

# Course 2DCis: 2D-Computer Graphics with C#

## Chapter C5: The Controls Project

Copyright © by V. Miszalok, last update: 11-12-2007

- ↓ [Projekt controls1](#)
- ↓ [Buttons](#)
- ↓ [Trackbars and Labels](#)
- ↓ [Checkboxes](#)
- ↓ [Radiobuttons](#)
- ↓ [Weitere Aufgaben](#)

A control is a prefabricated mini window of the Graphical User Interface GUI. Modern GUIs offer such prefabricated building blocks to build the surface of applications programs. They are delivered in form of a `Control` class containing `Button`, `CheckBox`, etc.

Programming such controls is extremely simple: Copy them per Drag&Drop from a toolbox onto Your `Form1`, and specify positions and sizes with the mouse. Visual Studio generates automatically the necessary code = Visual Programming. You just have to insert the strings for the names and the lettering. All beginners are fascinated by this comfort and use it extensively.

**Professionals don't use this comfort because they know the drawbacks:**

1. The automatically generated code is full of stupid redundancy.
2. The controls don't know their predecessor, successor and neighbour. They behave like loners, not as a team. Every control has its own code, being mostly identical to the code of its predecessor, successor and neighbour. When the programmer wants to force his controls to an uniform behaviour, he has to copy this uniform code into all definitions and event handlers. It's hard to treat the controls all together in form of a `for`-loop.
3. The controls have fixed positions and sizes. They do not react on zoom = `OnResize`-events of their home form. They do not adapt to changing window sizes, which is obligatory for programmes aimed for mobile devices = PDAs, handys, Internet etc.
4. Any control generates its own event-handler functions, creating a lot of useless and confusing code.

**Consequence:**

1. Professionals hate visual programming and prefer to program their controls by hand.
2. They summarise controls to control arrays and initialise them in a common `for`-loop.
3. They summarise controls with common behaviour in form of a dynamic array `cc` and program one common `OnResize`-event using a `for`-loop of `foreach`-loop.
4. Slim codes direct events of similar controls onto one single event handler with internal distribution:

```
protected void common_event_handler( object sender, System.EventArgs e )
{ switch( sender )
  { case control[0]: ;;; break;
    case control[1]: ;;; break;
    etc.           ;;; break;
  }
}
```

## Project controls1

Main Menu after starting VS 2005: File -> New Project... ->  
 Visual Studio installed templates: Windows Forms Application  
 Name: controls1 -> Location: C:\temp ->  
 Create directory for solution: switch off -> OK  
 Delete all prefabricated code as described in the former chapters C1 to C4.

# Buttons

Write the following lines into the empty window of Form1.cs:

```
using System;
using System.Drawing;
using System.Windows.Forms;

public class Form1 : Form
{
    static void Main() { Application.Run( new Form1() ); }
    const Int32 nButtons = 3, nTrackBars = 3, nCheckBoxes = 3, nRadioButtons = 3;
    Button [] button = new Button[nButtons];
    TrackBar[] trackbar = new TrackBar[nTrackBars];
    Label [] label = new Label[nTrackBars];
    CheckBox[] checkbox = new CheckBox[nCheckBoxes];
    static RadioButton[] radiobutton = new RadioButton[nRadioButtons];
    static Color linecolor = Color.Black;
    static Pen pen = new Pen( linecolor, 1 );
    static Panel panel = new Panel();
    static Graphics g;
    static Random r = new Random();
    Timer myTimer = new Timer();

    public Form1()
    {
        BackColor = Color.White;
        Text = "GUI";
        Int32 i;
        for ( i=0; i < nButtons; i++ )
        {
            button[i] = new Button(); Controls.Add( button[i] );
            button[i].Click += new EventHandler( button_handler );
        }
        button[0].Text = "Start";
        button[1].Text = "Stop";
        button[2].Text = "Clear";

        foreach ( Control c in Controls )
        {
            c.BackColor = Color.Gray;
            if ( c.Text == "" ) c.Text = "nothing";
        }
        Controls.Add( panel ); //Put a drawing space on Form1
        myTimer.Tick += new EventHandler( OnTimer );
        myTimer.Interval = 1;
        Width = 800; Height = 600;
    }
    protected override void OnResize( EventArgs e )
    {
        Int32 w = ClientRectangle.Width / 5;
        Int32 h = ClientRectangle.Height / (Controls.Count-1);
        Int32 i, top = 1;
        for ( i=0; i < Controls.Count-1; i++ )
        {
            Controls[i].Top = top;
            Controls[i].Left = 2;
            Controls[i].Width = w;
            Controls[i].Height = h - 2;
            top += h;
        }
        panel.Location = new Point( w+2, 0 );
        panel.Size = new Size( ClientRectangle.Width-panel.Location.X, ClientRectangle.Height );
        g = panel.CreateGraphics();
        g.Clear( SystemColors.Window );
    }
    protected void button_handler( object sender, System.EventArgs e )
    {
        switch( ((Button)sender).Text )
        {
            case "Start": myTimer.Start(); break;
            case "Stop" : myTimer.Stop(); break;
            case "Clear": g.Clear( SystemColors.Window ); break;
        }
    }
    protected static void OnTimer( Object myObject, EventArgs myEventArgs )
    {
        Int32 w = r.Next( panel.Width/2 ), h = r.Next( panel.Height/2 );
        Int32 x = r.Next( panel.Width-w ), y = r.Next( panel.Height-h );
        g.DrawLine ( pen, x, y, x+w, y+h );
    }
}
}
```

Click Debug -> Start Without Debugging Ctrl F5.

Try out the program. At run time change the window size. Increase the constant `const Int32 nButtons = 3;` to `nButtons = 8;` in the 3. line of Form1 just for fun.

## Trackbars and Labels

Version2: Finish program controls1.

Insert the following code lines into the constructor Form1() below the existing line button[2].Text = "Clear";:

```
for ( i=0; i < nTrackBars; i++ )
{
    trackbar[i] = new TrackBar(); Controls.Add( trackbar[i] );
    label [i] = new Label(); Controls.Add( label[i] );
    trackbar[i].AutoSize = false;
    trackbar[i].TickStyle = TickStyle.None;
    trackbar[i].ValueChanged += new EventHandler( trackbar_handler );
    label [i].TextAlign = ContentAlignment.TopCenter;
}
trackbar[0].Name = label[0].Text = "TimerInterval";
trackbar[1].Name = label[1].Text = "PenThickness";
trackbar[2].Name = label[2].Text = "Brightness";
trackbar[0].Minimum = 13; trackbar[0].Maximum = 1000;
trackbar[1].Minimum = 1; trackbar[1].Maximum = 20;
trackbar[2].Minimum = 0; trackbar[2].Maximum = 255;
```

Write the following two functions between the existing functions protected void button\_handler( ... ) and protected static void OnTimer( ... ):

```
protected void trackbar_handler( object sender, System.EventArgs e )
{
    Int32 value = ((TrackBar)sender).Value;
    switch( ((TrackBar)sender).Name )
    {
        case "TimerInterval":
            myTimer.Interval = value;
            label[0].Text = "TimerInterval = " + value.ToString(); break;
        case "PenThickness":
            pen.Width = value;
            label[1].Text = "PenThickness = " + value.ToString(); break;
        case "Brightness":
            label[2].Text = "Brightness = " + value.ToString();
            checkbox_handler( sender, e ); break; //call checkbox_handler
    }
}
protected void checkbox_handler( object sender, System.EventArgs e )
{
}
```

Write the following line into the function protected override void OnResize(...) **below** the for-loop and **in front of** line panel.Location = new Point( w+2, 0 );:

```
for ( i=0; i < nTrackBars; i++ ) trackbar[i].Height = h;
```

Click Debug -> Start Without Debugging Ctrl F5.

Try out controls1. The last trackbar "Brightness" doesn't work yet.

Increase constant const Int32 nTrackBars = 3; to nTrackBars = 8; in the 3. line of Form1.

## CheckBoxes

Version3: Finish controls1. Write the following additional code into the constructor Form1() below the line trackbar[2].Minimum = 0; trackbar[2].Maximum = 255;:

```
for ( i=0; i < nCheckBoxes; i++ )
{
    checkbox[i] = new CheckBox(); Controls.Add( checkbox[i] );
    checkbox[i].TextAlign = ContentAlignment.MiddleCenter;
    checkbox[i].Click += new EventHandler( checkbox_handler );
}
checkbox[0].Text = "Red";
checkbox[1].Text = "Green";
checkbox[2].Text = "Blue";
```

Fill the empty function protected void checkbox\_handler( ... ):

```
protected void checkbox_handler( object sender, System.EventArgs e )
{
    Int32 v;
    if ( sender == trackbar[2] ) v = trackbar[2].Value; //call from "Brightness"
    else v = trackbar[2].Value = 255; //call from CheckBox
    linecolor = Color.Black; //start color
    if ( checkbox[0].Checked ) linecolor = Color.FromArgb( v,0,0 );
    if ( checkbox[1].Checked ) linecolor = Color.FromArgb( linecolor.R,v,0 );
    if ( checkbox[2].Checked ) linecolor = Color.FromArgb( linecolor.R,linecolor.G,v );
    //If all checkboxes are empty, the linecolor remains black. Replace it by gray.
    if ( linecolor == Color.Black && v > 0 ) linecolor = Color.FromArgb( v,v,v );
    pen.Color = linecolor;
}
```

Click Debug -> Start Without Debugging Ctrl F5.

Try out controls1. The last trackbar "Brightness" works now.

Increase the constant const Int32 nCheckBoxes = 3; to nCheckBoxes = 8; in the 3. line of Form1.

## RadioButtons

Version5: Finish controls1.

Write the following additional code into the constructor Form1() below the line

```
checkbox[2].Text = "Blue";:
    for ( i=0; i < nRadioButtons; i++ )
    { radiobutton[i] = new RadioButton(); Controls.Add( radiobutton[i] );
      radiobutton[i].TextAlign = ContentAlignment.MiddleCenter;
    }
    radiobutton[0].Text = "Lines"; radiobutton[0].Checked = true;
    radiobutton[1].Text = "Rectangles";
    radiobutton[2].Text = "Ellipses";
```

Verändern Sie die Funktion protected static void OnTimer( ... ), dass sie so aussieht:

```
protected static void OnTimer( Object myObject, EventArgs myEventArgs )
{ Int32 w = r.Next( panel.Width/2 ), h = r.Next( panel.Height/2 );
  Int32 x = r.Next( panel.Width-w ), y = r.Next( panel.Height-h );
  if ( radiobutton[0].Checked ) g.DrawLine ( pen, x, y, x+w, y+h );
  else if ( radiobutton[1].Checked ) g.DrawRectangle( pen, x, y, w, h );
  else if ( radiobutton[2].Checked ) g.DrawEllipse ( pen, x, y, w, h );
}
```

Click Debug -> Start Without Debugging Ctrl F5.

Try out controls1. Increase the constant

const Int32 nRadioButtons = 3; auf nRadioButtons = 8; in the 3. line of Form1.

## Exercises

- 1) Change the color of the controls.
- 2) Narrow all controls and enlarge panel instead.
- 3) Shift all controls from the left to the right window border and shift panel to the left instead.
- 4) Shift all controls from the left to the upper window border and shift panel down instead.
- 5) Shift all controls from the left to the lower window border and shift panel up instead.
- 6) Distribute the controls between the left and the right window border.
- 7) Frame all controls with a black rim.