Course 2DCis: 2D-Computer Graphics with C#
Chapter C3: The XML Project

Projekt xml1 mit leerem Fenster

`0)` Main Menu nach dem Start von VS 2008: `Tools → Options →` check lower left checkbox: `Show all Settings → Projects and Solutions → Visual Studio projects location: → C:\temp`

`1)` Main Menu nach dem Start von VS 2008: `File → New Project... →`
`Visual Studio installed templates: Windows Forms Application`
`Name: xml1 → Location: C:\temp → Create directory for solution:` ausschalten `→ OK`
Es meldet sich `Form1.cs[Design]`.

`2)` Sie müssen zwei überflüssige Files löschen: `Form1.Designer.cs` und `Program.cs`.
Sie erreichen diese Files über das `Solution Explorer - xml1`-Window:
Klicken Sie das Pluszeichen vor `xml1` und dann das Pluszeichen vor `Form1.cs`.
Klicken Sie mit der **rechten** Maustaste auf den Ast `Program.cs`. Es öffnet sich ein Kontextmenu.
Sie Klicken auf `Delete`. Eine Message Box erscheint: `'Program.cs' will be deleted permanently.`
Sie quittieren mit `OK`.
Klicken Sie mit der **rechten** Maustaste auf den Ast `Form1.Designer.cs` und löschen auch dieses File.

`3)` Klicken Sie mit der **rechten** Maustaste auf das graue Fenster `Form1`.
Es öffnet sich ein kleines Kontextmenü. Klicken Sie auf `View Code`.
Sie sehen jetzt den vorprogrammierten Code von `Form1.cs`. Löschen Sie den gesamten Code vollständig.

`4)` Schreiben Sie in das vollständig leere File `Form1.cs` folgende 3 Zeilen:

```
public class Form1 : System.Windows.Forms.Form
{ static void Main() { System.Windows.Forms.Application.Run( new Form1() ); }
}
```

`5)` Klicken Sie `Debug` im Main Menu oberhalb des Hauptfensters.
Es öffnet sich ein Untermenü. Klicken Sie auf `Start Without Debugging Ctrl F5`.

Wichtig: Zuerst alle Instanzen von `xml1` beenden, bevor neuer Code eingegeben und übersetzt wird !
Im Zweifel Task Manager mit `Ctrl+Alt+Del` starten und kontrollieren, ob noch ein `xml1.exe`-Prozess läuft und diesen töten.

Malprogramm mit dynamischem Array

Falls Sie den Code nicht mehr sehen, klicken Sie in Visual Studio auf den Karteireiter **`Form1.cs`**,
das bringt den Code zum Vorschein.
Sie löschen alles und schreiben in das leere Codefenster `Form1.cs` folgenden Code:

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

public class Form1 : Form
{ [STAThread] static void Main() { Application.Run( new Form1() ); }
  Graphics g;
  Point p0 = new Point();
  Point p1 = new Point();
  ArrayList polygon = new ArrayList();
  System.Text.StringBuilder polygon_string = new System.Text.StringBuilder();
  Brush redbrush   = new SolidBrush( Color.Red );
  Brush graybrush  = SystemBrushes.Control;
  Brush blackbrush = SystemBrushes.ControlText;
  Pen   redpen     = new Pen( Color.Red, 4 );
  public Form1()
  { Text = "XML1: Store & Read Polygones in XML-Format";
    Width = 800;
    Height = 600;
    g = this.CreateGraphics();
  }
  protected override void OnMouseDown( MouseEventArgs e )
  { polygon.Clear(); Invalidate();
    p0.X = e.X;
    p0.Y = e.Y;
    polygon.Add( p0 );
  }
  protected override void OnMouseMove( MouseEventArgs e )
  { if ( e.Button == MouseButtons.None ) return;
    p1.X = e.X;
    p1.Y = e.Y;
    Int32 dx = p1.X - p0.X;
    Int32 dy = p1.Y - p0.Y;
    if ( dx*dx + dy*dy < 100 ) return;
    g.DrawLine( redpen, p0, p1 );
    polygon.Add( p1 );
    p0 = p1;
  }
  protected override void OnMouseUp( MouseEventArgs e )
  { if ( polygon.Count < 2 ) return;
    polygon_string.Length = 0;
    polygon_string.Append( "\"" );
    for ( Int32 i=0; i < polygon.Count; i++ )
    { p0 = (Point)polygon[i];
      polygon_string.Append( p0.X );
      polygon_string.Append( "," );
      polygon_string.Append( p0.Y );
      if ( i < polygon.Count-1 ) polygon_string.Append( ", " );
    }
    polygon_string.Append( "\"" );
    Invalidate();
  }
  protected override void OnPaint( PaintEventArgs e )
  { g.FillRectangle( graybrush, 0, 0, Width, 2*Font.Height );
    g.DrawString( "Press the left mouse button and move!", Font, redbrush, Width/2-50, 0 );
    g.DrawString( polygon_string.ToString(), Font, blackbrush, 0, Font.Height );
    for ( Int32 i=0; i < polygon.Count-1; i++ )
      g.DrawLine( redpen, (Point)polygon[i], (Point)polygon[i+1] );
  }
}
```

Erproben Sie das Programm.

File-Menu

Version2: Beenden Sie Ihr Programm `xml1`.
Schreiben Sie folgende weiteren Befehle in den Konstruktor von `public Form1()` unterhalb der Zeile `Text = "XML1: Store & Read Polygons in XML-Format";`:

```
    MenuItem miSaveTXT  = new MenuItem( "SaveAsTXT ", new EventHandler(MenuFileSaveAsTXT ) );
    MenuItem miSaveXAML = new MenuItem( "SaveAsXAML", new EventHandler(MenuFileSaveAsXAML) );
    MenuItem miSaveSVG  = new MenuItem( "SaveAsSVG ", new EventHandler(MenuFileSaveAsSVG ) );
    MenuItem miRead     = new MenuItem( "&Read     ", new EventHandler(MenuFileRead) );
    MenuItem miExit     = new MenuItem( "&Exit     ", new EventHandler(MenuFileExit) );
    MenuItem miFile     = new MenuItem( "&File     ", new MenuItem[]
                                  { miSaveTXT, miSaveXAML, miSaveSVG, miRead, miExit} );
   Menu = new System.Windows.Forms.MainMenu( new MenuItem[] { miFile } );
```

und fünf neue Funktionsrümpfe direkt unterhalb der letzten geschweiften Klammer
der Funktion `protected override void OnPaint( PaintEventArgs e )`, aber noch vor der geschweiften Klammer,
die die Klasse `public class Form1 : System.Windows.Forms.Form` abschließt:

```
  void MenuFileSaveAsTXT( object obj, EventArgs ea )
  {
  }
  void MenuFileSaveAsXAML( object obj, EventArgs ea )
  {
  }
  void MenuFileSaveAsSVG( object obj, EventArgs ea )
  {
  }
  void MenuFileRead( object obj, EventArgs ea )
  {
  }
  void MenuFileExit( object obj, EventArgs ea )
  { Application.Exit(); }
```

Klicken Sie `Debug` → `Start Without Debugging Ctrl F5`. Erproben Sie das Menu.

Text-File schreiben

Version3: Beenden Sie Ihr Programm `xml1`. chreiben Sie folgenden Code in die leere Funktion `void MenuFileSaveAsTXT( object obj, EventArgs ea )`:

```
    SaveFileDialog dlg = new SaveFileDialog();
    dlg.Filter = "txt files (*.txt)|*.txt|All files (*.*)|*.*"  ;
    if ( dlg.ShowDialog() != DialogResult.OK ) return;
    Int32 n0 = dlg.FileName.IndexOf( ".txt" );
    if ( n0 < 0 ) dlg.FileName += ".txt";
    StreamWriter sw = new StreamWriter( dlg.FileName );
    sw.WriteLine( "This file comes from Prof. Miszalok's XML1.exe" );
    sw.WriteLine( "Polyline Points " );
    sw.WriteLine( polygon_string );
    sw.Close();
```

Klicken Sie `Debug` → `Start Without Debugging Ctrl F5`.
Malen Sie eine 1 und speichern Sie diese unter "`one.txt`", malen Sie eine 2 und speichern Sie diese unter "`two.txt`" usw.
(Die Extension `.txt` ist nicht obligatorisch, wenn sie fehlt, hängt `xml1` diese automatisch an Ihren Filenamen an.)
Kontrollieren Sie, ob Sie sich beim Speichern in der Directory `c:\temp\xml1\bin\debug` befinden.
Ansonsten müssen Sie `one.txt` später mühsam suchen.
Starten Sie `Notepad` oder `Editor` oder `Textpad`. Öffnen Sie das File `c:\temp\xml1\bin\debug\one.txt`.
Vergleichen Sie die Zeilen des Files mit dem Code von `void MenuFileSaveAsTXT( object obj, EventArgs ea )`.
Ändern Sie den String "`This file comes from Prof. Miszalok's XML1.exe`" übersetzen Sie neu und versuchen Sie alles noch einmal.
Starten Sie `My Computer` und gehen Sie in die Directory `c:\temp\xml1\bin\debug`. Doppelklicken Sie dort auf `one.txt`. Ihr Standardeditor (meistens `Notepad`) wird den Inhalt anzeigen.

XAML Browser Application schreiben

Version3: Beenden Sie Ihr Programm `xml1`. Schreiben Sie folgenden Code in die leere Funktion `void MenuFileSaveAsXAML( object obj, EventArgs ea )`:

```
    SaveFileDialog dlg = new SaveFileDialog();
    dlg.Filter = "XAML files (*.xaml)|*.xaml|All files (*.*)|*.*"  ;
    if ( dlg.ShowDialog() != DialogResult.OK ) return;
    Int32 n0 = dlg.FileName.IndexOf( ".xaml" );
    if ( n0 < 0 ) dlg.FileName += ".xaml";
    StreamWriter sw = new StreamWriter( dlg.FileName );
    sw.WriteLine( "<Canvas xmlns  =\"http://schemas.microsoft.com/winfx/2006/xaml/presentation\"" );
    sw.WriteLine( "       xmlns:x=\"http://schemas.microsoft.com/winfx/2006/xaml\">" );
    sw.WriteLine( "<!--This file comes from Prof. Miszalok's XML1.exe.-->" );
    sw.WriteLine( "<Polyline Points = " );
    sw.WriteLine( polygon_string + " Stroke=\"Red\" StrokeThickness=\"4\" />" );
    sw.WriteLine( "</Canvas>" );
    sw.Close();
```

Klicken Sie `Debug` → `Start Without Debugging Ctrl F5`.
Malen Sie eine 1 und speichern Sie diese unter "`one.xaml`", malen Sie eine 2 und speichern Sie diese unter "`two.xaml`" usw. (Die Extension `.xaml` ist nicht obligatorisch, wenn sie fehlt, hängt `xml1` diese automatisch an Ihren Filenamen an.)
Kontrollieren Sie, ob Sie sich beim Speichern in der Directory `c:\temp\xml1\bin\debug` befinden.
Ansonsten müssen Sie `one.xaml` später mühsam suchen.
Starten Sie `Notepad` oder `Textpad`. Öffnen Sie das File `c:\temp\xml1\bin\debug\one.xaml`.
Vergleichen Sie die Zeilen des Files mit dem Code von `void MenuFileSaveAsXAML(object obj, EventArgs ea)`.
Ändern Sie den String `This file comes from Prof. Miszalok's XML1.exe` in der `< title >`-Zeile im Code, übersetzen Sie neu und versuchen Sie alles noch einmal.
Starten Sie `My Computer` und gehen Sie in die Directory `c:\temp\xml1\bin\debug`.
Doppelklicken Sie dort auf `one.xaml`. Internet Explorer ab Version 7.0 interpretiert Ihr File und stellt es dar.
Frühere Versionen und Mozilla Firefox tun das nicht freiwillig. Man muss in diesem Fall ein Plug-In installieren:
http://www.microsoft.com/silverlight/install.aspx.

SVG-File schreiben

Version4: Beenden Sie Ihr Programm `xml1`. Schreiben Sie folgenden Code in die leere Funktion `void MenuFileSaveAsSVG( object obj, EventArgs ea )`:

```
    SaveFileDialog dlg = new SaveFileDialog();
    dlg.Filter = "svg files (*.svg)|*.svg|All files (*.*)|*.*"  ;
    if ( dlg.ShowDialog() != DialogResult.OK ) return;
    Int32 n0 = dlg.FileName.IndexOf( ".svg" );
    if ( n0 < 0 ) dlg.FileName += ".svg";
    StreamWriter sw = new StreamWriter( dlg.FileName );
    sw.WriteLine( "<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?>" );
    sw.WriteLine( "<svg xmlns=\"http://www.w3.org/2000/svg\" width=\"100%\" height=\"100%\">" );
    sw.WriteLine( "<title>This file comes from Prof. Miszalok's XML1.exe</title>" );
    sw.WriteLine( "<polyline fill=\"none\" stroke=\"red\" stroke-width=\"4\" points = " );
    sw.WriteLine( polygon_string );
    sw.WriteLine( " />" );
    sw.WriteLine( "</svg>" );
    sw.Close();
```

Klicken Sie `Debug` → `Start Without Debugging Ctrl F5`.
Malen Sie eine 1 und speichern Sie diese unter "`one.svg`", malen Sie eine 2 und speichern Sie diese unter "`two.svg`" usw. (Die Extension `.svg` ist nicht obligatorisch, wenn sie fehlt, hängt `xml1` diese automatisch an Ihren Filenamen an.)
Kontrollieren Sie, ob Sie sich beim Speichern in der Directory `c:\temp\xml1\bin\debug` befinden.
Ansonsten müssen Sie `one.svg` später mühsam suchen.
Starten Sie `Notepad` oder `Textpad`. Öffnen Sie das File `c:\temp\xml1\bin\debug\one.svg`.
Vergleichen Sie die Zeilen des Files mit dem Code von `void MenuFileSaveAsSVG( object obj, EventArgs ea )`.
Ändern Sie den String `This file comes from Prof. Miszalok's XML1.exe` in der `< title >`-Zeile im Code, übersetzen Sie neu und versuchen Sie alles noch einmal.
`Internet Explorer:` Downloaden Sie den Adobe SVG Viewer von **www.adobe.com/svg/viewer/install** (2.3 MByte) und installieren Sie diesen. Mozilla Firefox und Opera brauchen kein Plugin.
Starten Sie `My Computer` und gehen Sie in die Directory `c:\temp\xml1\bin\debug`. Doppelklicken Sie dort auf `one.svg`. Falls Sie eine leere Seite sehen, hat Ihr Internet Explorer kein Plugin oder Ihr Firefox oder Opera sind alt.
Lesen Sie: **www.mediaevent.de**

Text-, XAML- und SVG-Files lesen

Version5: Beenden Sie Ihr Programm `xml1`. Schreiben Sie folgenden Code in die leere Funktion `void MenuFileRead( object obj, EventArgs ea )`:

```
    OpenFileDialog dlg = new OpenFileDialog();
    dlg.Filter = "All files (*.*)|*.*";
    if ( dlg.ShowDialog() != DialogResult.OK ) return;
    StreamReader sr = new StreamReader( dlg.FileName );
    String file_string = sr.ReadToEnd();
    sr.Close();
    Int32 n0 = file_string.IndexOf( "olyline"     ); if ( n0 < 0 ) return;
    Int32 n1 = file_string.IndexOf( "oints", n0+7 ); if ( n1 < 0 ) return;
    Int32 n2 = file_string.IndexOf( "\""    , n1+5 ); if ( n2 < 0 ) return;
    Int32 n3 = file_string.IndexOf( "\""    , n2+1 ); if ( n3 < 0 ) return;
    String all_coordinates_string = file_string.Substring( n2+1, n3-n2-1 );
    string[] coordinates_string_array = all_coordinates_string.Split(',');
    polygon.Clear();
    for ( Int32 i=0; i < coordinates_string_array.Length; i+=2 )
    {  p1.X = Convert.ToInt32( coordinates_string_array[i  ] );
       p1.Y = Convert.ToInt32( coordinates_string_array[i+1] );
       polygon.Add( p1 );
    }
    Invalidate();
```

Klicken Sie `Debug` → `Start Without Debugging Ctrl F5`.
Sie können jetzt Ihre gespeicherten Files mit `xml1` lesen und anzeigen.

Weitere Aufgaben

Klicken Sie auf `Help` in der Menüleiste von Visual Studio. Klicken Sie auf das Untermenü `Index`.
Gehen Sie in das Feld `Filtered by:` und wählen Sie dort `.NET Framework SDK`.
Dann geben Sie im Feld `Look for:` folgende Schlüsselwörter ein: `StringBuilder, ArrayList, MainMenu, FileDialog, SaveFileDialog, OpenFileDialog, StreamWriter, StreamReader, Convert, String class, methods: IndexOf` und `Split` etc. Lesen Sie die Hilfetexte.
Die Hilfetexte überdecken Ihren Code. Wenn Sie mit dem lesen fertig sind, entfernen Sie die Hilfetexte mit deren X-Button in deren rechter oberer Fensterecke.
Erfinden und erproben Sie neue Varianten des Programms in Form von neuen Projekten `xml2, xml3` usw. nach obigem Muster.
Einfach: Ändern Sie Farbe und Pinseldicke in `Form1` und in `void MenuFileSaveAsXAML( object obj, EventArgs ea )`, so dass beide zu einander passen.
Nicht einfach: Versuchen Sie eine Version zu schreiben mit einer übergeordneten `ArrayList`, die Unter-`ArrayListen` `polygon[i]` enthält, um mehrere Polygone gleichzeitig zu verwalten, zu schreiben und zu lesen.

Animieren Sie Ihre XAML Browser Application !

Beispiel 1: Drehanimation:

Laden Sie eines Ihrer XAML-Files in Textpad und erweitern Sie dieses nach folgendem Muster:

```
<Canvas xmlns  ="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
<!--This file comes from Prof. Miszalok's XML1.exe.-->
<Polyline Points =
"127,97, 125,87, 131,79, 139,73, 149,70, 160,74, 169,85, 176,93, 183,108, 185,121, 181,136,
176,152, 171,170, 167,181, 162,192, 158,202, 149,220, 140,237, 132,248, 124,254, 115,262,
107,268, 97,277, 89,285, 85,295, 83,305, 82,315, 93,315, 110,316, 126,316, 145,316,
165,316, 176,316, 192,316, 203,316, 213,317, 224,317"
Stroke="Red" StrokeThickness="4"> <!--Caution: No backslash!-->
<Polyline.RenderTransform>
  <RotateTransform x:Name="mytrans" CenterX="200" CenterY="200"/>
</Polyline.RenderTransform>
<Polyline.Triggers>
  <EventTrigger RoutedEvent="Polyline.Loaded">
    <BeginStoryboard>
      <Storyboard TargetName="mytrans" TargetProperty="Angle" RepeatBehavior="Forever">
        <DoubleAnimation From="0" To="360" Duration="0:0:2"/>
      </Storyboard>
    </BeginStoryboard>
  </EventTrigger>
</Polyline.Triggers>
</Polyline>
</Canvas>
```

Beispiel 2: ± ZoomX Animation: Ändern Sie 3 Zeilen von Beispiel 1.

```
alt: <RotateTransform x:Name="mytrans" CenterX="200" CenterY="200"/>
neu: <ScaleTransform  x:Name="mytrans" CenterX="200" CenterY="200"/>
alt: <Storyboard TargetName="mytrans" TargetProperty= "Angle" RepeatBehavior="Forever">
neu: <Storyboard TargetName="mytrans" TargetProperty="ScaleX" RepeatBehavior="Forever"
                                                              AutoReverse="True">
alt: <DoubleAnimation From="0" To="360" Duration="0:0:2"/>
neu: <DoubleAnimation From="-1" To= "1" Duration="0:0:2"/>
```

Beispiel 3: ± ZoomY Animation: Ändern Sie in Beispiel 2 einen einzigen Buchstaben:

```
alt: TargetProperty="ScaleX"
neu: TargetProperty="ScaleY"
```

Lesen Sie: **www.xaml.net** und **http://msdn2.microsoft.com/en-us/library/bb404703.aspx**.
Lesen Sie: **http://de.wikipedia.org/wiki/Scalable_Vector_Graphics**.
Lesen Sie: **www.adobe.com/svg/overview/svg.htm** ff.

Miniprogramme zum Training

MiniXML1: Zugriff auf eine Textfile

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.IO;
public class Form1 : Form
{ [STAThread] static void Main() { Application.Run( new Form1() ); }
  public Form1()
  { Text = "Click the Form !";
    StreamWriter sw = new StreamWriter( "C:\\temp\\test.txt" );
    sw.WriteLine( "That's a text" );
    sw.Close();
    Click += new System.EventHandler( Form1_Click );
  }
  private void Form1_Click(object sender, EventArgs e)
  { StreamReader sr = new StreamReader( "C:\\temp\\test.txt" );
    Graphics g = CreateGraphics();
    g.DrawString( "File content: " + sr.ReadToEnd(), new Font( "Arial", 10 ),
SystemBrushes.ControlText, 0, 0 );
    sr.Close();
  }
}
```

MiniXML2: schreiben und lesen einer Zeile

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

public class Form1 : Form
{ [STAThread] static void Main() { Application.Run( new Form1() ); }
  private Button button1   = new Button();
  private Button button2   = new Button();
  private TextBox textBox1 = new TextBox();
  private TextBox textBox2 = new TextBox();
  public Form1()
  { Width = 340; Height = 120;
    button1.Location  = new Point( 10, 20); button1 .Width = 100;
    button2.Location  = new Point( 10, 60); button2 .Width = 100;
    textBox1.Location = new Point(120, 20); textBox1.Width = 200;
    textBox2.Location = new Point(120, 60); textBox2.Width = 200;
    button1.Text = "WriteToFile";
    button2.Text = "ReadFromFile";
    button1.Click += new EventHandler( button1_Click );
    button2.Click += new EventHandler( button2_Click );
    Controls.Add( button1 );
    Controls.Add( button2 );
    Controls.Add( textBox1 );
    Controls.Add( textBox2 );
  }
  private void button1_Click( object sender, EventArgs e )
  { StreamWriter sw = new StreamWriter( "C:\\temp\\test.txt" );
    sw.WriteLine( textBox1.Text );
    sw.Close();
  }
  private void button2_Click( object sender, EventArgs e )
  { StreamReader sr = new StreamReader( "C:\\temp\\test.txt" );
    textBox2.Text = sr.ReadToEnd();
    sr.Close();
  }
}
```

Visual Programming:
Sie können ein Programm mit gleicher Funktionalität wie `MiniXML2` auch
automatisch von Visual Studio 2005 und Visual C# Express generieren lassen.
1. Öffnen Sie ein neues Projekt `MiniXML2Automatic`.
2. Hauptmenu `View` → `Toolbox`. Ziehen Sie nun mit der Maus zweimal einen Button und dann zweimal eine Textbox aus der Toolbox auf `Form1`.
3. Doppelklicken Sie auf `button1` und kehren Sie auf `Form1.cd[Design]*` zurück.
4. Doppelklicken Sie auf `button2`.
5. Schreiben Sie in die Funktion `private void button1_Click(object sender, EventArgs e)` die gleichen 3 Zeilen wie in `MiniXML2`.
6. Schreiben Sie in die Funktion `private void button2_Click(object sender, EventArgs e)` die gleichen 3 Zeilen wie in `MiniXML2`.
7. `Debug` → `Start Without Debugging`

Den gesamten automatisch generierten Code sehen Sie, wenn Sie auf das +-Zeichen vor der grauen Zeile "`Windows Form designer generated code`" klicken.
Obwohl funktionell identisch, ist er erheblich länger, komplizierter und unübersichtlicher als der von `MiniXML2`.

Profis benutzen kein "Visual Programming" weil:
1. WPF (allerdings noch sehr neu) ist viel leistungsfähiger als Visual Programming.
2. Der Code ist lang und enthält viel Redundanz.
3. Die Controls haben feste Größe und Position und reagieren nicht auf Größenänderungen des Fensters.
4. Die Controls verhalten sich verschieden und kooperieren nicht.

Windows Presentation Foundation WPF:
WPF ist die modernste Art der Programmierung von Benutzeroberflächen = User Interface UI.
WPF formuliert Zahl, Form, Lage und Verhalten aller Controls (= Buttons, TextBoxes etc.) mit Hilfe des Designerwerkzeugs **Microsoft Expression** in der XML-Sprache XAML,
der neuen Markup-Language zur Gestaltung von Benutzeroberflächen.
Visual Studio 2008 enthält WPF.