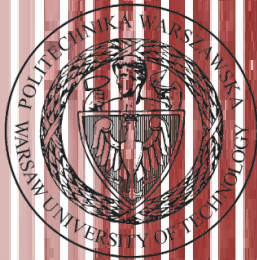


Advanced Programming C#

Lecture 3

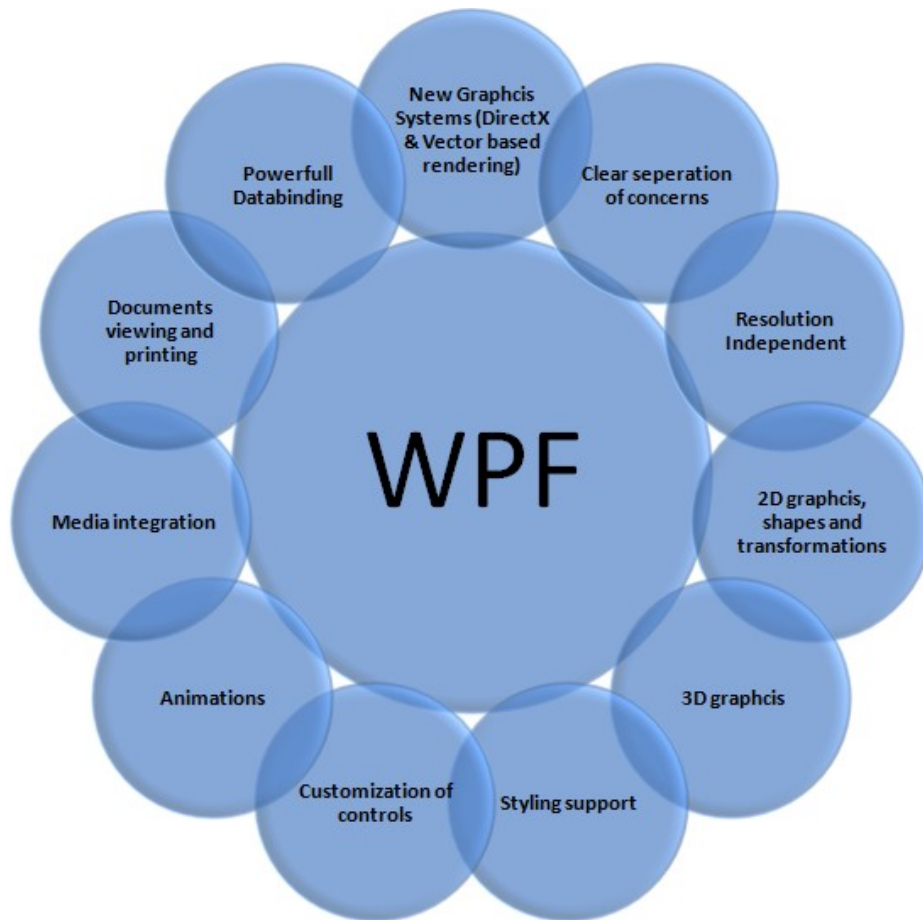
dr inż. Małgorzata Janik
malgorzata.janik@pw.edu.pl

Winter Semester 2018/2019

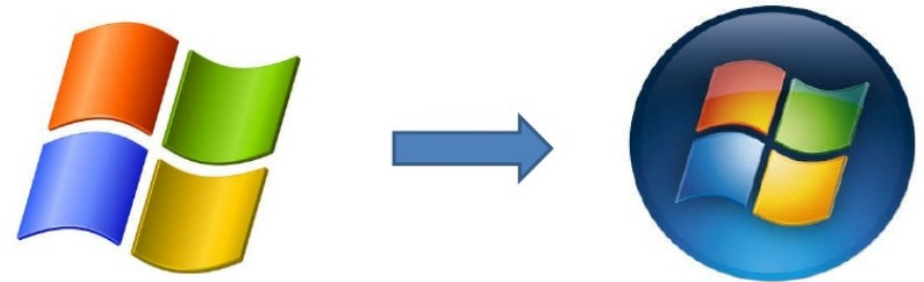


Windows Presentation Foundation

Windows Presentation Foundation



- Allows for clear separation between Design and Functionality
- Better GUI



WPF vs Windows Forms

Window Form



Windows Presentation Foundation



http://www.slideshare.net/nagaharish_movva/windows-presentation-foundation-4377923

Unified presentation

	GDI/Windows Forms	Flash	PDF	COM Interop	Directx	WPF
3D					✓	✓
Documents			✓			✓
Animation		✓			✓	✓
Video		✓		✓	✓	✓
Interactive UI controls	✓	✓		✓		✓

http://www.slideshare.net/nagaharish_movva/windows-presentation-foundation-4377923

Working with WPF

- User interface is designed in XAML
- Underlying functionality is build using C#

XAML

```
<Button Width="60">  
    OK  
<Button.Background>  
    Blue  
</Button.Background>  
</Button>
```

Designer

C#

```
Button btn=new  
    Button();  
btn.Content="OK";  
btn.Width=60;  
btn.Background=new  
SolidColorBrush(Colors.  
    Blue);
```

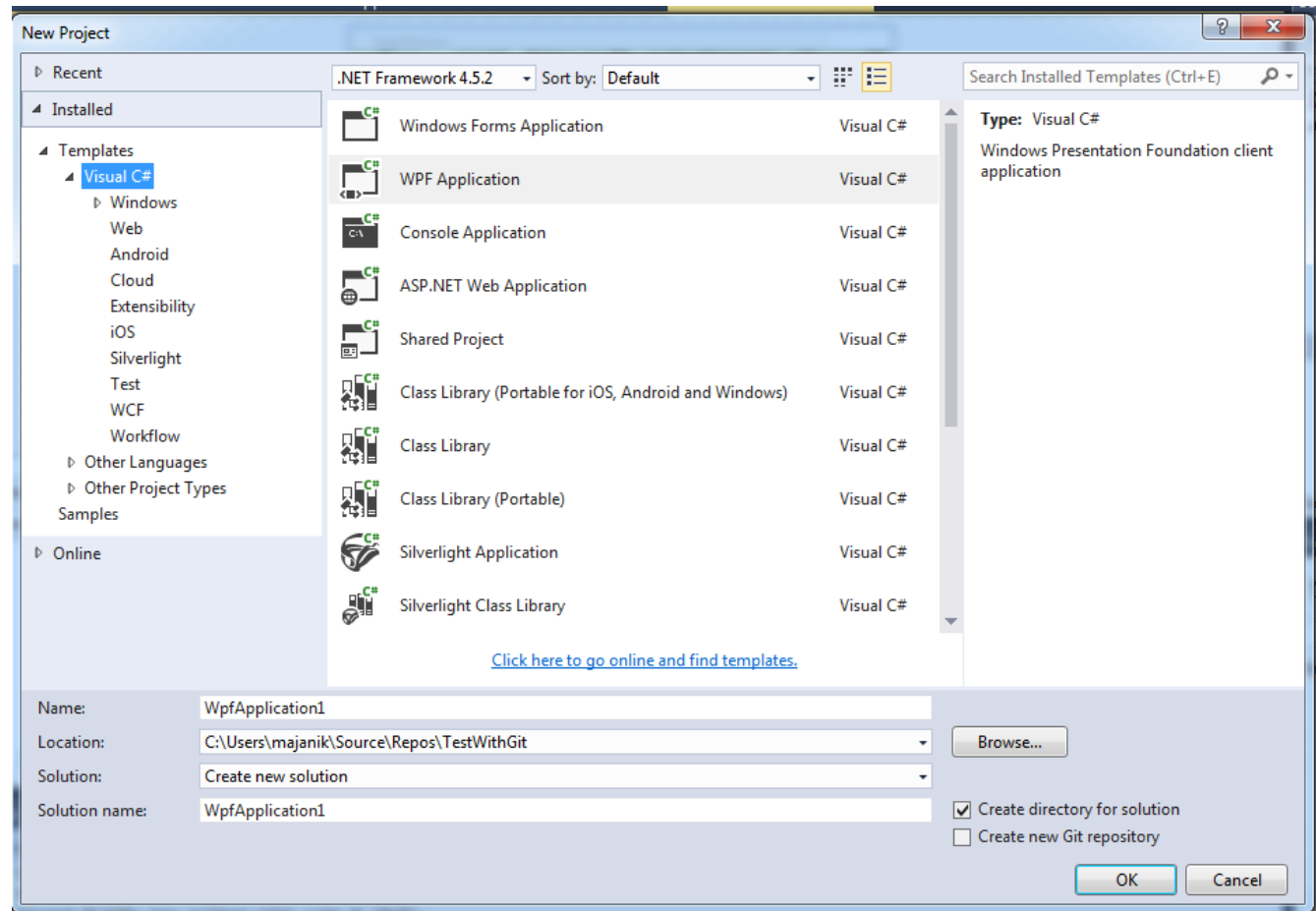
VB.NET

```
Dim btn As New  
    Button  
btn.Content="OK"  
btn.Width=60  
btn.Background=new  
_SolidColorBrush  
_ (Colors.Blue)
```

Developer

New Project: WPF Application

- File → New → Project... → WPF Application

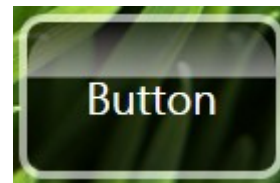


Today

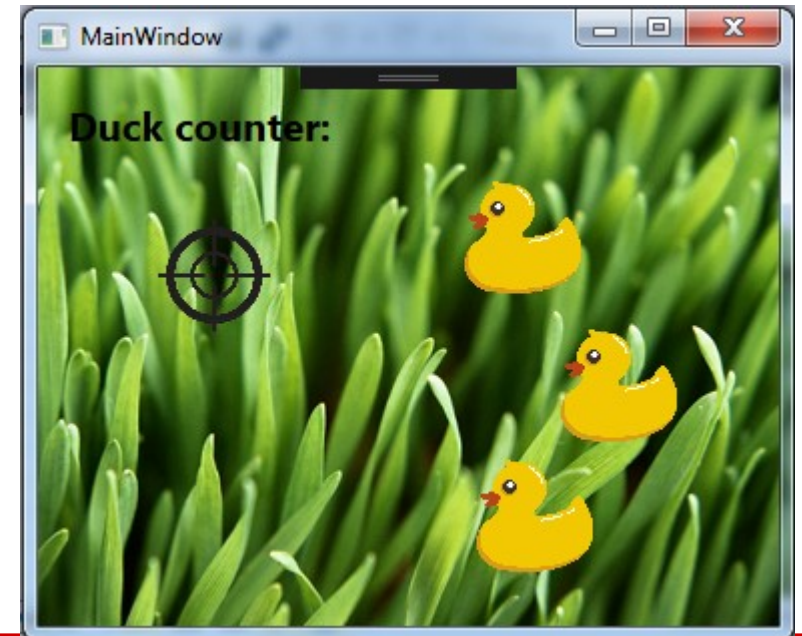
(1) WPF tutorial



(2) Advanced controls



(3) Ducks-shooting game



WPF

The screenshot displays the Microsoft Visual Studio IDE with the 'WpfApplicationLab3' project open. The interface is divided into several panes:

- Toolbox:** Located on the left, it contains 'Common WPF Controls' (Pointer, Border, Button, CheckBox, ComboBox, DataGrid, Grid, Image, Label, ListBox, RadioButton, Rectangle, StackPanel, TabControl, TextBox) and 'All WPF Controls' (Pointer, Border, Button, Calendar, Canvas, CheckBox, ComboBox, ContentControl, DataGrid, DatePicker, DockPanel).
- Designer:** The central area, outlined in red, shows a visual representation of the application window. It features a grid layout with a 'Show text' button, a 'Show' button with an eye icon, a text box containing 'Tu cos napisz', and a 'Label'. A red circle highlights the 'Show' button, and a blue circle highlights the text box. The word 'Designer' is written in red on the right side of this pane.
- XAML Editor:** The bottom pane, outlined in purple, shows the XAML code for the 'MainWindow'. The code is as follows:

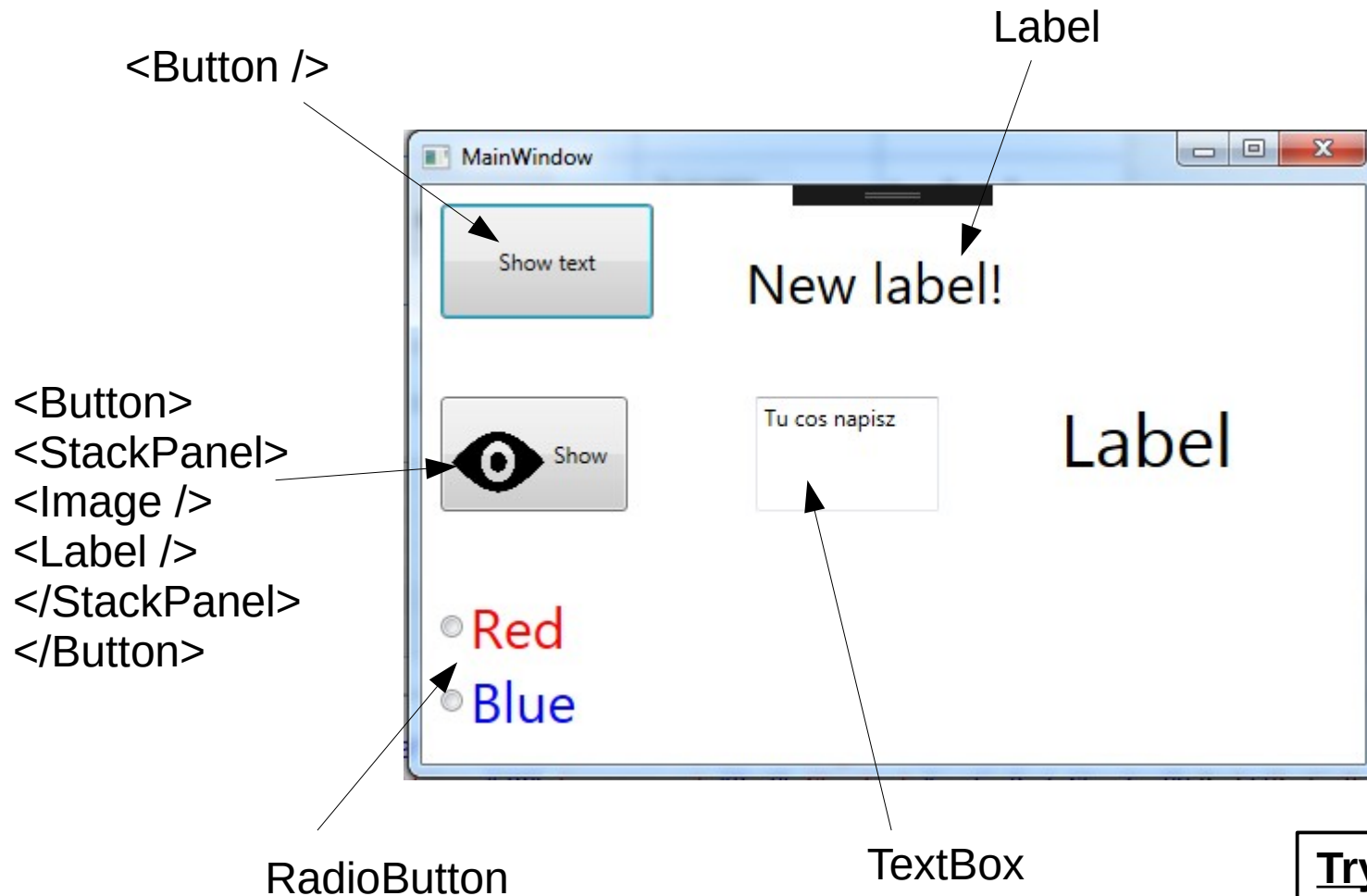
```
1 <Window x:Class="WpfApplicationLab3.MainWindow"
2     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
5     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
6     xmlns:local="clr-namespace:WpfApplicationLab3"
7     mc:Ignorable="d"
8     Title="MainWindow" Height="350" Width="525">
9
```

The word 'XAML' is written in purple on the right side of this pane.
- Solution Explorer:** On the right, it shows the project structure, including 'Properties', 'Resources.resx', 'Settings.settings', 'References', and 'Resources' (dollar.jpg, eye.png).
- Properties Window:** At the bottom right, it shows the properties for the selected 'textBox' control, including 'Name', 'Type', 'Appearance', 'Opacity', 'Visibility', and 'BorderThickness'.

MainWindow.xaml → Design (XAML)

MainWindow.xaml.cs → Functionality (C#)

WPF Tutorial - Controls



Try both:

- Using Toolbox
- Creating Control by hand
in XAML

WPF Tutorial - Properties

<Button Content = "Show text" />

[Label] Content = "New label!"
FontSize="30"

[Image]
Source="..."



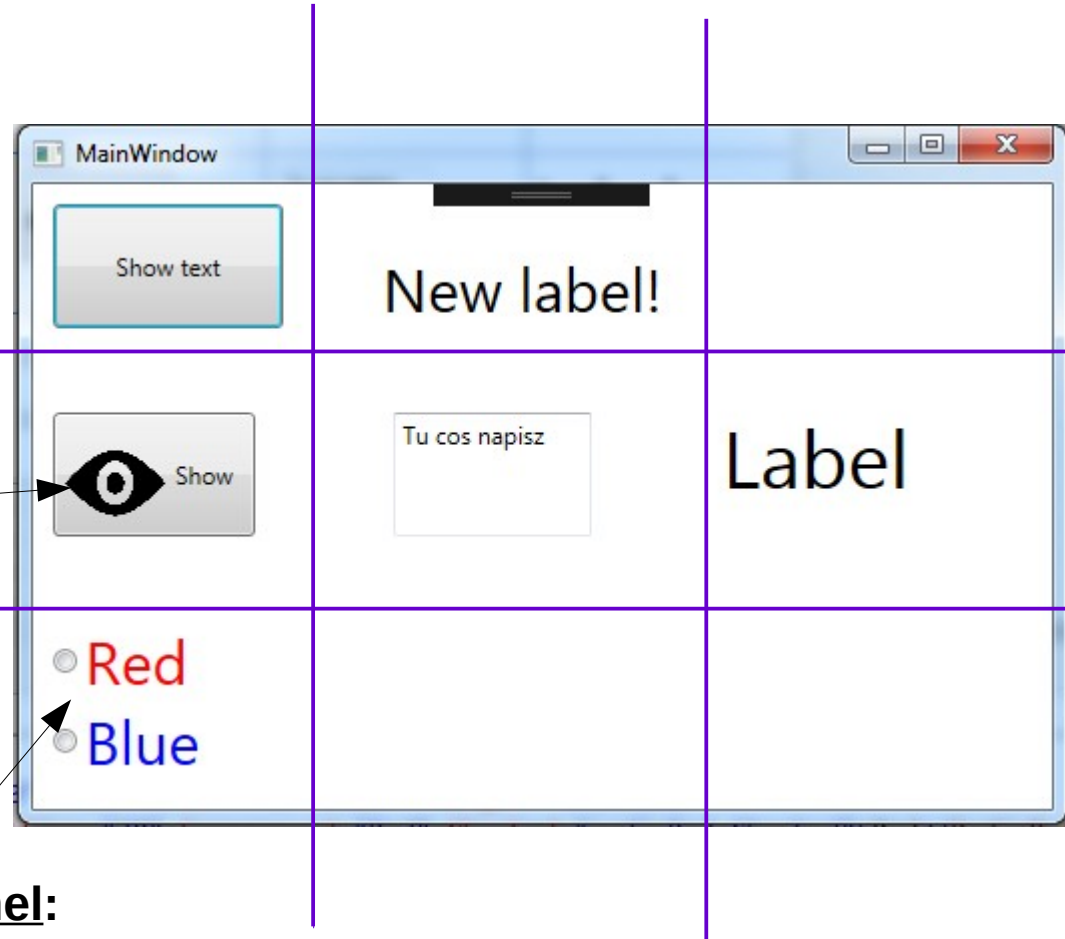
[RadioButton]
Foreground="Blue"

Try both:

- Using Properties window
- Adding Property by hand
in XAML

WPF Tutorial - Layouts

GridLayout: 3 rows, 3 columns



StackPanel:
Horizontal

StackPanel:
Vertical

WPF: Grid Layout

Everything between:

```
<grid>
```

```
</grid>
```

is in grid layout. By default there is 1 row and 1 column.

To create 2 rows and 3 columns we would need to add in the beginning following code:

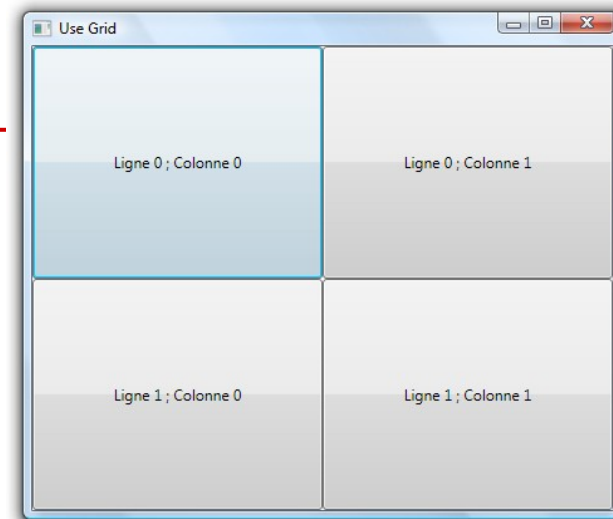
```
<Grid.ColumnDefinitions>  
  <ColumnDefinition></ColumnDefinition>  
  <ColumnDefinition></ColumnDefinition>  
  <ColumnDefinition></ColumnDefinition>  
</Grid.ColumnDefinitions>  
<Grid.RowDefinitions>  
  <RowDefinition></RowDefinition>  
  <RowDefinition></RowDefinition>  
</Grid.RowDefinitions>
```

3 columns

2 rows

To put a control into certain column & row do:

```
<Button x:Name="button" Grid.Row="0" Grid.Column="0"  
  VerticalAlignment="Center" Content="Show text" />
```



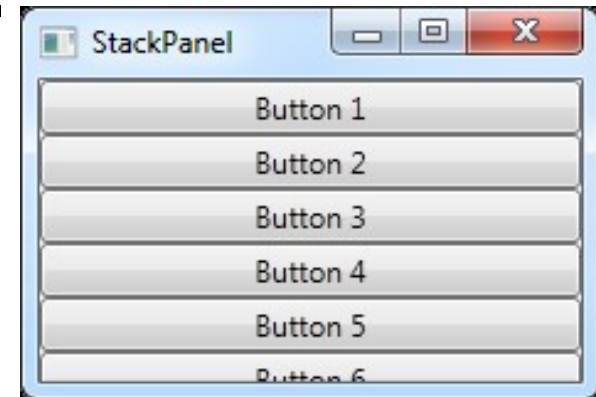
WPF: Stack Layout

Everything between:

```
<StackPanel>
```

```
</StackPanel>
```

is in stack layout. Basic property of the StackPanel is Orientation, which can be **Horizontal** or **Vertical**.



```
<StackPanel Orientation="Horizontal" Height="52">  
    <Image x:Name="image1" HorizontalAlignment="Left" Height="60"  
        VerticalAlignment="Top" Width="50" Source="Resources/eye.png" Stretch="Fill"/>  
    <Label VerticalAlignment="Center">Show</Label>  
</StackPanel>
```



Adding image

- Similarly as before, add to the project using **Resources.resx**
- After adding an image, do:
 - Right click the image file in the Solution Explorer
 - Click Properties
 - Select Build Action to Resource
 - Clean and Build (from menu)
 - (→ Afterwards add an image / reload the image in the Properties window again → Source property → click arrow → choose again from the list)

Events

- When event is added, we move to the c# in the MainWindow.xaml.cs file

Try both:

- Using Properties window
- Creating Event by hand
in XAML

- Commands that may be useful:

```
labelShowText.Visibility = Visibility.Visible;  
labelCopyText.Foreground = Brushes.Red;  
labelCopyText.Content = textBox.Text;
```


Designing New Control

- Glass Button

More info:

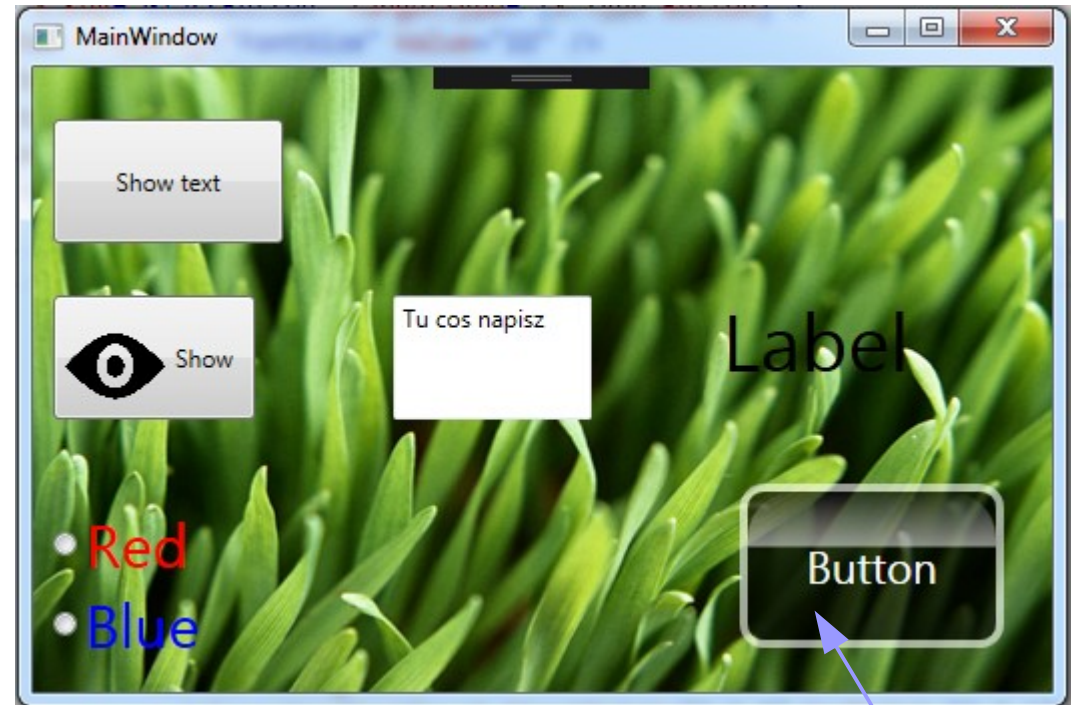
<http://www.jadeskaggs.com/2009/05/04/creating-custom-glass-buttons-with-xaml-in-wpf/>

- Project → Add
→ Resource
Dictionary...

- Resource Dictionary

Copy file:

<http://www.if.pw.edu.pl/~majanik/data/Csharp/Files/GlassButton.xaml>



Glass effect of transparency

To use our GlassButton style, we must now add a resource reference into App.xaml file:

```
<ResourceDictionary Source="GlassButton.xaml"/>
```

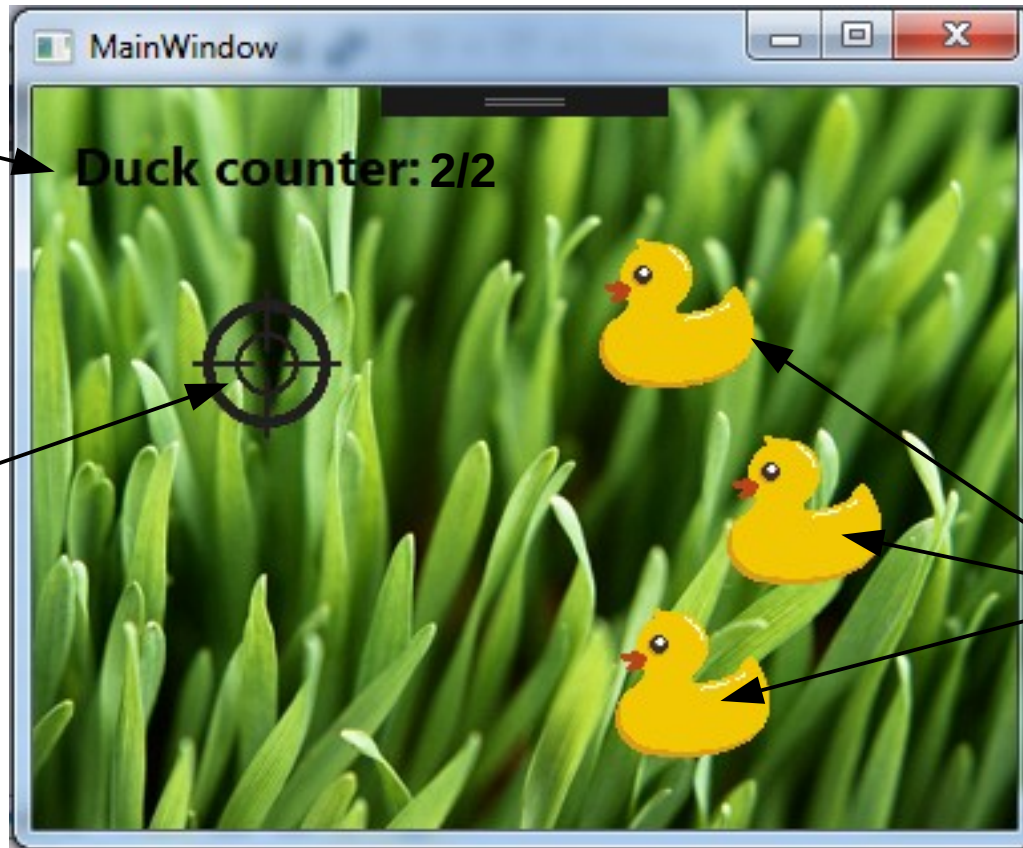
Add control:

```
<Button Style="{DynamicResource GlassButton}"  
        Margin="0,0,24,22" x:Name="buttonGlass1"  
        Height="82" VerticalAlignment="Bottom"  
        HorizontalAlignment="Right" Grid.Row="2" Grid.Column="2"  
        Width="132" Click="buttonGlass1_Click">Button</Button>
```

Shooting ducks

COUNTER

Label control
(changes when duck is shot or dissaper)



TARGET

Image control
(moved with mouse)

DUCKS

Image control
(moved by the Timer)

Useful tools

- Moving objects:

```
duck.Margin = new Thickness(duck1.Margin.Left + 2, duck1.Margin.Top, 0, 0);
```

- Get Mouse Position:

```
var point = e.GetPosition(imageBackground);
```

- Timer

```
System.Windows.Threading.DispatcherTimer dispatcherTimer = new
```

```
System.Windows.Threading.DispatcherTimer();
```

```
    dispatcherTimer.Tick += dispatcherTimer_Tick; //event!!
```

```
    dispatcherTimer.Interval = new TimeSpan(20, 0, 0);
```

```
    dispatcherTimer.Interval = TimeSpan.FromMilliseconds(1);
```

```
    dispatcherTimer.Start();
```

Useful tools

- Flip image:

```
duck.RenderTransformOrigin = new Point(0.5, 0.5);  
  
    ScaleTransform flipTrans = new ScaleTransform();  
    flipTrans.ScaleX = -1; // or 1  
    //flipTrns.ScaleY = -1;  
    duck.RenderTransform = flipTrans;
```


Resources that can be used

- Eye:

<http://www.if.pw.edu.pl/~majanik/data/Csharp/Files/eye.png>

- Glass button:

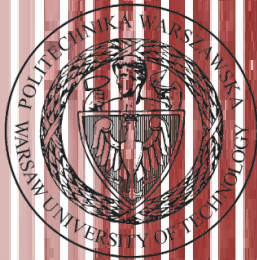
<http://www.if.pw.edu.pl/~majanik/data/Csharp/Files/GlassButton.xaml>

- Ducks:

<http://www.if.pw.edu.pl/~majanik/data/Csharp/Files/duck.png>

http://www.if.pw.edu.pl/~majanik/data/Csharp/Files/Aero_Grass.jpg

<http://www.if.pw.edu.pl/~majanik/data/Csharp/Files/target.png>



THE END

dr inż. Małgorzata Janik
malgorzata.janik@pw.edu.pl