



## WHAT IS WINDOWS PRESENTATION FOUNDATION?

- Presentation system for building Windows-based Applications with a strong focus on UI:
  - Windows Desktop
  - Windows Store
  - Xbox One (UWP – Universal Windows Platform)
  - ...
- Free and Open Source, Developed by Microsoft
  - Included in the .NET Framework
- Main Features:
  - Designer friendly (XAML)
  - Layout
  - 2D/3D Graphics, Media, Animations
  - Styles and Templates
  - Data Binding
  - ....
- Widely used for enterprise applications

## WHAT IS WINDOWS PRESENTATION FOUNDATION?

- Markup + Code-behind

- Markup: **XAML**

  - Define the appearance (layout, colors, visual components, ... )

- Code-behind: **C#**

  - Create functionality that responds to user interaction
  - Link visuals to the data
  - Business logic

# Front-end (WPF & XAML)

## WHAT IS XAML?

## EXTENSIBLE APPLICATION MARKUP LANGUAGE

- Separates UI from Behavior
- Designer Friendly
- Based on XML

```
<Grid>
  <Grid.ColumnDefinitions>
    <ColumnDefinition />
    <ColumnDefinition Width="Auto"/>
  </Grid.ColumnDefinitions>

  <!-- Banner Image -->
  <Image VerticalAlignment="Bottom" Grid.Column="2" />

  <!--Add User Button-->
  <Button Grid.Column="0" Grid.Row="0" HorizontalAlign="Center"
    <StackPanel Orientation="Horizontal" Margin="10"
      <Image Width="20" Height="20" Source="/Resources/Avatar.png" />
      <Label Foreground="White" FontSize="20">Add User</Label>
    </StackPanel>
  </Button>
</Grid>
```

# XML

## SHORT INTRODUCTION

## WHAT IS XML?

- initial goal: data transport
- does not execute anything, just carries data
- needs some software to process



```
<game>  
    <name>Ring fit adventure</name>  
    <publisher>Nintendo</publisher>  
    <release>2019</release>  
</game>
```

# XML basics

## XML TAGS <element/>

- can hold:
  - other elements (nested)  
eg.: game
  - a piece of data  
eg.: name, release
- each tag must be closed
  - closing tag: <game>..</game>
  - close right away: <release />
  - case sensitive!
- only one **root element**
  - highest level
  - Eg.: games

```
<games>
  <game>
    <name>Wii Sports</name>
    <publisher>Nintendo</publisher>
    <release>2006</release>
  </game>
  <game>
    <name>Just Dance</name>
    <publisher>Ubisoft</publisher>
    <release />
  </game>
  <game>
    <name>Ring fit adventure</name>
    <publisher>Nintendo</publisher>
    <release>2019</release>
  </game>
</games>
```

## XML ATTRIBUTES attribute = "some value"

- can hold:
  - a piece of data  
eg.: id, release
- can have multiple per tag
- give more info on the tag
- easier to search / filter,...

```
<games>  
  <game id="f239af3d" release="2006">  
    <name>Wii Sports</name>  
    <publisher>Nintendo</publisher>  
    <release>2006</release>  
  </game>  
  <game id="4ebd0208">  
    <name>Just Dance</name>  
    <publisher>Ubisoft</publisher>  
  </game>  
  <game id="6a23dabe" release="2019">  
    <name>Ring fit adventure</name>  
    <publisher>Nintendo</publisher>  
  </game>  
</games>
```



## XML STRUCTURE

- Comment tag
  - `<!-- your comment -->`
- Ways to define a fixed structure using an XML schema
  - predefined tags / attributes
- Examples:
  - HTML
  - XAML

```
<games>
  <game id="f239af3d" release="2006">
    <name>Wii Sports</name>
    <publisher>Nintendo</publisher>
    <release>2006</release>
  </game>
  <workoutGame code="4ebd0208">
    <name>Just Dance</name>
    <publisher>Ubisoft</publisher>
  </workoutGame>
  <!-- my favorite -->
  <Game id="6a23dabe" year="2019">
    <name>Ring fit adventure</name>
    <publisher>Nintendo</publisher>
  </Game>
</games>
```







# XAML user interface

## XAML IN A WPF PROJECT

### Create a new project

#### Recent project templates

-  Console App (.NET Framework) C#
-  Windows Desktop Wizard C++
-  Windows Desktop Application C++
-  Empty Project C++

Search for templates (Alt+S)



Clear all

C#

All platforms

Desktop



#### NUnit Test Project (.NET Core)

A project that contains NUnit tests that can run on .NET Core on Windows, Linux and MacOS.

C#

Linux

macOS

Windows

Desktop

Test

Web



#### Windows Forms App (.NET Framework)

A project for creating an application with a Windows Forms (WinForms) user interface

C#

Windows

Desktop



#### WPF App (.NET Framework)

Windows Presentation Foundation client application

C#

XAML

Windows

Desktop



#### WPF App (.NET)

Windows Presentation Foundation client application

C#

XAML

Windows

Desktop



WPF Custom Control Library (.NET)

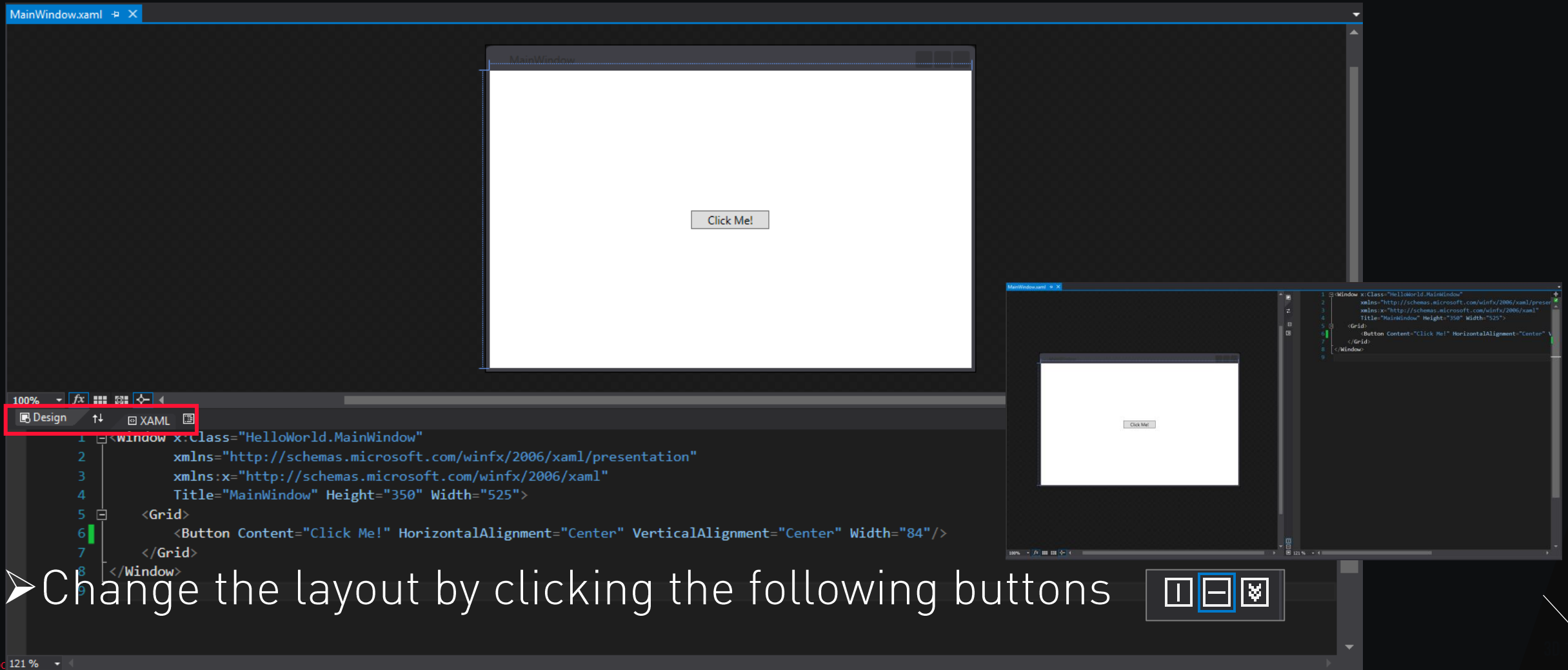
Next

.11

# XAML user interface

## WPF: XAML VIEW vs DESIGN VIEW

- Modifying XAML has a direct impact on the UI (Immediate feedback)
  - UI Designer and XAML editor are linked together



The screenshot shows the Visual Studio IDE with the WPF XAML editor in Design view. The main window displays a 'Click Me!' button. The XAML code is visible in the background, and a smaller inset shows the XAML code in the background.

100% ▾ fxc [Design] [XAML] [Code]

```
1 <Window x:Class="HelloWorld.MainWindow"
2     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4     Title="MainWindow" Height="350" Width="525">
5     <Grid>
6         <Button Content="Click Me!" HorizontalAlignment="Center" VerticalAlignment="Center" Width="84"/>
7     </Grid>
8 </Window>
```

➤ Change the layout by clicking the following buttons

121 %

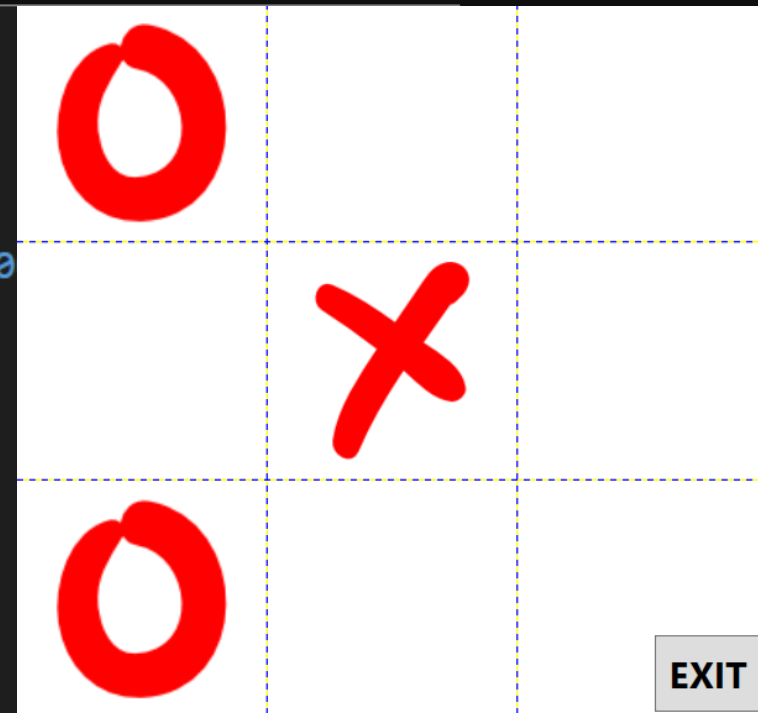
# XAML user interface

## XAML EXAMPLE

```
<Window x:Class="DemoXaml.MainWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:local="clr-namespace:DemoXaml"
  mc:Ignorable="d"
  Title="TicTacToe" Height="500" Width="500">

  <Grid x:Name="grdGame" ShowGridLines="True">
    <Grid.ColumnDefinitions...>
    <Grid.RowDefinitions...>
    <Image Source="Resources/O.png" Margin="13" />
    <Image Grid.Column="1" Grid.Row="1" Source="Resources/X.png" Margin="13" />
    <Image Grid.Column="0" Grid.Row="2" Source="Resources/O.png" Margin="13" />

    <Button Grid.Column="2" Grid.Row="2" Content="EXIT" FontSize="24" FontWeight="Bold"
      Padding="8" Margin="4" HorizontalAlignment="Right" VerticalAlignment="Bottom"
      Height="50" Click="Button_Click" />
  </Grid>
</Window>
```



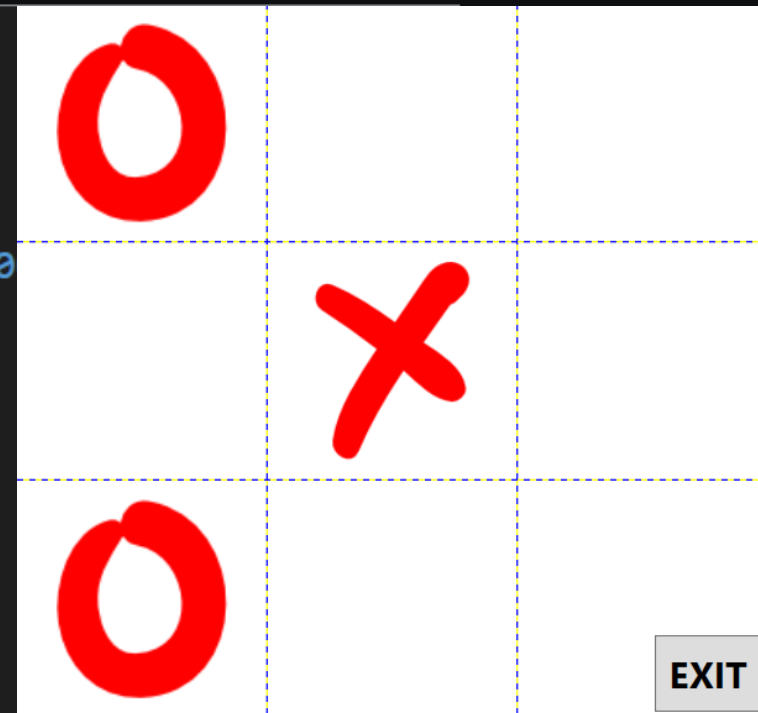
# XAML user interface

## XAML EXAMPLE

```
<Window x:Class="DemoXaml.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:DemoXaml"
    mc:Ignorable="d"
    Title="TicTacToe" Height="500" Width="500">

    <Grid x:Name="grdGame" ShowGridLines="True">
        <Grid.ColumnDefinitions...>
        <Grid.RowDefinitions...>
        <Image Source="Resources/O.png" Margin="13" />
        <Image Grid.Column="1" Grid.Row="1" Source="Resources/X.png" Margin="13" />
        <Image Grid.Column="0" Grid.Row="2" Source="Resources/O.png" Margin="13" />

        <Button Grid.Column="2" Grid.Row="2" Content="EXIT" FontSize="24" FontWeight="Bold"
            Padding="8" Margin="4" HorizontalAlignment="Right" VerticalAlignment="Bottom"
            Height="50" Click="Button_Click" />
    </Grid>
</Window>
```



# XAML user interface

## BUTTON: CONTENT?

- nesting tags in XAML



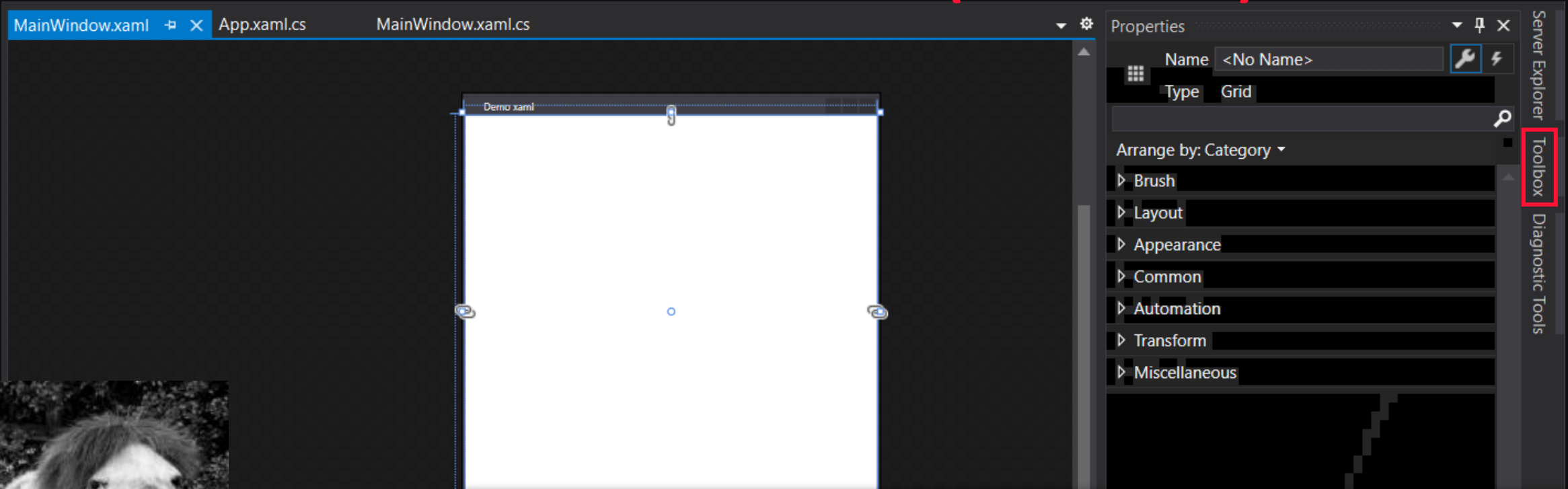
```
<Button Width="200" Height="120" Background="Black" Click="Button_Click">
    <Button.Content>
        <Grid>
            <Image Source="Resources/shawshank-redemption.jpg" Opacity="0.7" Stretch="UniformToFill" />
            <Ellipse Fill="#66FFFFFF" Stroke="Black" StrokeThickness="3" Width="60" Height="60" />
            <TextBlock Text="GO" FontSize="28" FontWeight="Bold"
                Foreground="Black" VerticalAlignment="Center" HorizontalAlignment="Center" />
            <Image Source="Resources/filmstrip.png" Stretch="Fill" />
        </Grid>
    </Button.Content>
</Button>
```

# GENERATE / WRITE XAML CODE

VARIOUS WAYS + WHICH TO CHOOSE



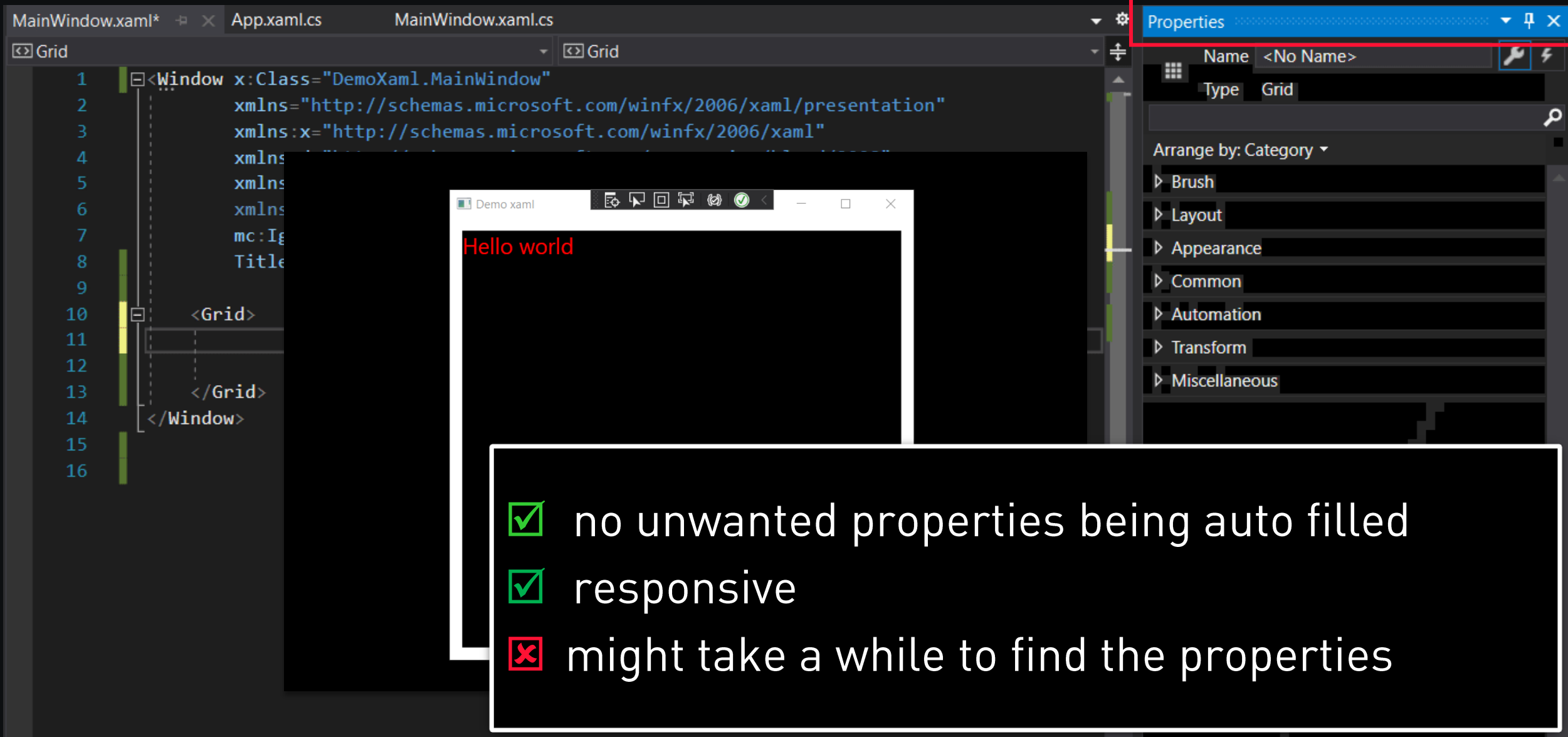
## CREATE XAML CODE: DRAG & DROP (TOOLBOX)



- ✓ easy drag & drop
- ✗ might take a while to find the properties
- ✗ auto fills a whole lot of unwanted properties
- ✗ is not responsive!!



## CREATE XAML CODE: PROPERTIES WINDOW



The screenshot displays the Visual Studio IDE with three main components:

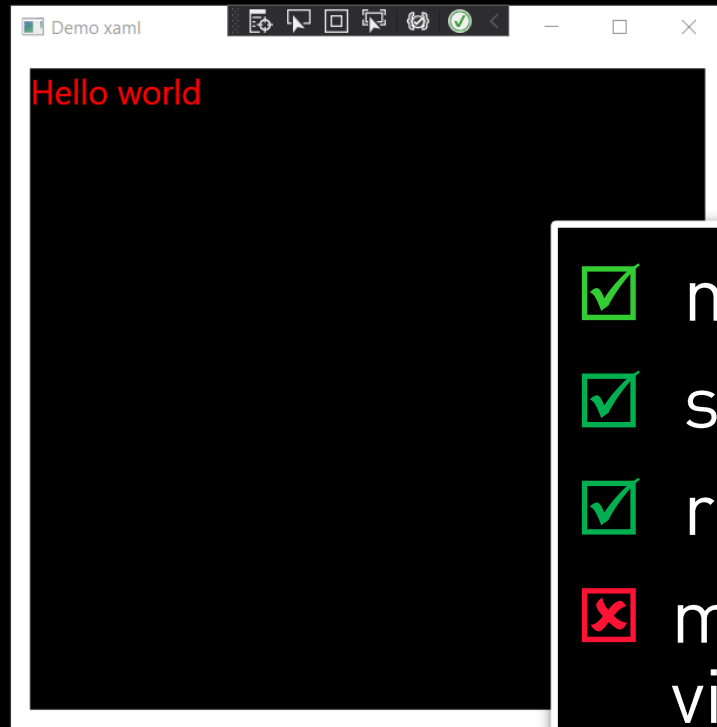
- Code Editor:** Shows the XAML code for a `Window` with a `Grid` child. The code is as follows:

```
1 <Window x:Class="DemoXaml.MainWindow"
2       xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3       xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4       xmlns:mc="http://schemas.microsoft.com/winfx/2006/xaml"
5       xmlns:d="http://schemas.microsoft.com/winfx/2006/xaml"
6       mc:Ignorable="d"
7       Title="Demo xaml"
8       Width="300"
9       Height="200"
10      >
11     <Grid>
12     </Grid>
13 </Window>
```
- Preview Window:** A small window titled "Demo xaml" showing a black rectangle with the text "Hello world" in red.
- Properties Window:** Located on the right, it shows the properties of the selected `Grid` element. The "Name" property is set to "<No Name>" and the "Type" is "Grid". The window is organized into categories: Brush, Layout, Appearance, Common, Automation, Transform, and Miscellaneous.

A red box highlights the "Properties" window title bar. A white box highlights the preview window and the list of categories in the Properties window.

- ✓ no unwanted properties being auto filled
- ✓ responsive
- ✗ might take a while to find the properties

## CREATE XAML CODE: MANUALLY



- ✓ no unwanted properties being added
- ✓ super fast with intellisense (code completion)!
- ✓ responsive
- ✗ more complex properties might be easier with visual properties

## ALTERNATIVE: CODE GENERATED UI

```
TextBlock txt = new TextBlock();  
txt.Text = "Hello world";  
txt.Background = new SolidColorBrush(Colors.Black);  
txt.Foreground = new SolidColorBrush(Colors.Red);  
txt.FontSize = 23;  
txt.Margin = new Thickness(13);  
  
Grid mainGrid = new Grid();  
mainGrid.Children.Add(txt);
```

- ❌ no design viewer
- ❌ takes a long time / more code
- ❌ no more separation of code / UI

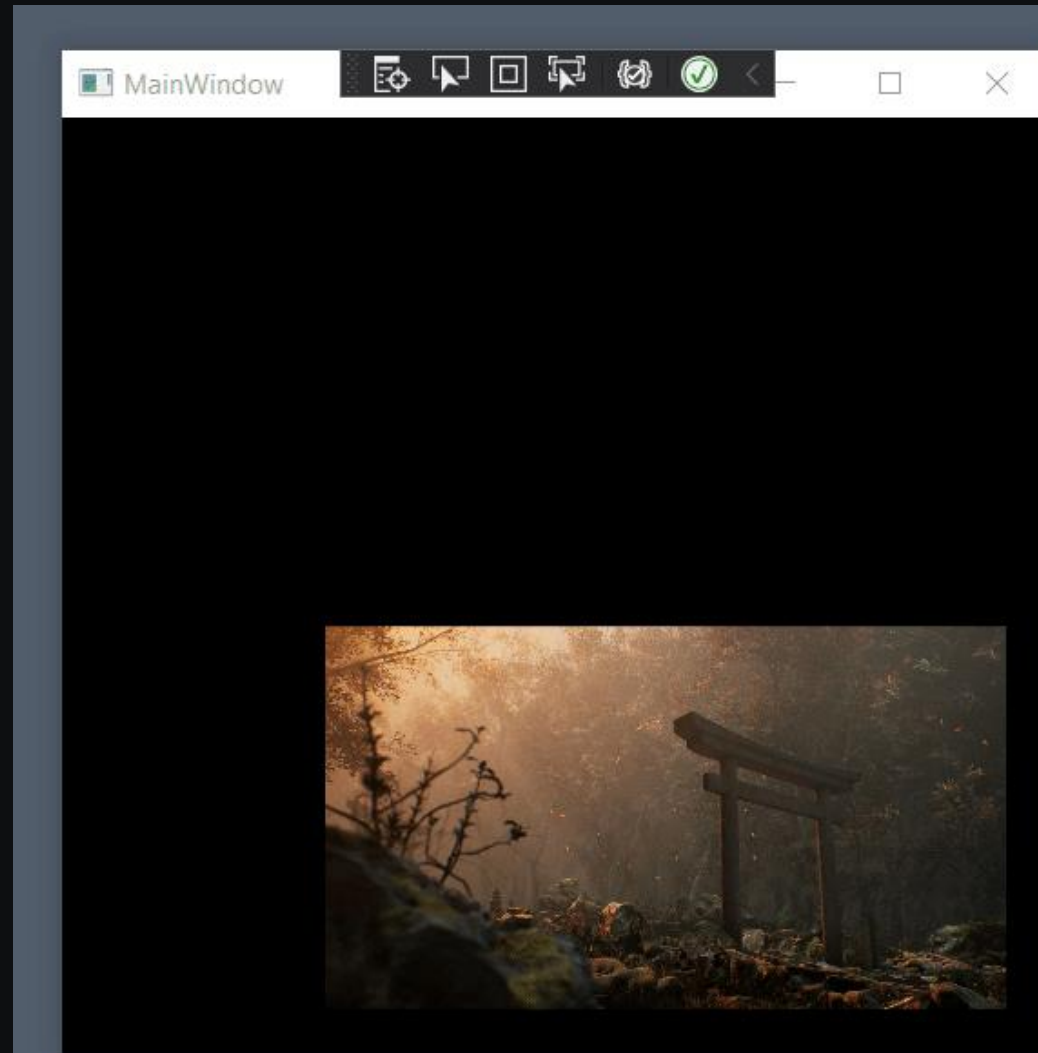
# EXERCISE

WPF / XAML BASICS

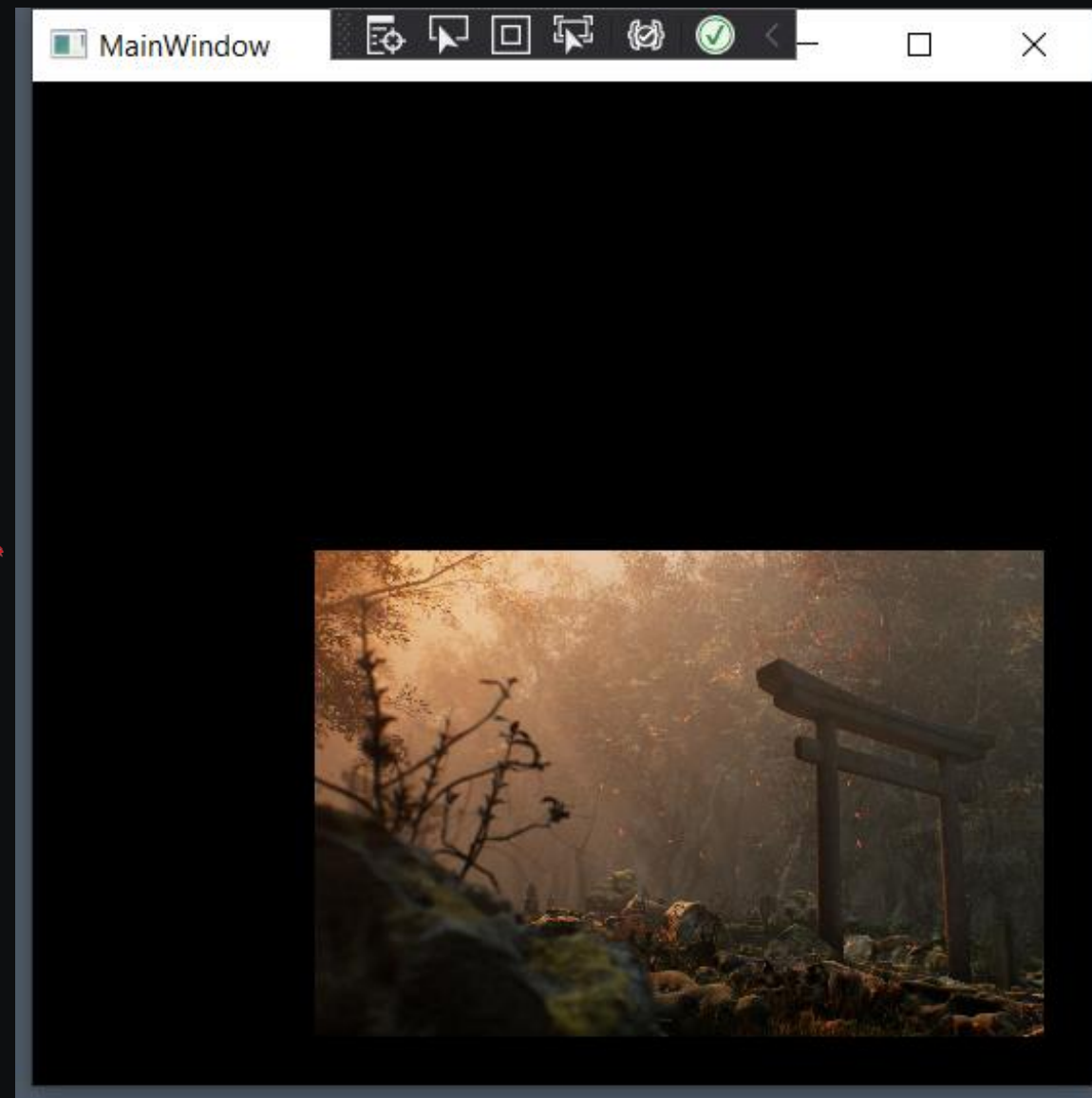
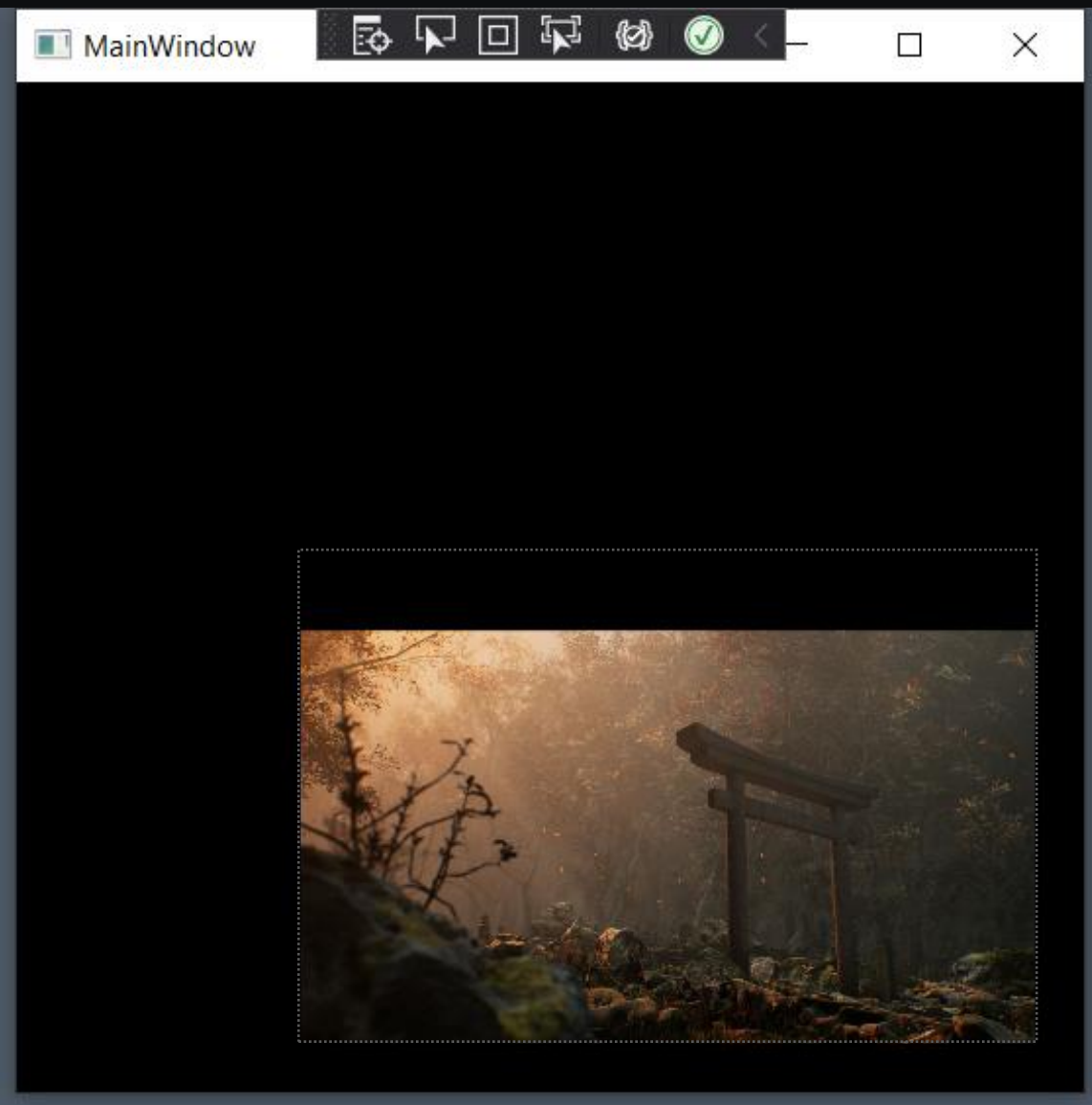
# XAML user interface

## EXERCISE: BASIC XAML

- Start window size:  
450 x 450
- Background: black
- Add the image
  - First, add a folder called 'Resources'
  - Add the given image
- Display the image:
  - ✓ In the bottom right
  - ✓ Exactly 300 x 200
  - ✓ 20px away from sides



## EXERCISE: BASIC XAML – ONE LAST DETAIL



.23

# ADDING BEHAVIOR

CODE BEHIND



# ADDING BEHAVIOR

## XAML vs CODE BEHIND

```
MainWindow.xaml
MainWindow.xaml.cs

<Window x:Class="DemoXaml.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:DemoXaml"
        mc:Ignorable="d"
        Title="Demo xaml" Height="500" Width="500">

    <Grid>
        <Button Content="CLICK ME!" />
    </Grid>
</Window>
```

InitializeComponent() method:

- loads / processes XAML file
- generates object graph to display
- no object graph = no code interaction!

```
MainWindow.xaml
MainWindow.xaml.cs

using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace DemoXaml
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    2 references
    public partial class MainWindow : Window
    {
        0 references
        public MainWindow()
        {
            InitializeComponent();
        }
    }
}
```

# ADDING BEHAVIOR

## NAMING THE ELEMENTS

```
<Grid x:Name="grdMain">
  <Grid.RowDefinitions...>
  <TextBox x:Name="txtName" FontSize="23"/>
  <Button x:Name="btnClick" Content="CLICK ME!" FontSize="35" Grid.Row="1" />
</Grid>
```

```
public partial class MainWindow : Window
{
    0 references
    public MainWindow()
    {
        InitializeComponent();

        txtName.Text = "default";
    }
}
```

➤ naming an element makes it accessible to the code behind

# ADDING BEHAVIOR

## ADD BEHAVIOR USING CODE

```
21 public partial class MainWindow : Window
22 {
23     0 references
24     public MainWindow()
25     {
26         InitializeComponent();
27     }
28 }
29
30
31
32
33
```

- ✓ good overview in code of handled events
- ✗ more depending on specific UI

# ADDING BEHAVIOR

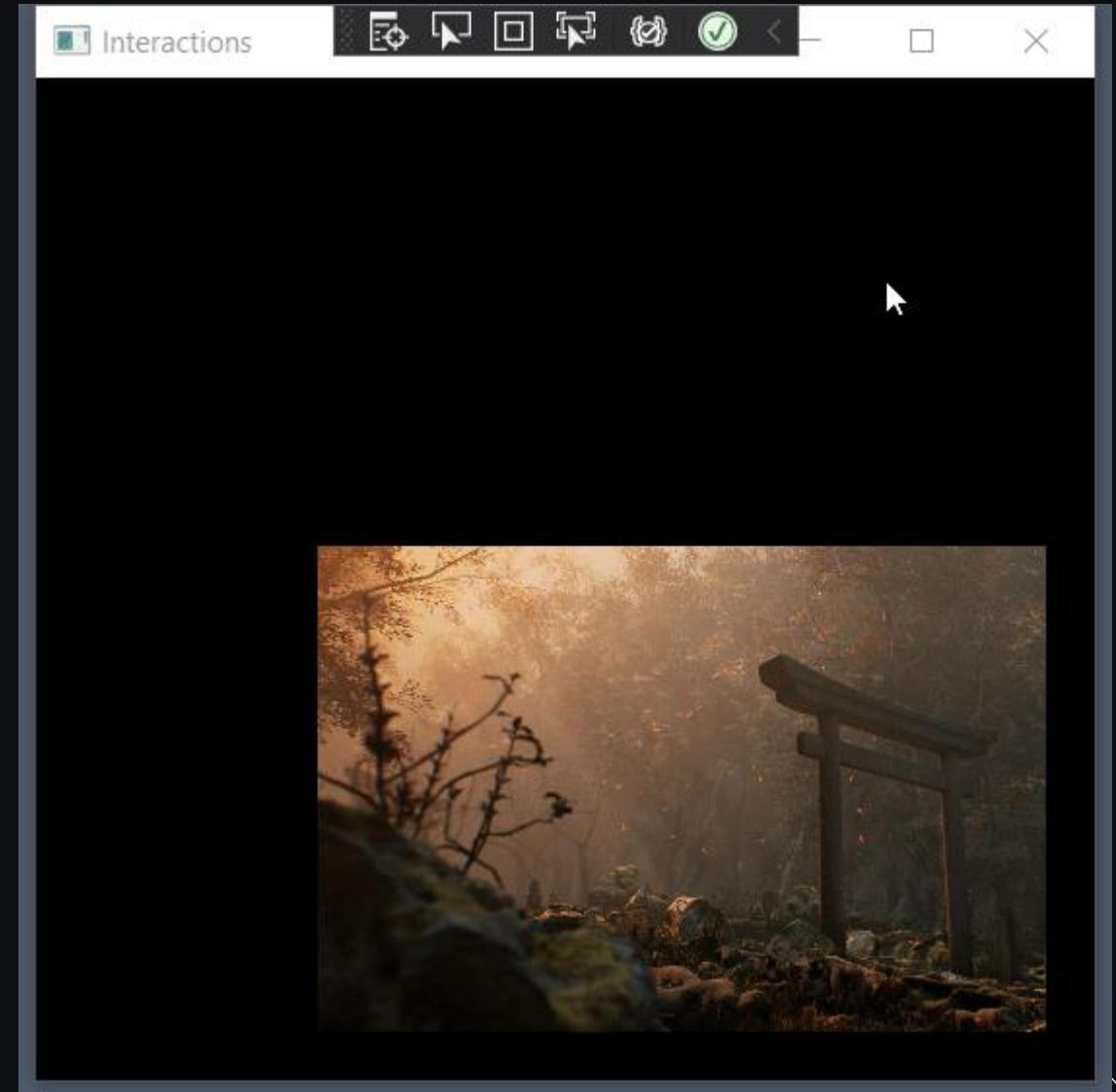
## ADD BEHAVIOR FROM XAML

The screenshot shows the Visual Studio IDE with the following components:

- File Explorer:** App.xaml, MainWindow.xaml, App.xaml.cs, MainWindow.xaml.cs, App.config.
- Code Editor:** Displays the XAML code for MainWindow.xaml. The code defines a Window with a Grid containing a TextBox and a Button. The Button has a Click event handler.
- Properties Window:** Shows the 'Automation' category selected in the 'Arrange by: Category' dropdown.
- Callout Box:** A white box with a black border containing the following text:
  - ✓ (slightly) less depending on specific UI
  - ✗ no quick overview of handled events (in code)

## EXERCISE: ADDING BEHAVIOR

- Create a field `_scale` of type `double`
  - ✓ Initialize to 1.2
- When the mouse hovers over the image, multiply its size with `_scale`
- When the mouse is no longer in the image, divide size by `_scale`
- All done? Try this alternative:
  - When mouse hovers over image, hide it
  - When mouse goes back out, reveal again
  - *Spoiler alert: something seems to go wrong, but why?*



# XAML - THERE IS MORE

WHAT ELSE SHOULD WE KNOW?

what else is there to know?

## XAML: THERE IS MORE....

### ➤ Markup extensions

- Indicated by { }
- `<TextBox text="{Binding FirstName}" foreground="{StaticResource ThemeClr}" />`
- Will be handled later (databelinding)

### ➤ Layout containers

### ➤ Resources & Templates

### ➤ Converters (will be covered in 04-MVVM)

### ➤ Etc., etc., etc., etc.....