Course 2D_WPF: 2D-Computer Graphics with C# + WPF
Chapter C1a: The Intro Project Written in XAML and C#

An Empty Window

Guidance for **Visual C# 2010 Express**, see **Introduction into all Courses**.

1) Main Menu after start of VC# 2010: `Tools → Options →`
check lower left checkbox: `Show all Settings → Projects and Solutions →`
`Visual Studio projects location: → C:\temp`

2) Main Menu after start of VC# 2010: `File → New Project... →`
`Installed templates: WPF Application → Name: intro1_XAML_CS → OK.`

3) Stop the automatic code formatter before it will drive you crazy.
Main menu of VC# 2010 → `Tools` → `Options...` → An `Options`-window appears.
Double-click the branch `Text Editor`. **Double**-click `C#`. **Double**-click `Formatting`.
Click `General`. Uncheck all three check boxes.
Click `Indentation`. Uncheck all four check boxes.
Click `New Lines`. Uncheck all thirteen check boxes.
Click `Spacing`. Uncheck all twenty three check boxes.
Click `IntelliSense`. Uncheck all six check boxes.
Quit the `Options`- window with the `OK`-button.

4) Click `Debug` in the main menu of VC# 2010.
A submenu opens. Click `Start Debugging F5`.

The default program code automatically compiles, links and starts.
Our program starts automatically as stand-alone window containing three parts:
- main window = `MainWindow` with a blue title row
- three buttons `Minimize, Maximize, Close`
- a narrow frame with 4 movable borders and 4 movable edges.
  Enlarge and shrink the window by dragging its borders and edges.
- Kill `MainWindow` by clicking the close button in its right upper edge.

Build the project: `Debug → Build Solution F6`.
Save the project: `File → Save All Ctrl+Shift+S → Name: intro1_XAML_CS →`
`Location: C:\temp →` uncheck `Create directory for solution → Save`.
Minimize VC# 2010.
`intro1_XAML_CS.exe` can now be started as a stand-alone windows program as follows:
- Look for `C:\temp\intro1_XAML_CS\bin\Release.`
- Double click `intro1_XAML_CS.exe`.
- You can start an arbitrary number of instances of `intro1_XAML_CS.exe`.
- (Caution: You will have kill all instances before You can write new versions.)

Important: Always finish all instances of `intro1_XAML_CS` before writing new code and starting it !
Start the Task Manager with `Ctrl+Alt+Del` and check if any `intro1_XAML_CS.exe`-processes are still running and kill them.

Hallo World

If there is no Solution Explorer-window, open it via the main menu of VC# 2010 → View → Solution Explorer.

1. Click the **MainWindow.xaml**-branch inside the Solution Explorer-window and replace its complete default code by these lines:

```
<Window x:Class="intro1_XAML_CS.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="A first WPF-Program written in XAML and C#" Height="300" Width="300"
        Top="50" Left="50" Background="Cornsilk" Foreground="Red"
        Content="Hello World ">
</Window>
```

2. Click the **MainWindow.xaml.cs**-branch below the **MainWindow.xaml**-branch inside the Solution Explorer-window and replace its complete default code by these lines:

```
using System;
using System.Windows;

namespace intro1_XAML_CS
{ public partial class MainWindow:Window
  { public MainWindow()
    { InitializeComponent();
      Content += DateTime.Now.ToString();
    }
  }
}
```

Click the Debug-tab of the main menu obove the main window.
A sub-menu opens. Click Start Debugging F5.
The program automatically compiles, links and starts again.

Exercises: Try out other start positions, other window sizes, other strings, other brushes and other colors.
Important Tip: In case of mistype, VC# 2010 presents a Message Box: There were build errors. ....
You quit with **No**. An Error List-window with warnings and errors will appear in Visual Studio below Your program. In this error list scroll up to the first error (ignore the warnings !). Double click the line with the first error. The cursor jumps automatically into Your code into the line where the error was detected. Look for mistypes in this line and remove them. (Sometimes You will not find the error in this line but above, where You forgot a comma or a semicolon.) Ignore all errors below the first error (in most cases they are just followers of the first one) and compile. Repeat this procedure until further error message boxes disappear and Your program compiles, links and starts as expected.

Window Size

Version 2: Finish all instances of `intro1_XAML_CS` and replace the complete codes of version 1 by:

MainWindow.xaml:
```
<Window x:Class="intro1_XAML_CS.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="A first WPF-Program written in XAML and C#" Height="300" Width="300"
        Top="50" Left="50" SizeChanged="Window_SizeChanged" >
 <DockPanel Name="myPanel">
  <TextBox Name="centerText" HorizontalAlignment="Center" VerticalAlignment="Center"
           TextAlignment="Center"/>
 </DockPanel>
</Window>
```
MainWindow.xaml.cs:
```
using System;
using System.Windows;
using System.Windows.Media;
using System.Windows.Controls;
using System.Windows.Threading;

namespace intro1_XAML_CS
{ public partial class MainWindow:Window
  { public MainWindow()
    { InitializeComponent();
    }
    private void Window_SizeChanged( object sender, SizeChangedEventArgs e )
    { //Compose the text of the "centerText"-TextBox:
      String s1 = "Hello World " + DateTime.Now.ToString() + "\n";
      int width  = Convert.ToInt32( this.Width  );
      int height = Convert.ToInt32( this.Height );
      String s2 = "Window Size = " + width.ToString() + " x " + height.ToString();
      centerText.Text = s1 + s2;
    }
  }
}
```

Click `Debug` in the main menu above the main window and `Start Debugging F5`.
Drag the window border and observe the content of the text box.

Left, right, top, bottom

Version 3: Finish all instances of `intro1_XAML_CS` and replace the object `<DockPanel Name="myPanel">` by this one:

MainWindow.xaml:
```
 <DockPanel Name="myPanel">
  <TextBox Name="left"    Text="left"   VerticalAlignment  ="Center" DockPanel.Dock="Left"   />
  <TextBox Name="right"   Text="right"  VerticalAlignment  ="Center" DockPanel.Dock="Right"  />
  <TextBox Name="top"     Text="top"    HorizontalAlignment="Center" DockPanel.Dock="Top"    />
  <TextBox Name="bottom"  Text="bottom" HorizontalAlignment="Center" DockPanel.Dock="Bottom" />
  <TextBox Name="centerText"            HorizontalAlignment="Center" VerticalAlignment="Center"/>
 </DockPanel>
```

Click `Debug` and `Start Debugging F5`.
Drag the window border and observe how the text boxes follow.

Line, Rectangle, Ellipse

Version 4a: Finish `intro1_XAML_CS` and replace the complete `<DockPanel Name="myPanel">`-object by a `Canvas`-object.
(It will be needed in the next version 4b to draw 2 lines, a rectangle and an ellipse.)
Just for fun let's fill up the background of the canvas with a 3-color-brush.

```
 <Canvas Name ="myCanvas">
  <Canvas.Background>
   <LinearGradientBrush>
    <GradientStop Color="Blue"  Offset="0"  />
    <GradientStop Color="Green" Offset="0.5"/>
    <GradientStop Color="Red"   Offset="1"  />
   </LinearGradientBrush>
  </Canvas.Background>
  <DockPanel Name="myPanel">
   <TextBox Name="left"    Text="left"   VerticalAlignment  ="Center" DockPanel.Dock="Left"   />
   <TextBox Name="right"   Text="right"  VerticalAlignment  ="Center" DockPanel.Dock="Right"  />
   <TextBox Name="top"     Text="top"    HorizontalAlignment="Center" DockPanel.Dock="Top"    />
   <TextBox Name="bottom"  Text="bottom" HorizontalAlignment="Center" DockPanel.Dock="Bottom" />
   <TextBox Name="centerText"           HorizontalAlignment="Center"
VerticalAlignment="Center"/>
  </DockPanel>
 </Canvas>
```

In **`MainWindow.xaml.cs`** in `private void Window_SizeChanged( ... )`
add these four lines below `centerText.Text = s1 + s2;`:

```
      //myCanvas automatically resizes to the new window size, but its content
      //myPanel doesn't and must be forced to resize to full myCanvas-size::
      myPanel.Width  = myCanvas.ActualWidth;
      myPanel.Height = myCanvas.ActualHeight;
```

Click `Debug` and `Start Debugging F5`.
Drag the window border and notice that the new `Canvas myCanvas` is still invisible and
that Version 4a looks and behaves exactly as Version 3 did.
****

Version 4b: Finish `intro1_XAML_CS`.
In **`MainWindow.xaml`** add two lines, a rectangle and an ellipse to `<Canvas Name ="myCanvas">` until **`MainWindow.xaml`** looks like this:

```
<Window x:Class="intro1_XAML_CS.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="A first WPF-Program written in XAML and C#" Height="300" Width="300"
        Top="50" Left="50" SizeChanged="Window_SizeChanged">
 <Canvas Name="myCanvas">
  <Canvas.Background>
   <LinearGradientBrush>
    <GradientStop Color="Blue"  Offset="0"  />
    <GradientStop Color="Green" Offset="0.5"/>
    <GradientStop Color="Red"   Offset="1"  />
   </LinearGradientBrush>
  </Canvas.Background>
  <Line      Name="line1" Stroke="Black" StrokeThickness="3"/>
  <Line      Name="line2" Stroke="Black" StrokeThickness="3"/>
  <Rectangle Name="rect"  Stroke="Black" StrokeThickness="3" Fill="White"/>
  <Ellipse   Name="elli"  Stroke="Black" StrokeThickness="3"/>

  <DockPanel Name="myPanel">
   <TextBox Name="left"    Text="left"   VerticalAlignment  ="Center" DockPanel.Dock="Left"   />
   <TextBox Name="right"   Text="right"  VerticalAlignment  ="Center" DockPanel.Dock="Right"  />
   <TextBox Name="top"     Text="top"    HorizontalAlignment="Center" DockPanel.Dock="Top"    />
   <TextBox Name="bottom"  Text="bottom" HorizontalAlignment="Center" DockPanel.Dock="Bottom" />
   <TextBox Name="centerText"           HorizontalAlignment="Center" VerticalAlignment="Center"
            TextAlignment="Center"/>
  </DockPanel>
 </Canvas>
</Window>
```

In **MainWindow.xaml.cs** change `private void Window_SizeChanged( ... )`
until it looks like this:
Do not touch the 2 lowermost braces of **MainWindow.xaml.cs** !

```
    private void Window_SizeChanged( object sender, SizeChangedEventArgs e )
    { //compose the text of the "centerText"-TextBox
      String s1 = "Hello World " + DateTime.Now.ToString() + "\n";
      int width  = Convert.ToInt32( this.Width  );
      int height = Convert.ToInt32( this.Height );
      String s2 = "Window Size = " + width.ToString() + " x " + height.ToString() + "\n";
      width  = Convert.ToInt32( myCanvas.ActualWidth  );
      height = Convert.ToInt32( myCanvas.ActualHeight );
      String s3 = "Client Size = " + width.ToString() + " x " + height.ToString() + "\n";
      String s4 = String.Format( "Font Size = {0,2:F1}", centerText.FontSize );
      centerText.Text = s1 + s2 + s3 + s4;
      //myCanvas automatically resizes to the new window size,
      //but its content doesn't and must be forced to resize to full myCanvas-size:
      line1.X1 = 0; line1.Y1 = 0; line1.X2 = myCanvas.ActualWidth;
                                  line1.Y2 = myCanvas.ActualHeight;
      line2.X1 = myCanvas.ActualWidth; line2.Y1 = 0;
      line2.X2 = 0; line2.Y2 = myCanvas.ActualHeight;
      Canvas.SetLeft( rect, myCanvas.ActualWidth /5 ); //20% empty left space
      Canvas.SetLeft( elli, myCanvas.ActualWidth /5 ); //20% empty left space
      Canvas.SetTop ( rect, myCanvas.ActualHeight/5 ); //20% empty top  space
      Canvas.SetTop ( elli, myCanvas.ActualHeight/5 ); //20% empty top  space
      rect .Width  = elli.Width  = 3 * myCanvas.ActualWidth  / 5; //width  = 60%
      rect .Height = elli.Height = 3 * myCanvas.ActualHeight / 5; //height = 60%
      myPanel.Width  = myCanvas.ActualWidth;
      myPanel.Height = myCanvas.ActualHeight;
    }
```

Click `Debug` and `Start Debugging F5`.
Resize the program window.

Window Size Animation

Version5: Finish `intro1_XAML_CS`.
Declare and initialize a global variable `double zoom = 1.1;` in the head of
`public partial class MainWindow : Window`, start a `DispatcherTimer`-object in the constructor and write its event handler "`private void TimerOnTick(...)`".
Replace the complete code of **MainWindow.xaml.cs** by :

```
using System;
using System.Windows;
using System.Windows.Media;
using System.Windows.Controls;
using System.Windows.Threading;

namespace intro1_XAML_CS
{ public partial class MainWindow:Window
  { double zoom = 1.1;
    public MainWindow()
    { InitializeComponent();
      //myTimer is a clock intended to animate the window size
      DispatcherTimer myTimer = new DispatcherTimer();
      myTimer.Interval = TimeSpan.FromMilliseconds( 1 );
      myTimer.Tick += TimerOnTick;
      myTimer.Start();
    }
    private void TimerOnTick( Object sender, EventArgs args )
    { if ( this.ActualWidth < 200 ) zoom = 1.1;  //fast enlargement
      if ( this.ActualWidth > 800 ) zoom = 0.99; //slow shrinking
      this.Width  *= zoom;
      this.Height *= zoom;
      centerText.FontSize *= zoom;
    }
```

```
    private void Window_SizeChanged( object sender, SizeChangedEventArgs e )
    { //compose the text of the "centerText"-TextBox
      String s1 = "Hello World " + DateTime.Now.ToString() + "\n";
      int width  = Convert.ToInt32( this.Width  );
      int height = Convert.ToInt32( this.Height );
      String s2 = "Window Size = " + width.ToString() + " x " + height.ToString() + "\n";
      width  = Convert.ToInt32( myCanvas.ActualWidth  );
      height = Convert.ToInt32( myCanvas.ActualHeight );
      String s3 = "Client Size = " + width.ToString() + " x " + height.ToString() + "\n";
      String s4 = String.Format( "Font   Size = {0,2:F1}", centerText.FontSize );
      centerText.Text = s1 + s2 + s3 + s4;
      //adjust all contents of "MainWindow" to its new window size
      line1.X1 = 0; line1.Y1 = 0; line1.X2 = myCanvas.ActualWidth;
                                  line1.Y2 = myCanvas.ActualHeight;
      line2.X1 = myCanvas.ActualWidth; line2.Y1 = 0;
      line2.X2 = 0; line2.Y2 = myCanvas.ActualHeight;
      Canvas.SetLeft( rect, myCanvas.ActualWidth /5 );
      Canvas.SetLeft( elli, myCanvas.ActualWidth /5 );
      Canvas.SetTop ( rect, myCanvas.ActualHeight/5 );
      Canvas.SetTop ( elli, myCanvas.ActualHeight/5 );
      rect .Width  = elli.Width  = 3 * myCanvas.ActualWidth  / 5;
      rect .Height = elli.Height = 3 * myCanvas.ActualHeight / 5;
      myPanel.Width  = myCanvas.ActualWidth;
      myPanel.Height = myCanvas.ActualHeight;
    }
  }
}
```

Click Debug and Start Debugging F5.
Observe the content of the central text box.

Ellipse Animation

Version 5: Finish intro1_XAML_CS.
In **MainWindow.xaml** replace the line <Ellipse Name="elli" Stroke="Black" StrokeThickness="3"> by:

```
  <Ellipse   Name="elli"  Stroke="Black" StrokeThickness="3">
   <Ellipse.Fill>
    <RadialGradientBrush x:Name="elliBrush" RadiusX="0.1" RadiusY="0.1" SpreadMethod="Repeat">
     <GradientStop x:Name="stop1" Offset="0"  />
     <GradientStop x:Name="stop2" Offset="0.5"/>
     <GradientStop x:Name="stop3" Offset="1  "/>
    </RadialGradientBrush>
   </Ellipse.Fill>
  </Ellipse>
```

In **MainWindow.xaml.cs** write four additional declarations in the head of public partial class MainWindow:Window below the line double zoom = 1.1;:

```
    double angle = 0;
    Random r = new Random();
    Byte r1, g1, b1, r2, g2, b2, r3, g3, b3;
    Point radialGradientBrushOrigin = new Point( 0.5, 0.5 );
```

In the constructor public MainWindow() insert two new lines below the line myTimer.Start();:

```
      //initial random colors of RadialGradientBrush "elliBrush"
      r1 = (Byte)r.Next(255); g1 = (Byte)r.Next(255); b1 = (Byte)r.Next(255);
      r2 = (Byte)r.Next(255); g2 = (Byte)r.Next(255); b2 = (Byte)r.Next(255);
      r3 = (Byte)r.Next(255); g3 = (Byte)r.Next(255); b3 = (Byte)r.Next(255);
```

Insert 10 new lines into the `private void Window_SizeChanged( ... )`-function below the line: `myPanel.Height = myCanvas.ActualHeight;`:

```
//Increment colors of RadialGradientBrush "elliBrush".
//Whenever a color byte exceeds 255, it automatically resets to 0.
stop1.Color = Color.FromRgb( r1++, g1++, b1++ );
stop2.Color = Color.FromRgb( r2++, g2++, b2++ );
stop3.Color = Color.FromRgb( r3++, g3++, b3++ );
//Move the center of RadialGradientBrush "elliBrush" slightly around point (0.5, 0.5).
radialGradientBrushOrigin.X = 0.5 + 0.05 * Math.Cos(angle);
radialGradientBrushOrigin.Y = 0.5 + 0.05 * Math.Sin(angle);
elliBrush.GradientOrigin = radialGradientBrushOrigin;
angle += Math.PI / 32;
```

Click `Debug` and `Start Debugging F5`.
Finish and remove the comment slashes `//` in front of
`elliBrush.GradientOrigin = new Point( 0.5+0.05*Math.Cos(angle), 0.5+0.05*Math.Sin(angle) );`
and start again.

Complete Code

The complete code of `intro1_XAML_CS` can be found here: **C2D_WPF_Intro_XAML_CS_Code.htm**.

Exercises

Change the initial window size from
`Width="300" Height="300"` to
`Width="600" Height="600"`

2) Change the stroke thickness from
`StrokeThickness="3"` to
`StrokeThickness="10"`.

3) Change the timer interval from
`myTimer.Interval = TimeSpan.FromMilliseconds( 1 );` to
`myTimer.Interval = TimeSpan.FromMilliseconds( 100 );`

4) Lower the minimal window size and the zoom speed from
`if ( myCanvas.ActualWidth < 200 ) zoom = 1.1;` to
`if ( myCanvas.ActualWidth < 100 ) zoom = 1.01;`

5) Increase the maximal window size and the down zoom speed from
`if ( myCanvas.ActualWidth > 800 ) zoom = 0.99;` to
`if ( myCanvas.ActualWidth > 1024 ) zoom = 0.98;`

6) Reduce the number of rings of `elliBrush` from
`RadiusX="0.1" RadiusY="0.1"` to
`RadiusX="0.15" RadiusY="0.15"`

7) Speed up the angular velocitiy of the radial center of `elliBrush` from
`angle += Math.PI / 32;` to
`angle += Math.PI / 8;`