

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

Chapter C3: The XML Project

Copyright © by V. Miszalok, last update: 09-12-2007

- ↓ [Project xml1 with an empty window](#)
- ↓ [Scribble program with dynamic array](#)
- ↓ [File menu](#)
- ↓ [Write a text file](#)
- ↓ [Write a XAML Browser Application](#)
- ↓ [Write a SVG file](#)
- ↓ [Read TXT, XAML and SVG files](#)
- ↓ [Exercises](#)
- ↓ [Mini programs for training](#)

Project xml1 with an empty window

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

1) Main Menu after start of VS 2008: `File` → `New Project...` → `Visual Studio installed templates: Windows Forms Application`
 Name: `xml1` → Location: `C:\temp` → `Create directory for solution: switch off` → `OK`
`Form1.cs[Design]` appears.

2) Two superfluous files must be deleted: `Form1.Designer.cs` and `Program.cs`.
 You reach these files via the `Solution Explorer` - `xml1-window`:
 Click the plus-sign in front of branch `xml1` and the plus-sign in front of branch `Form1.cs`.
 Right-click the branch `Program.cs`. A context menu opens. Click `Delete`. A message box appears: `'Program.cs' will be deleted permanently. Quit with OK.`
 Right-click the branch `Form1.Designer.cs` and delete this file too.

3) Right-click the gray window `Form1`. A small context menu opens. Click `View Code`.
 You see now the preprogrammed code of `Form1.cs`. Erase this code completely.

4) Write the following three lines into the empty `Form1.cs`:

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

5) Click `Debug` in the main menu of VS 2005.
 A submenu opens. Click `Start Without Debugging Ctrl F5`.

Important: Always finish all instances of `xml1` before writing new code and starting it !
Start the Task Manager with `Ctrl+Alt+Del` and check if any `xml1.exe`-process is still running.
If yes, kill it.

Scribble program with dynamic array

If You don't see Your code anymore, click the tab **Form1.cs** of Visual Studio.

Delete everything (including the last brace). No code remains.

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

```
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 Polygons 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 == MouseButton.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] );
  }
}
```

Click Debug → Start Without Debugging Ctrl F5. Try out the program.

File menu

Version2: Finish xml1.

Insert further lines into the constructor of `public Form1()`

below the line `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 } );
```

Insert five new empty function calls directly below the last brace of

function protected override `void OnPaint(PaintEventArgs e)`

but in front of the brace, which closes `public class Form1 : System.Windows.Forms.Form:`

```
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(); }
```

Click Debug → Start Without Debugging Ctrl F5. Try out the menu.

Write a text file

Version3: Finish xml1.

Write the following code into the empty function

`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();
```

Click Debug → Start Without Debugging Ctrl F5.

Draw a "1" and store it as "one.txt", Draw a "2" and store it as "two.txt", etc.

(The extension .txt is not obligatory, xml1 appends it automatically if necessary.)

Please make sure that You store into the directory `c:\temp\xml1\bin\debug`.

(Otherwise You have to look for one.txt later on.)

Start Notepad or Editor or Textpad. Open file `c:\temp\xml1\bin\debug\one.txt`.

Compare the lines of one.txt with the code of `void MenuFileSaveAsTXT(object obj, EventArgs ea)`.

Change the string "This file comes from Prof. Miszalok's XML1.exe" compile and open one.txt again.

Start My Computer and enter directory `c:\temp\xml1\bin\debug`. Double click one.txt.

Your standard editor (mostly Notepad) will display the content.

Write a XAML Browser Application

Version3: Finish xml1.

Write the following code into the empty function

```
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();
```

Click Debug → Start Without Debugging Ctrl F5.

Draw a "1" and store it as "one.html", Draw a "2" and store it as "two.html", etc. (The extension .xaml is not obligatory, xml1 appends it automatically if necessary.)

Please make sure, that You store into the directory c:\temp\xml1\bin\debug. (Otherwise You have to look for one.html later on.)

Start Notepad or Textpad. Open file c:\temp\xml1\bin\debug\one.xaml.

Compare the lines of one.xaml with the code of void MenuFileSaveAsXAML(object obj, EventArgs ea).

Change the string "This file comes from Prof. Miszalok's XML1.exe" compile and open one.xaml again.

Start My Computer and enter directory c:\temp\xml1\bin\debug. Double click one.xaml. Internet Explorer vers. 7.0 will interpret and display the file. Older versions and Mozilla Firefox don't interpret xaml. In this case You have to install a Plug-In: www.microsoft.com/downloads/details.aspx?FamilyId=E63992D3-CCF5-40B9-B98A-D16BCA57467C&displaylang=en.

Write a SVG file

Version4: Finish xml1.

Write the following code into the empty function 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();
```

Click Debug → Start Without Debugging Ctrl F5.

Draw a "1" and store it as "one.svg", Draw a "2" and store it as "two.svg", etc.

(The extension .svg is not obligatory, xml1 appends it automatically if necessary.)

Please make sure, that You store into the directory c:\temp\xml1\bin\debug.

(Otherwise You have to look for one.svg later on.)

Start Notepad or Editor or Textpad. Open file c:\temp\xml1\bin\debug\one.svg.

Compare the lines of one.svg with the code of void MenuFileSaveAsSVG(object obj, EventArgs ea).

Change the string "This file comes from Prof. Miszalok's XML1.exe" compile and open one.svg again.

Internet Explorer: Download Adobe SVG Viewer from www.adobe.com/svg/viewer/install (2.3 MByte) and install it. Current versions of Firefox and Opera need no plugin.

Start My Computer and enter directory c:\temp\xml1\bin\debug. Double click one.svg. If You just see an empty page, Your Internet Explorer has no plugin or You run old versions of Firefox or Opera.

Read TXT, XAML and SVG files

Version5: Finish xml1.

Write the following code into function 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();

```

Click Debug → Start Without Debugging Ctrl F5. Try to read and display Your stored *.txt, *.xaml and *.svg files.

Exercises

Click Help in the main menu of Visual Studio. Click the sub-menu Index.

Go to Filtered by: and choose: .NET Framework SDK. Then enter into Look for: one of the following key words: StringBuilder, ArrayList, MainMenu, FileDialog, SaveFileDialog, OpenFileDialog, StreamWriter, StreamReader, Convert, String class, methods: IndexOf and Split etc. You obtain a selection of key words beginning with those characters. Read the help texts. The help window covers Your code. When you finished reading, get rid of it with the X-button in the upper right window corner.

Invent and try out new versions in form of new projects xml2, xml3.

Simple: Change color and pen thickness in Form1 and in

void MenuFileSaveAsxaml(object obj, EventArgs ea), so that both fit together.

Not so simple: Try to write a version with a superior ArrayList, containing sub-ArrayLists polygon[i] in order to administrate, write and read more than one polygon.

Animate Your XAML Browser Application !

Sample 1: Rotation Animation: Load one of Your XAML-files in Textpad and expand it as follows:

```

<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>

```

Sample 2: ± ZoomX Animation: Change 3 lines of Sample 1.

```
old: <RotateTransform x:Name="mytrans" CenterX="200" CenterY="200"/>
new: <ScaleTransform x:Name="mytrans" CenterX="200" CenterY="200"/>
old: <Storyboard TargetName="mytrans" TargetProperty="Angle" RepeatBehavior="Forever">
new: <Storyboard TargetName="mytrans" TargetProperty="ScaleX" RepeatBehavior="Forever"
      AutoReverse="True">

old: <DoubleAnimation From="0" To="360" Duration="0:0:2"/>
new: <DoubleAnimation From="-1" To="1" Duration="0:0:2"/>
```

Sample 3: ± ZoomY Animation: Change one single character of Sample 2:

```
old: TargetProperty="ScaleX"
new: TargetProperty="ScaleY"
```

Link: www.xaml.net und <http://msdn2.microsoft.com/en-us/library/bb404703.aspx>.

Link: http://de.wikipedia.org/wiki/Scalable_Vector_Graphics.

Link: www.adobe.com/svg/overview/svg.htm ff.

Mini programs for training

MiniXML1: Access a text file

```
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: Write and read one line

```
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:

Both Visual Studio and Visual C# Express can generate automatically a complete program having the same functionality as `MiniXML2`.

1. Open a new project `MiniXML2Automatic`.
 2. Main menu `View` → `Toolbox`. Drag twice a `Button` and twice a `Textbox` from the `Toolbox` to `Form1`.
 3. Double click `button1` and return to `Form1.cd[Design]*`.
 4. Double click `button2`.
 5. Write into function `private void button1_Click(object sender, EventArgs e)` the same three lines as You did in `MiniXML2`.
 6. Write into function `private void button2_Click(object sender, EventArgs e)` the same three lines as You did in `MiniXML2`.
 7. `Debug` → `Start Without Debugging`
- You see the complete automatically generated code, when You click onto the ++-sign in front of the gray lines "Windows Form designer generated code". This code is functionally identical, but it is longer, more complicated and more confusing than the code of `MiniXML2`.

Professionals don't like "Visual Programming" because of:

1. WPF (which is quite new) is more powerful than Visual Programming
2. The code is full of stupid redundancy.
3. The controls have fixed size and position and do not react when the window gets resized.
4. The controls do not behave in the same way and do not cooperate.

Windows Presentation Foundation WPF:

WPF is the most modern way to generate user interfaces UI.

WPF defines the number, design, position and behavior of all controls (= Buttons, TextBoxes etc.) using a designer tool **Microsoft Expression** in the XML-language XAML, the new markup-language to define rich and powerful UIs.

Visual Studio 2008 contains WPF.