

SAMPLE CODE: GRAPHICS

EXAMPLE 1: SIMPLE GRAPHICS PROGRAM TO DRAW A RECTANGLE

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

namespace Graphics1
{
    /// This is the first graphics example from class on 7/9/07.
    /// This program does some simple drawing when Button1 is clicked.
    /// The drawing is not persistent--it disappears if the program
    /// is minimized or covered by something else and then uncovered.
    /// Button2 also demonstrates the lack of persistence by invalidating
    /// all or part of the image (depending on which line you use). When
    /// this happens, there is nothing to redraw the image.

    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Graphics g = this.CreateGraphics();
            Pen p = new Pen(Color.Blue, 5);
            g.DrawRectangle(p, 10, 10, 250, 250);
            p.Dispose(); //We should call Dispose since we created the
Pen and
            g.Dispose(); //Graphics object
        }

        private void button2_Click(object sender, EventArgs e)
        {
            // Invalidate();
            Invalidate(new Rectangle(0, 0, 100, 100));
        }
    }
}
```

EXAMPLE 2: USING A GRADIENT FILL, AND PROVIDING A HANDLER FOR THE FORM'S PAINT EVENT

```

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

namespace Graphics2
{
    /// This is the second graphics example from class on 7/9/07.
    /// This program makes the graphics persistent by installing a
    /// handler for the form's Paint event and doing the drawing
    /// in the handler. The button doesn't do anything--it's there
    /// just to show that Windows redraws it for us automatically.
    /// The program also demonstrates the use of a gradient brush
    /// for filling a rectangle. Also, calls of the paint handler
    /// are logged in the debug output window. (You have to run
    /// under the debugger to see this.) The IDE was used to create
    /// the stub of the Form2_Paint method and register it for the
    /// Paint event.

    public partial class Form2 : Form
    {
        private int paintHandlerCalls = 0;

        public Form2()
        {
            InitializeComponent();
        }

        private void Form2_Paint(object sender, PaintEventArgs e)
        {
            Graphics g = e.Graphics; //We don't create this so we won't call
Dispose on it
            Brush br = new LinearGradientBrush(new Point(10, 10), new Point(250,
250),
                Color.Blue, Color.Red);
            g.FillRectangle(br, 10, 10, 240, 240);
            br.Dispose();
            Debug.WriteLine("MyPaintHandler " + paintHandlerCalls++);
        }
    }
}

```

This line of code was generated by Visual Studio in the file Form2.Designer.cs to register for the Paint event:

```
this.Paint += new System.Windows.Forms.PaintEventHandler(this.Form2_Paint);
```

EXAMPLE 3: OVERRIDING THE ONPAINT METHOD. ALSO: HOW OFTEN DOES REPAINTING TAKE PLACE?

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

namespace Graphics3
{
    /// This is the third graphics example from class on 7/9/07.
    /// This program overrides the built-in OnPaint method and
    /// does the drawing there. It also counts the calls of OnPaint
    /// and displays the count in a text box. This code follows the
    /// .NET recommendation and calls base.OnPaint.

    public partial class Form3 : Form
    {
        int nPaints = 0;

        public Form3()
        {
            InitializeComponent();
        }

        protected override void OnPaint(PaintEventArgs e)
        {
            base.OnPaint(e);
            Graphics g = e.Graphics;
            Pen p = new Pen(Color.Red, 5);
            g.DrawRectangle(p, 10, 10, 240, 240);
            p.Dispose();
            nPaints++;
            textBox1.Text = nPaints.ToString();
        }
    }
}
```