

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

Chapter C5: Comments to the Controls Project

Copyright © by V. Miszalok, last update: 04-01-2006

using namespaces

```
using System; //Namespace of the base class of all classes "System.Object" and of all primitive data
types such as Int32, Int16, Single, Double, String.
```

```
using System.Drawing; //Namespace of the "Graphics" class and its drawing methods such as
DrawString, DrawLine, DrawRectangle, FillClosedCurve etc.
```

```
using System.Windows.Forms; //Namespace of the "Form" class (base class of our main window Form1)
and its method Application.Run.
```

Global variables of public class Form1 : Form

All objects that are used inside the function OnTimer(...) (see below) must be declared "static", because the function OnTimer(...) is static. There is only one Timer object for all instances of Form1. No matter how often You start Form1, any instance uses the same Timer. Any "static" object resides only once inside its class and not inside each instance of its class as nonstatic objects do. See the help text in Visual Studio .NET: Help -> Index -> Filtered by: .NET Framework SDK -> Look for: static keyword, C#.

```
static void Main() { Application.Run( new Form1() ); } //Create an instance of Form1 and
ask the operating system to start it as main window of our program.
```

```
const Int32 nButtons = 3, nTrackBars = 3, nCheckBoxes = 3, nRadioButtons = 3;
//Integers that never change their values indicating the numbers of Buttons, TrackBars etc.
```

```
Button [] button = new Button[nButtons]; //Array of 3 Buttons: Start, Stop, Clear.
```

```
TrackBar[] trackbar = new TrackBar[nTrackBars]; //Array of 3 TrackBars.
```

```
Label [] label = new Label[nTrackBars]; //Array of 3 Labels aimed at lettering the 3 TrackBars:
TimerInterval, PenThickness, Brightness.
```

```
CheckBox[] checkbox = new CheckBox[nCheckBoxes]; //Array of 3 CheckBoxes: Red, Green, Blue.
```

```
static RadioButton[] radiobutton = new RadioButton[nRadioButtons]; //static Array of 3
RadioButtons: Lines, Rectangles, Ellipses. This array must be static because it will be accessed by the static
event handler OnTimer(...).
```

```
static Color linecolor = Color.Black; //Initial drawing color.
```

```
static Pen pen = new Pen( linecolor, 1 ); //Initial Pen.
```

```
static Panel panel = new Panel(); //Drawing area covering the complete window space right of the
vertical control bar.
```

```
static Graphics g; //Device context that all drawing commands need.
```

```
static Random r = new Random(); //Generator for random x,y-coordiantes of lines, rectangles and
ellipses.
```

```
Timer myTimer = new Timer(); //This Timer sends messages at fixed time intervals to Form1, that
trigger Form1 to execute its OnTimer(...) method.
```

Constructor public Form1()

```

BackColor = Color.White; //Background color of Form1 and of panel.

Text = "GUI" //Title in the blue header line of Form1.

Int32 i; //for-loop counter.

for ( i=0; i < nButtons; i++ ) //3 buttons for Form1.

button[i] = new Button(); Controls.Add( button[i] ); //Create a button and affiliate it to
Form1.

button[i].Click += new EventHandler( button_handler ); //Redirect all click events to our
button_handler-function.

button[0].Text = "Start"; //Inscription of button 0;

button[1].Text = "Stop"; //Inscription of button 1;

button[2].Text = "Clear"; //Inscription of button 2;

for ( i=0; i < nTrackBars; i++ ) //3 track bars for Form1.

trackbar[i] = new TrackBar(); Controls.Add( trackbar[i] ); //Create a track bar and affiliate
it to Form1.

label [i] = new Label(); Controls.Add( label[i] ); //Create a label and affiliate it to Form1.

trackbar[i].AutoSize = false; //A value indicating whether the height or width of the track bar is being
automatically sized.

trackbar[i].TickStyle = TickStyle.None; //A value indicating how to display the tick marks on the
track bar.

trackbar[i].ValueChanged += new EventHandler( trackbar_handler ); //Redirect all
ValueChanged events to our trackbar_handler-function.

label [i].TextAlign = ContentAlignment.TopCenter; //Text is vertically aligned at the top, and
horizontally aligned at the center.

trackbar[0].Name = label[0].Text = "TimerInterval" //Inscription of label 0 underneath track bar
0.

trackbar[1].Name = label[1].Text = "PenThickness" //Inscription of label 1 underneath track bar
1.

trackbar[2].Name = label[2].Text = "Brightness" //Inscription of label 2 underneath track bar 2.

trackbar[0].Minimum = 13; trackbar[0].Maximum = 1000; //Lower and upper delimiter of
"TimerInterval".

trackbar[1].Minimum = 1; trackbar[1].Maximum = 20; //Lower and upper delimiter of
"PenThickness".

trackbar[2].Minimum = 0; trackbar[2].Maximum = 255; //Lower and upper delimiter of
"Brightness".

for ( i=0; i < nCheckBoxes; i++ ) //3 check boxes for Form1.

checkbox[i] = new CheckBox(); Controls.Add( checkbox[i] ); //Create a check box and
affiliate it to Form1.

```

```
checkbox[i].TextAlign = ContentAlignment.MiddleCenter; //Text is vertically aligned at the
middle, and horizontally aligned at the center.

checkbox[i].Click += new EventHandler( checkbox_handler ); //Redirect all click events to our
checkbox_handler-function.

checkbox[0].Text = "Red"; //Inscription of check box 0

checkbox[1].Text = "Green"; //Inscription of check box 1

checkbox[2].Text = "Blue"; //Inscription of check box 2

for ( i=0; i < nRadioButtons; i++ ) //3 radio buttons for Form1.

radiobutton[i] = new RadioButton(); Controls.Add( radiobutton[i] ); //Create a radio
button and affiliate it to Form1.

radiobutton[i].TextAlign = ContentAlignment.MiddleCenter; //Text is vertically aligned at the
middle, and horizontally aligned at the center.

radiobutton[0].Text = "Lines"; radiobutton[0].Checked = true; //Inscription of radio button
0. Button is pressed.

radiobutton[1].Text = "Rectangles"; //Inscription of radio button 1.

radiobutton[2].Text = "Ellipses"; //Inscription of radio button 2.

foreach ( Control c in Controls ) //loop around all different controls.

c.BackColor = Color.Gray; //All controls are painted in gray.

if ( c.Text == "" ) c.Text = "nothing"; //All controls without inscription obtain inscription
"nothing".

Controls.Add( panel ); //Put a drawing canvas on Form1 which will cover the empty client area. Its better
to use such a canvas instead of drawing directly onto the client area of Form1, because this canvas can be
shifted on Form1 but holds its own relative coordinates always starting at (0/0).

myTimer.Tick += new EventHandler( OnTimer ); //Redirects the Timer messages to our Form1-
method OnTimer(...).

myTimer.Interval = 1; //Send the Timer messages at the shortest possible time interval (1 millisecond
means as fast as possible).

Width = 800; //Starting width of Form1. This statement is not obligatory but the default width is rather narrow
for drawing. This statement raises a first OnResize-event.

Height = 600; //Starting height of Form1. This statement is not obligatory but the default height is rather
narrow for drawing. This statement raises a second OnResize-event which overrides the first one.
```

```
Overridden event handler protected override void OnResize( EventArgs e )
```

```

Int32 w = ClientRectangle.Width / 5; //Control width = 20% of Form1-width.

Int32 h = ClientRectangle.Height / (Controls.Count-1); //Control height = Form1-width
divided by the number of controls (without the last one = panel).

Int32 i, top = 1; //i = loop counter, top = stepping y-coordinate.

for ( i=0; i < Controls.Count-1; i++ ) //loop through all controls without the last one = panel.
Controls[i].Top = top; //y-position.

Controls[i].Left = 2; //Common x-position of all controls leaving a narrow left margin.

Controls[i].Width = w; //Common width of all controls.

Controls[i].Height = h - 2; //Common height of all controls leaving a narrow bottom margin.

top += h; //Next y-position below the predecessor control.

for ( i=0; i < nTrackBars; i++ ) trackbar[i].Height = h; //Suppress the narrow bottom
margins between the track bars and their labels.

panel.Location = new Point( w+2, 0 ); //Canvas relative coordinates: x = right of the horizontal
control bar leaving a left narrow margin, y = top of <Form1 client area.

panel.Size = new Size( ClientRectangle.Width-panel.Location.X,
ClientRectangle.Height ); //Canvas covers all the rest of Form1 client area.

g = panel.CreateGraphics(); //All further Draw-commands will address this canvas.

g.Clear( SystemColors.Window ); //Clear the canvas with Form1 background color.

```

```
Event handler protected void button_handler( object sender, System.EventArgs e )
```

```

switch( ((Button)sender).Text ) // Cast object sender to type Button and read its inscription, which
serves as dispenser.

case "Start": myTimer.Start(); break; //Unlock myTimer.

case "Stop" : myTimer.Stop(); break; //Lock myTimer.

case "Clear": g.Clear( SystemColors.Window ); break; //Erase the canvas.

```

```
Event handler protected void trackbar_handler( object sender, System.EventArgs e )
```

```

Int32 value = ((TrackBar)sender).Value; //Cast object sender to type TrackBar and read its
current Value.

switch( ((TrackBar)sender).Name ) //Cast object sender to type TrackBar and read its Name, which
serves as dispenser.

case "TimerInterval": //The event comes from trackbar[0].

myTimer.Interval = value; //Set new Timer delay.

label[0].Text = "TimerInterval = " + value.ToString(); break; //Display the new delay in
label[0] below trackbar[0].

```

```

case "PenThickness": //The event comes from trackbar[1].


---


pen.Width = value; //Change the pen.Width property.


---


label[1].Text = "PenThickness = " + value.ToString(); break; //Display the new thickness in
label[1] below trackbar[1].


---


case "Brightness": //The event comes from trackbar[2].


---


label[2].Text = "Brightness = " + value.ToString(); //Display the new brightness in label[2]
below trackbar[2].


---


checkbox_handler( sender, e ); break; //call checkbox_handler //Call the event handler for
color changes.


---



```

```

Event handler protected void checkbox_handler( object sender, System.EventArgs e )


---



```

```

Int32 v; //Local variable carrying TrackBar values.


---


if ( sender == trackbar[2] ) v = trackbar[2].Value; //Call comes from "Brightness"-trackbar.
Take the new brightness from trackbar[2].


---


else v = trackbar[2].Value = 255; //Call comes from a CheckBox. In this case create a full red
and/or green and/or blue.


---


linecolor = Color.Black; //Black = default color = start color.


---


if ( checkbox[0].Checked ) linecolor = Color.FromArgb( v,0,0 ); //Just new red.


---


if ( checkbox[1].Checked ) linecolor = Color.FromArgb( linecolor.R,v,0 ); //Old red
plus new green.


---


if ( checkbox[2].Checked ) linecolor = Color.FromArgb( linecolor.R,linecolor.G,v
); //Old red plus old green plus new blue.


---


if ( linecolor == Color.Black && v > 0 ) linecolor = Color.FromArgb( v,v,v ); //If
linecolor == default color all checkboxes have been empty. But if brightness is > 0, the line color should be
some sort of gray.


---


pen.Color = linecolor; //Change the pen.Color property.


---



```

```

Event handler protected static void OnTimer( Object myObject, EventArgs myEventArgs )


---



```

```

Int32 w = r.Next( panel.Width/2 ), h = r.Next( panel.Height/2 ); //Random width with
maximally half the panel-width and random height with maximally half the panel-height.


---


Int32 x = r.Next( panel.Width-w ), y = r.Next( panel.Height-h ); //Random x so, that
width always fits into panel-width and random y so, that height always fits into panel-height.


---


if ( radiobutton[0].Checked ) g.DrawLine ( pen, x, y, x+w, y+h ); //A random line.


---


else if ( radiobutton[1].Checked ) g.DrawRectangle( pen, x, y, w, h ); //A random
rectangle.


---


else if ( radiobutton[2].Checked ) g.DrawEllipse ( pen, x, y, w, h ); //A random
ellipse.


---



```