Course 2DCis: 2D-Computer Graphics with C#
Chapter C5: The Controls Project

- Projekt controls1
- Buttons
- Trackbars und Labels
- Checkboxes
- Radiobuttons
- Weitere Aufgaben

In Informatiksprache ist ein Control ein vorgefertigtes Steuerelement des Graphical User Interface GUI.
Ein modernes GUI bietet dem Programmierer vorgefertigte Bausteine für die Programmoberfläche als Bibliothek von rechteckigen Steuerelementen = Controls.
Das Auswählen, Einbauen und Programmieren solcher Controls ist in Visual Studio extrem einfach:
Man zieht sie per Drag&Drop aus einer Toolbox auf sein Programmfenster, bestimmt Lage und Größe mit der Maus und Visual Studio generiert Code für jeden Control = Visual Programming.
Man braucht nur noch die Strings für Namen und/oder Beschriftung eingeben. Alle Anfänger sind fasziniert von diesem Komfort und benutzen ihn.
Der Profi benutzt diesen Komfort nicht, weil er die Nachteile kennt:
1. Der automatisch generierte Code ist voll von dummer Redundanz.
2. Kein Control kennt seinen Vorgänger, Nachfolger, Nachbarn, es verhält sich wie ein Einzelgänger, keinesfalls als Teil eines Teams.
Jedes Control besitzt seinen eigenen Code, auch wenn dieser mehr oder weniger identisch mit dem seines Vorgängers, Nachfolgers, Nachbarn ist. Will der Programmierer alle Controls zu gleichartigem Verhalten zwingen, dann muss er in jeden einzelnen Code dieses Verhalten nachträglich einbauen.
Er kann seine Controls nicht in Form einer for-Schleife gemeinsam bearbeiten.
3. Die Controls haben feste Plätze und Größen. Sie reagieren nicht auf Zoom = `OnResize`-Ereignisse der Mutterform. Sie passen sich nicht an wechselnde Fenstergrößen an, was aber obligatorisch ist für Programme, die auf Mobile Devices = PDAs, Handies, Internet etc. laufen sollen.
4. Jedes Control generiert seine eigene(n) EventHandler-Funktion(en),
auch wenn diese nutzlos und/oder unübersichtlich sind.
Folge:
1. Profis hassen Visual Programming und programmieren ihre Controls selbst.
2. Sie fassen gleichartige Controls zu Control-Arrays zusammen und initialisieren diese gemeinsam in einer for-Schleife.
3. Sie fassen Controls, die sich gleich verhalten sollen in einem dynamischen Array `cc` zusammen und programmieren das `OnResize`-Ereignis gemeinsam für alle in einer Schleife.
4. Sie lenken Events gleichartiger Controls zur besseren Übersicht gemeinsam auf einen EventHandler mit interner Verteilung:

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

Projekt controls1

Main Menu nach dem Start von VS 2008: `File -> New Project... ->`
`Visual Studio installed templates: Windows Forms Application`
`Name: controls1 -> Location: C:\temp ->`
`Create directory for solution:` ausschalten `-> OK`
Löschen Sie den gesamten vorgefertigten Code wie es in den vorigen Kapiteln C1 bis C4 beschrieben wurde.

Buttons

Sie schreiben in das leere Codefenster `Form1.cs` folgenden Code:

```
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 );
  }
}
```

Klicken Sie `Debug` -> `Start Without Debugging Ctrl F5`.
Erproben Sie das Programm. Ändern Sie zur Laufzeit die Fenstergröße. Erhöhen Sie probeweise in der 3. Zeile von `Form1` die Konstante `const Int32 nButtons = 3;` auf `nButtons = 8;`

Trackbars und Labels

Version2: Beenden Sie Ihr Programm `controls1`.
Schreiben Sie in den Konstruktor `Form1()` unterhalb der Zeile `button[2].Text = "Clear";` folgenden zusätzlichen Code:

```
    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;
```

Schreiben Sie zwischen die beiden Funktionen `protected void button_handler( ... )` und `protected static void OnTimer( ... )` folgende zwei weiteren Funktionen:

```
  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 )
  {
  }
```

Schreiben Sie in die Funktion `protected override void OnResize(...)` **unter** die `for`-Schleife und **vor** die Zeile `panel.Location = new Point( w+2, 0 );` noch folgende Zeile:

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

Klicken Sie `Debug` -> `Start Without Debugging Ctrl F5`.
Erproben Sie `controls1`. Der letzte Trackbar "`Brightness`" funktioniert noch nicht.
Erhöhen Sie probeweise in der 3. Zeile von `Form1` die Konstante
`const Int32 nTrackBars = 3;` auf `nTrackBars = 8;`

CheckBoxes

Version3: Beenden Sie Ihr Programm `controls1`.
Schreiben Sie in den Konstruktor `Form1()` unterhalb der Zeile `trackbar[2].Minimum = 0; trackbar[2].Maximum = 255;` folgenden zusätzlichen Code:

```
    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";
```

Verändern Sie die noch leere Funktion `protected void checkbox_handler( ... )`, dass sie so aussieht:

```
  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;
  }
```

Klicken Sie `Debug` -> `Start Without Debugging Ctrl F5`.
Erproben Sie `controls1`. Der letzte Trackbar "`Brightness`" funktioniert jetzt. Erhöhen Sie probeweise in der 3. Zeile von `Form1` die Konstante `const Int32 nCheckBoxes = 3;` auf `nCheckBoxes = 8;`

RadioButtons

Version4: Beenden Sie Ihr Programm `controls1`.
Schreiben Sie in den Konstruktor `Form1()` unterhalb der Zeile `checkbox[2].Text = "Blue";` folgenden zusätzlichen Code:

```
    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  );
  }
```

Klicken Sie `Debug` -> `Start Without Debugging Ctrl F5`.
Erproben Sie `controls1`. Erhöhen Sie probeweise in der 3. Zeile von `Form1` die Konstante `const Int32 nRadioButtons = 3;` auf `nRadioButtons = 8;`

Weitere Aufgaben zur Übung

1) Geben Sie den Controls eine andere Farbe.
2) Machen Sie alle Controls schmaler und dafür die Zeichenfläche `panel` breiter.
3) Verlagern Sie alle Controls vom linken an den rechten Fensterrand und verschieben Sie dafür `panel` nach links.
4) Verlagern Sie alle Controls vom linken an den oberen Fensterrand und verschieben Sie dafür `panel` nach unten.
5) Verlagern Sie alle Controls vom linken an den unteren Fensterrand und verschieben Sie dafür `panel` nach oben.
6) Verteilen Sie die Controls auf den linken und den rechten Fensterrand.
7) Geben Sie den Controls je einen schwarzen Rand.