubleBuffer.BackColor                              =
System.Drawing.Color.PaleTurquoise;
this.chkDoubleBuffer.Location = new System.Drawing.Point(8,
```

```
8);
this.chkDoubleBuffer.Name = “chkDoubleBuffer”;
this.chkDoubleBuffer.Size = new System.Drawing.Size(336, 16);
this.chkDoubleBuffer.TabIndex = 1;
this.chkDoubleBuffer.Text = “Use Double Buffering”;
//
// tmrRefresh
//
this.tmrRefresh.Interval = 10;
this.tmrRefresh.Tick                    +=                    new
System.EventHandler(this.tmrRefresh_Tick);
//
// DoubleBuffering
//
this.AutoScaleBaseSize = new System.Drawing.Size(5, 14);
this.ClientSize = new System.Drawing.Size(376, 294);
this.Controls.AddRange(new System.Windows.Forms.Control[] {
this.chkDoubleBuffer});
this.Font  =  new  System.Drawing.Font(“Tahoma”,  8.25F,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((System.Byte)(0)));
this.Name = “DoubleBuffering”;
this.Text = “DoubleBuffering”;
this.Load                         +=                         new
System.EventHandler(this.DoubleBuffering_Load);
this.Paint                         +=                         new
System.Windows.Forms.PaintEventHandler(this.DoubleBuffering_Pa
int);
this.ResumeLayout(false);

}
#endregion

private bool isShrinking = false;
private int extraSize = 0;

private void tmrRefresh_Tick(object sender, System.EventArgs
e)
```

```
{
// Change the circle dimensions.

if (isShrinking)
{
extraSize–;
}
else
{
extraSize++;
}

// Change the sizing direction if needed.
if (extraSize > (this.Width – 150))
{
isShrinking = true;
}
```
else if (extraSize < 1) { isShrinking = false; } // Repaint the form. this.Invalidate(); } protected override void OnPaintBackground(System.Windows.Forms.PaintEventArgs pevent) { // Do nothing. } [STAThread] static void Main() { Application.Run(new DoubleBuffering()); } private void DoubleBuffering_Load(object sender, System.EventArgs e) { tmrRefresh.Start(); } private void DoubleBuffering_Paint(object sender, System.Windows.Forms.PaintEventArgs e) { Graphics g; Bitmap drawing = null; // Check if double buffering is needed, and assign the GDI+ context. if (chkDoubleBuffer.Checked) { drawing = new Bitmap(this.Width, this.Height, e.Graphics); g = Graphics.FromImage(drawing); } else { g = e.Graphics; } g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.HighQuality; // Draw a rectangle. Pen drawingPen = new Pen(Color.Black, 10); g.FillRectangle(Brushes.PaleTurquoise , new Rectangle(new Point(0, 0), this.ClientSize)); g.DrawEllipse(drawingPen, 50, 50, 50 + extraSize, 50 + extraSize); // If using double buffering, render the final image and dispose of it. if

```
(chkDoubleBuffer.Checked)                                    {
e.Graphics.DrawImageUnscaled(drawing, 0, 0); g.Dispose(); } }
} } [/csharp]
```