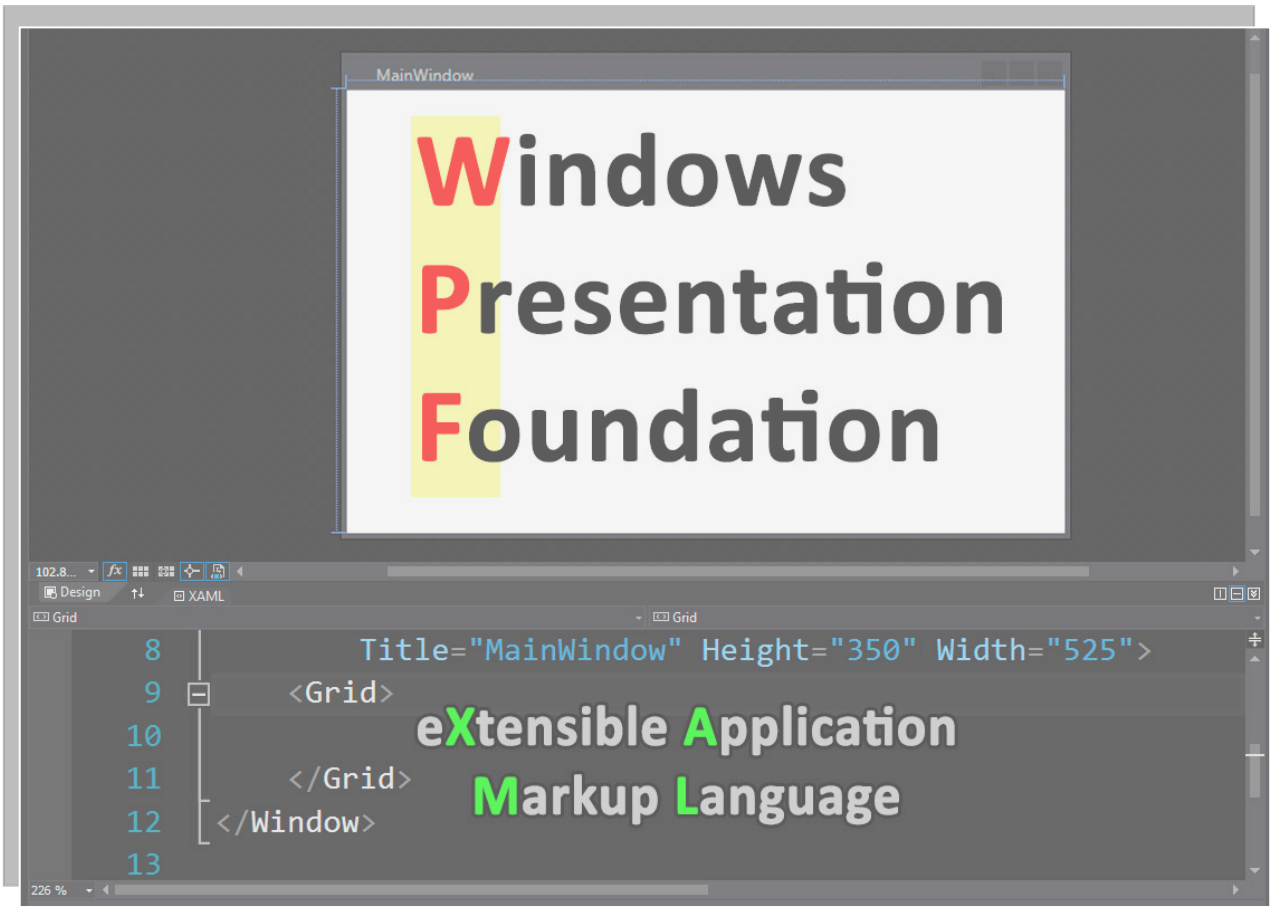
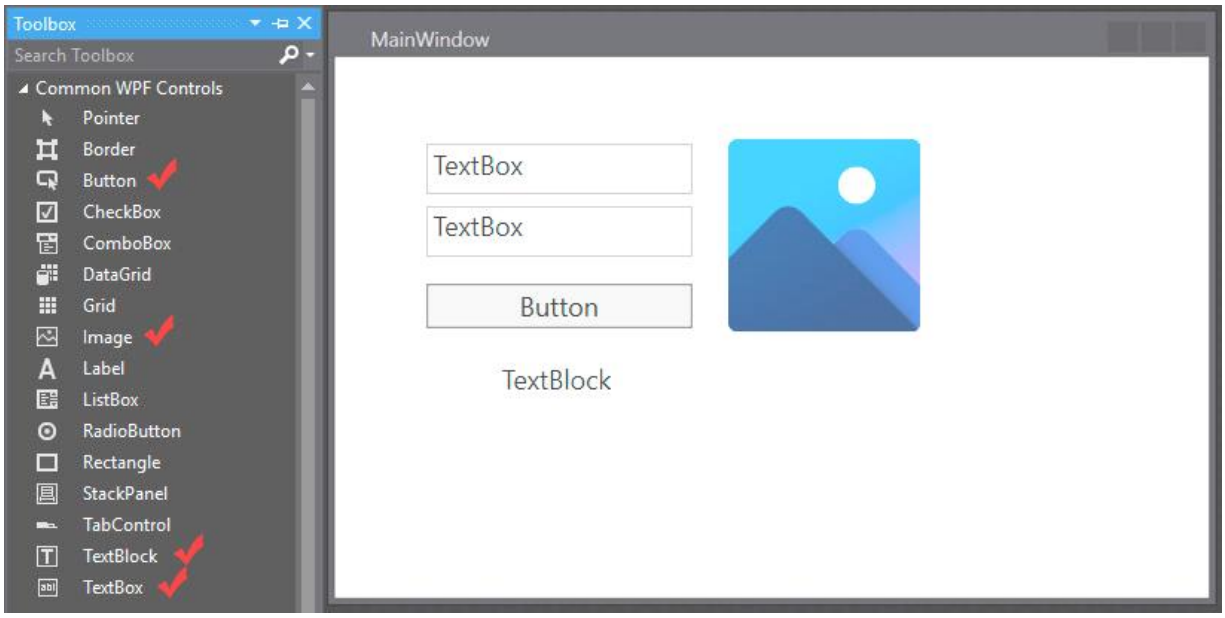



# C# WPF

*Андрій Янковський*

*2020*

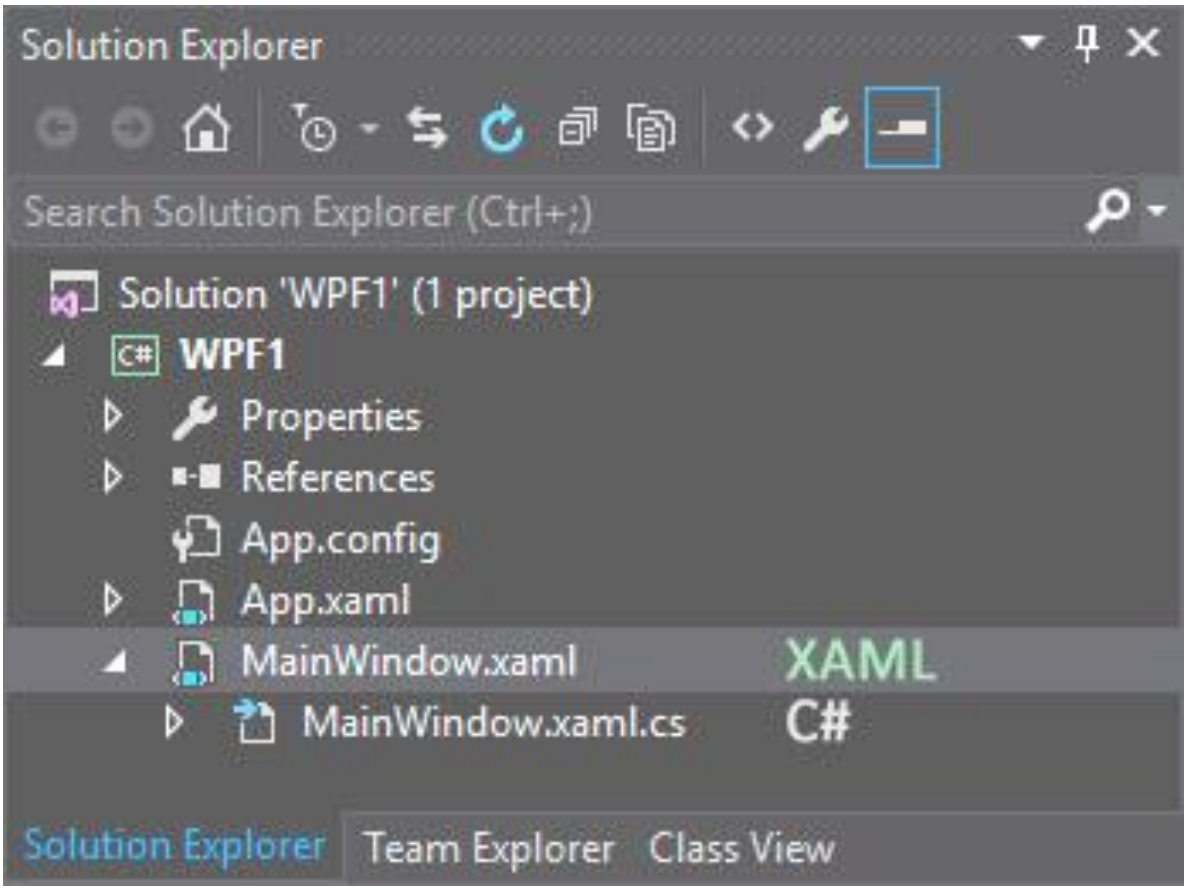
**WPF** – це вже готовий графічний каркас програми, тобто вікно і набір компонентів для нього: текст, кнопка, поле вводу, світлина тощо




Ми можемо просто перетягувати компоненти з панелі елементів у вікно, але для більш точного їх розміщення, один відносно одного, нам потрібні команди розмітки, щось на зразок HTML для сайтів, і Microsoft створив таку розмітку

XAML – розширювана розмітка для програм, яка знаходиться в спеціальному файлі:

MainWindow.xaml

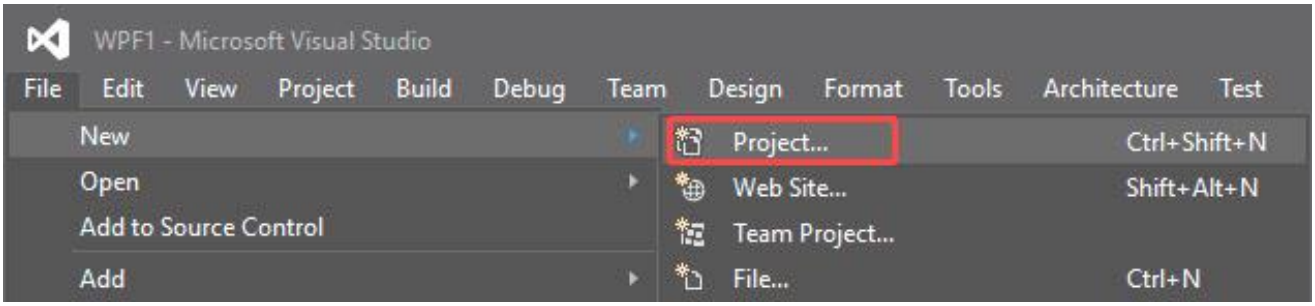



Відповідно, ваш код мовою C# буде знаходитися у файлі:

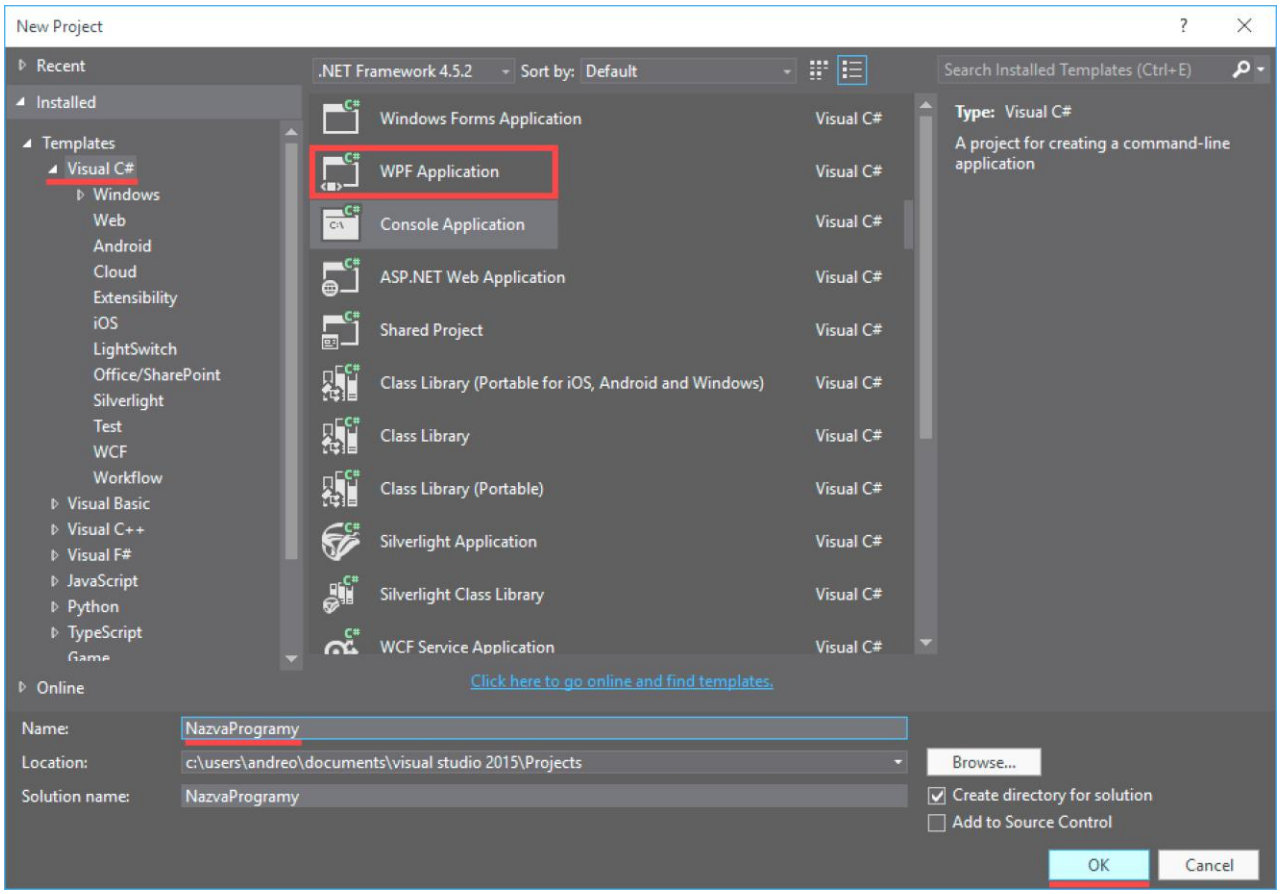
MainWindow.xaml.cs

# Як створити WPF вікно?

## Файл –> Новий –> Проект




У новому вікні ми зліва обираємо мову C#, справа – WPF програма, вказуємо назву програми і натискаємо ОК



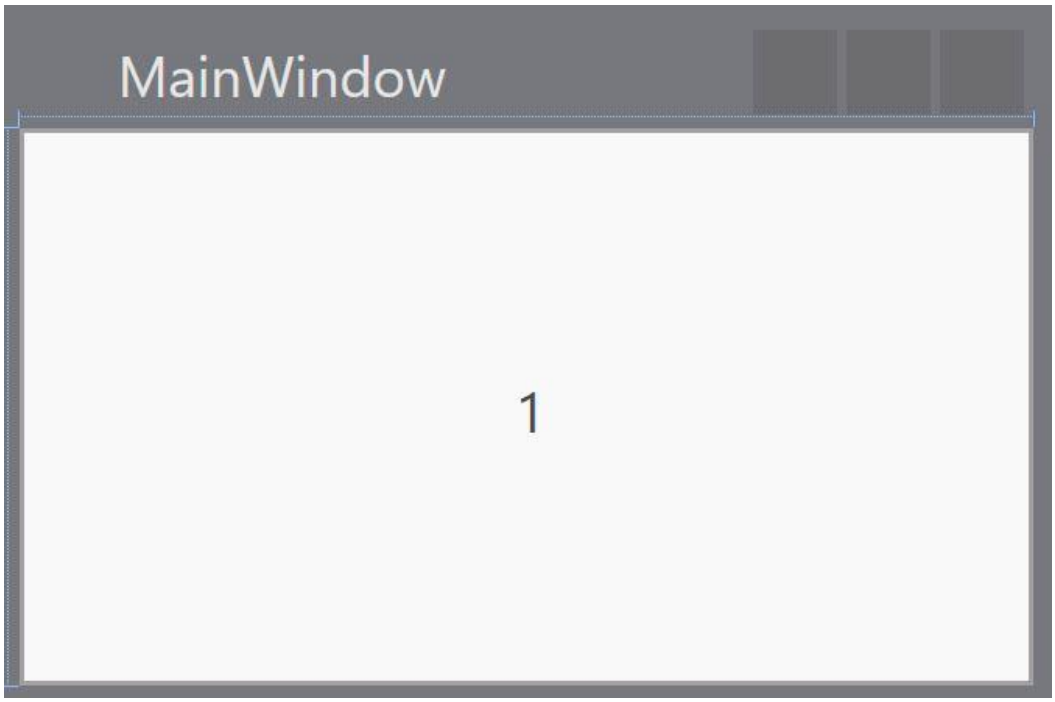
В новому проекті у нас буде пусте вікно і ось така розмітка:

```
<Window x:Class="WPF1.MainWindow"
xmlns="http://schemas.microsoft.com/wi
nfx/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.openxmlformat
s.org/markup-compatibility/2006"
xmlns:local="clr-namespace:WPF1"
mc:Ignorable="d"
Title="MainWindow"
Height="350"
Width="525">
    <Grid>


    </Grid>
</Window>
```

Тут можна вказати назву програми і розмір вікна

А давайте додамо в середину сітки (Grid) кнопку 1:

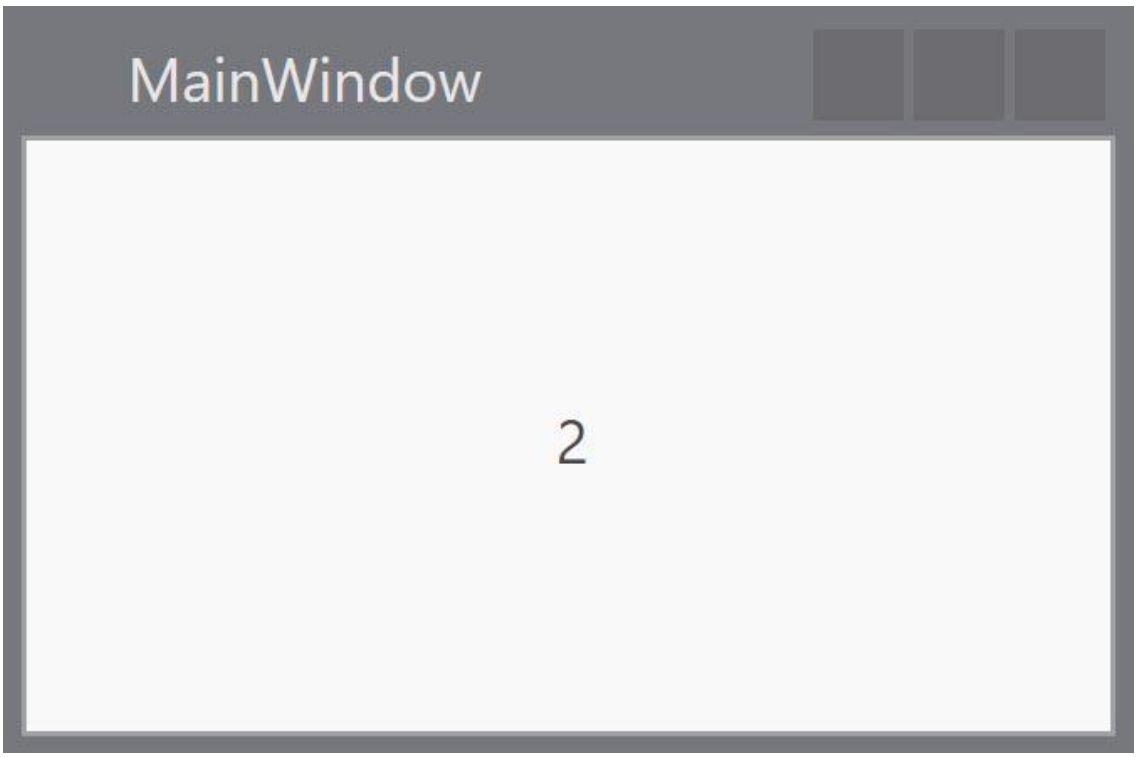
```
<Grid>
      <Button Content="1" />
</Grid>
```




Кнопка автоматично розтягнулася на весь екран, бо все, що знаходиться в сітці – автоматично займає увесь вільний простір.

# А якщо додати ще одну кнопку?

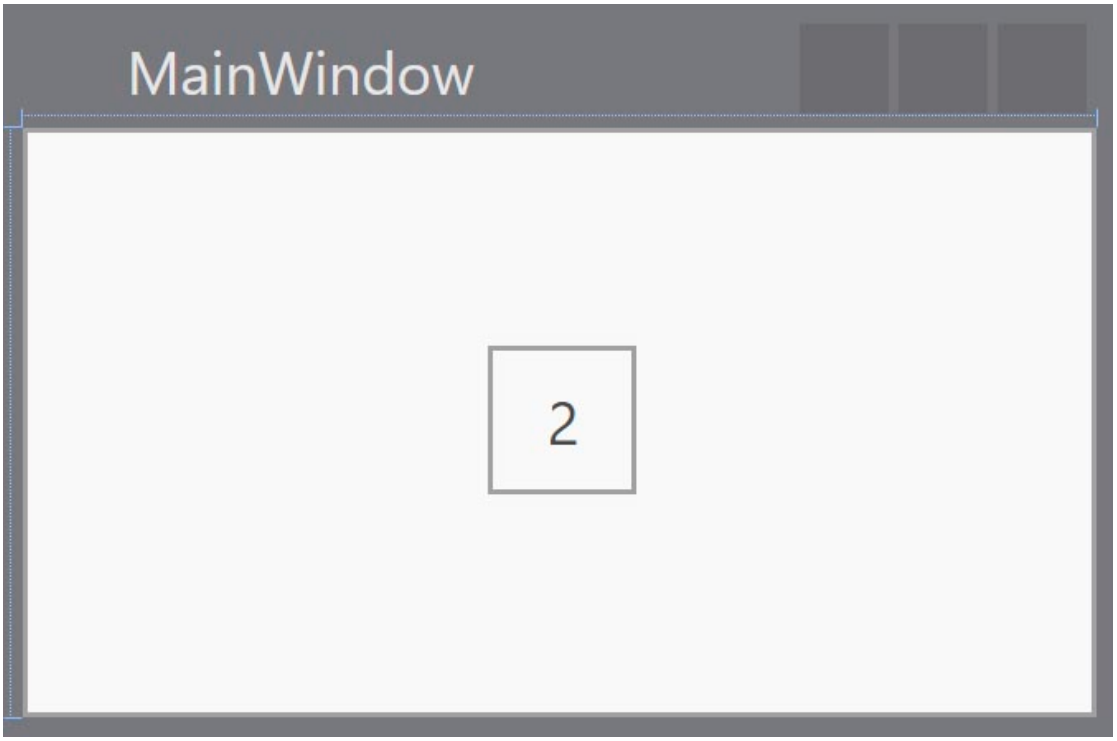
```
<Grid>
      <Button Content="1" />
      <Button Content="2" />
</Grid>
```




Тоді обидві кнопки розтягнуться на весь екран і займуть увесь вільний простір, але остання кнопка закриватиме попередню.

Зменшуємо другу кнопку

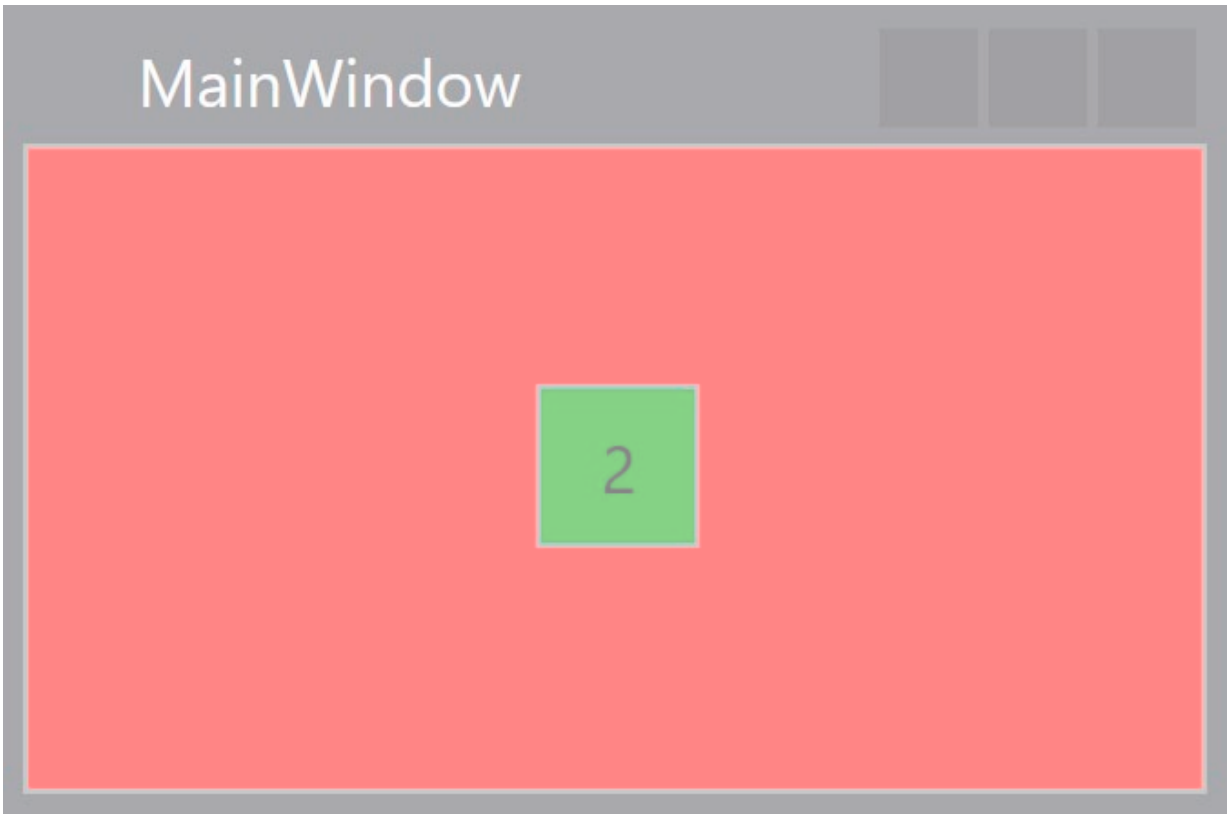
```
<Grid>
      <Button Content="1" />
      <Button Content="2"
              Width ="30"
              Height="30" />
</Grid>
```




Тепер за другою кнопкою видно першу кнопку.

# А давайте їх ще й пофарбуємо

```
<Grid>
      <Button Content="1"
              Background="Red" />
      <Button Content="2"
              Width ="30"
              Height="30"
              Background="Green" />
</Grid>
```

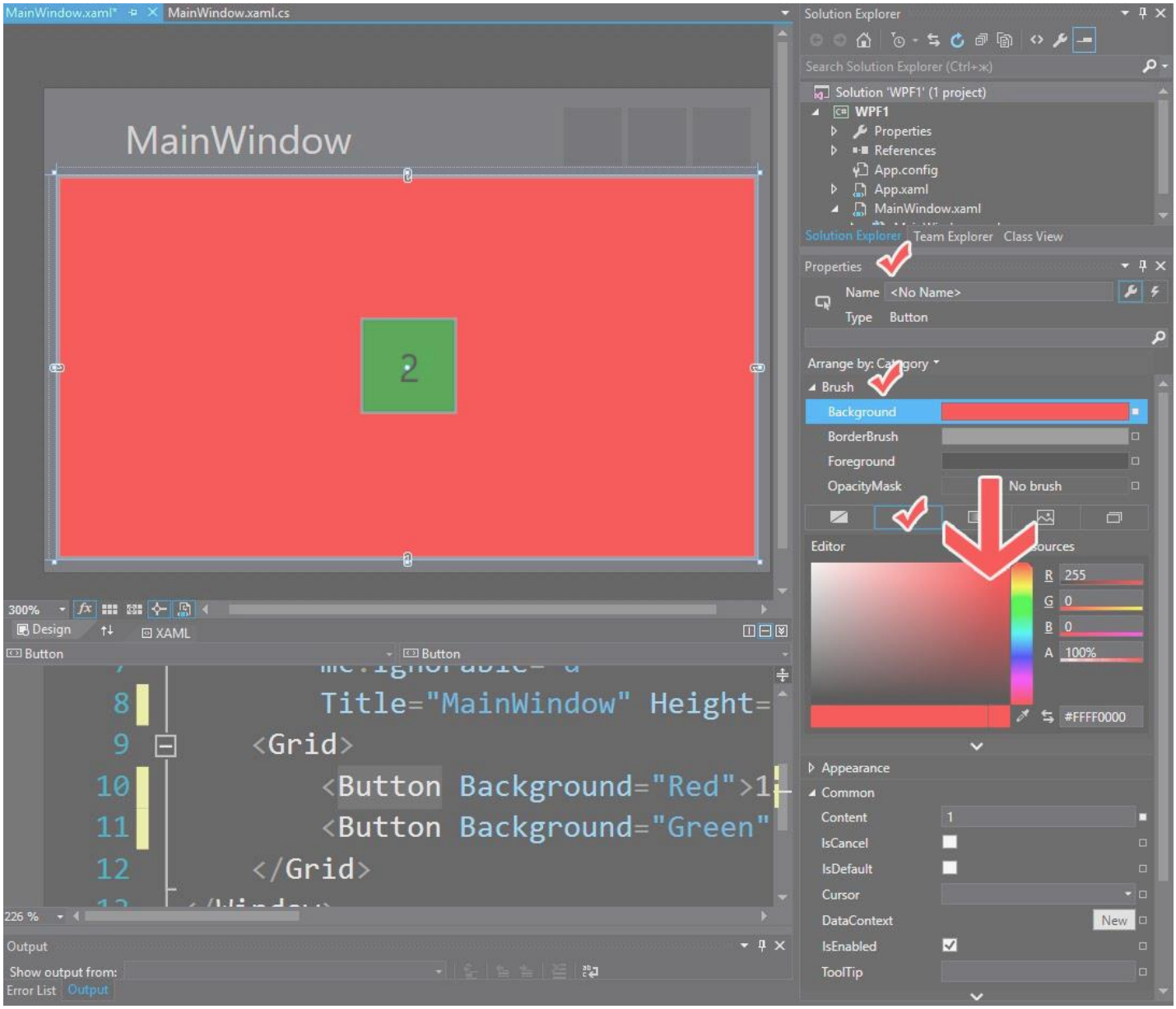


Для більшої зручності є спеціальна панель справа (Properties -> Brush) де можна обрати колір:

Фону – Background

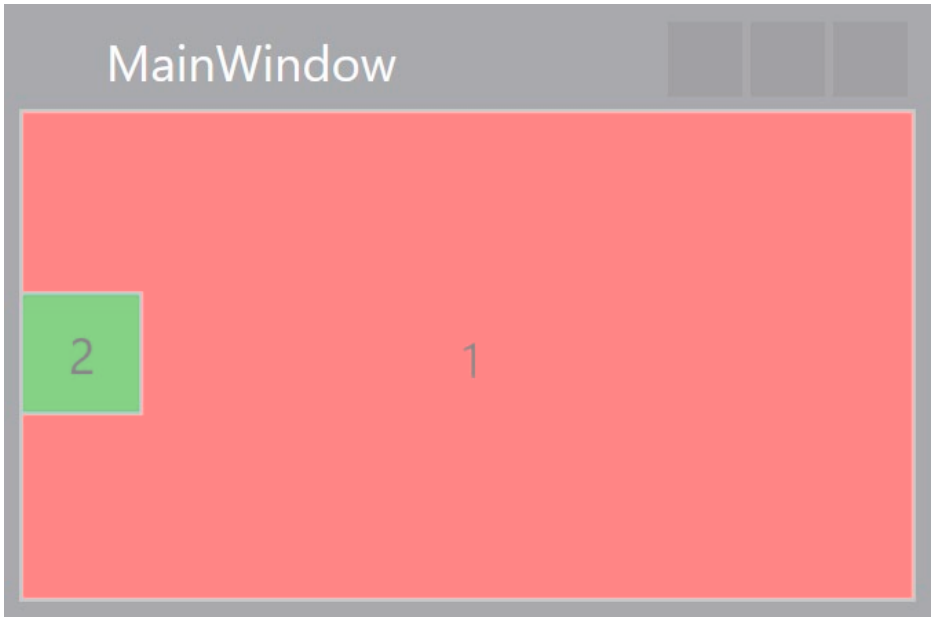
Рамки – BorderBrush

Тексту – Foreground



Типово кнопки розташовуються по центру сітки, але ми можемо їх вирівняти, як нам зручно:

```
<Grid>
    <Button Background="Red"
            Content="1" />
    <Button Background="Green"
            Width="30" Height="30"
            HorizontalAlignment="Left"
            Content="2" />
</Grid>
```

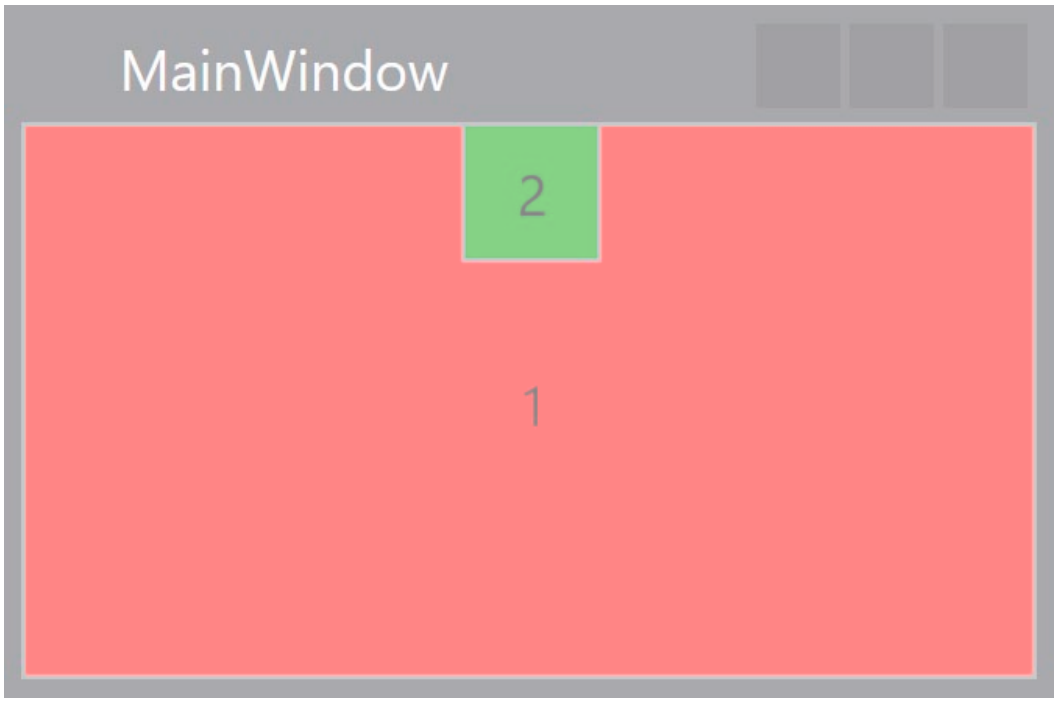



Left / Right / Center / Stretch

Зліва / Справа / По центру / Розтягнути

І по вертикалі:

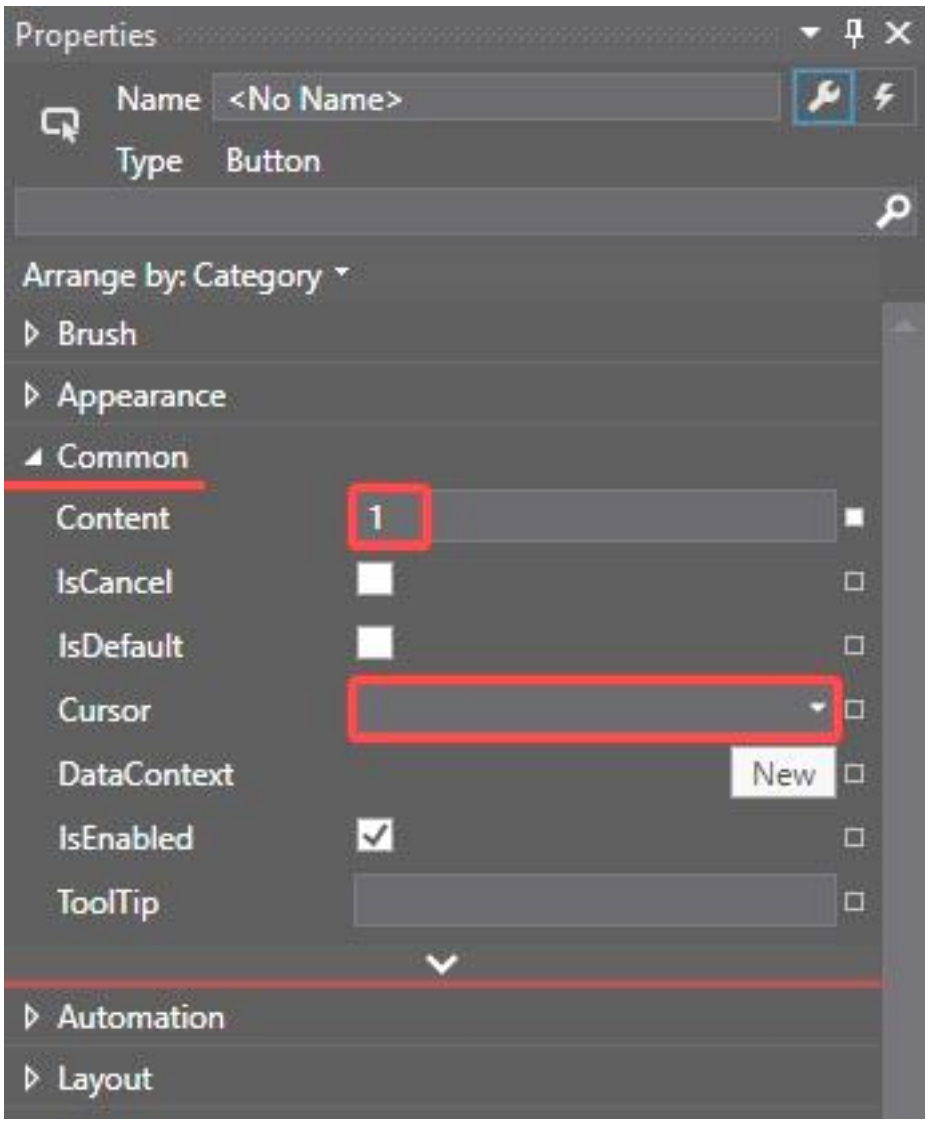
```
<Grid>
    <Button Background="Red"
            Content="1" />
    <Button Background="Green"
            Width="30" Height="30"
            VerticalAlignment="Top"
            Content="2" />
</Grid>
```




Top / Bottom / Center / Stretch

Вгору / Вниз / По центру / Розтягнути

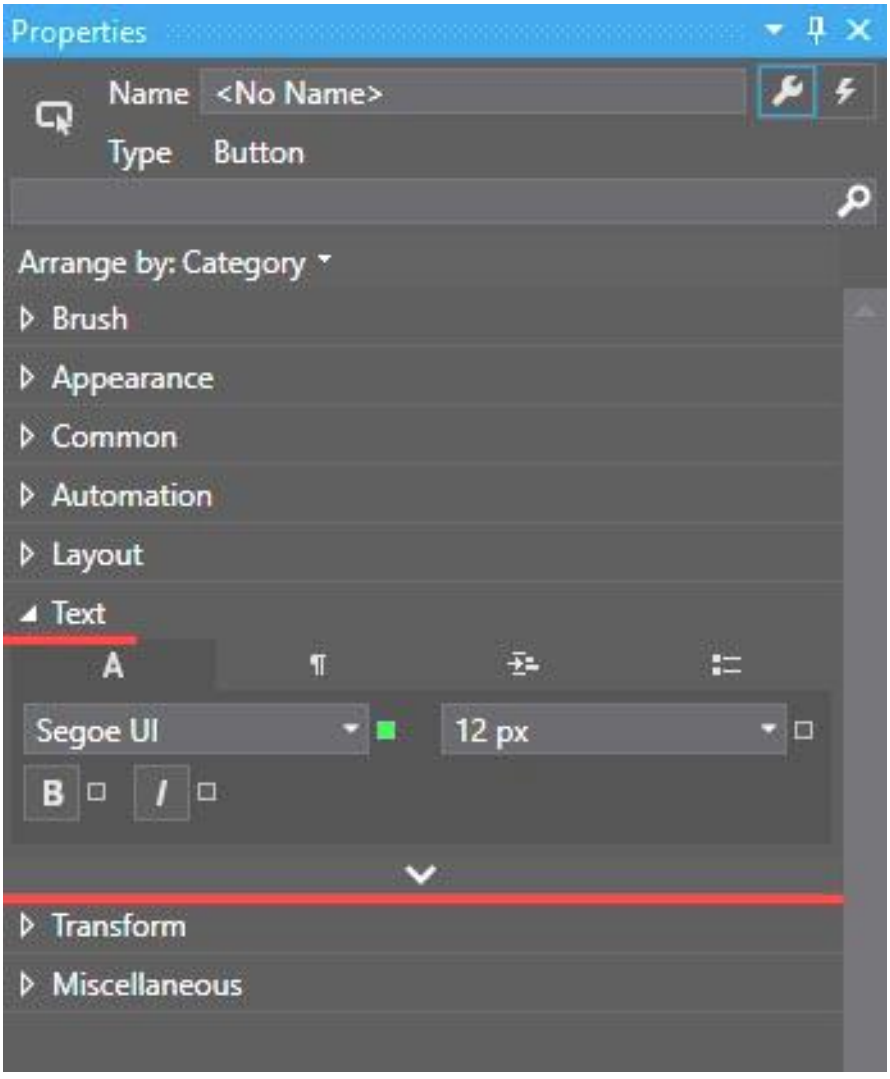
Текст кнопки можна змінювати в XAML, а можна у меню справа:




Також в полі курсор, можна обрати як буде виглядати курсор при наведенні на компоненту

А як змінити шрифт, розмір та інші властивості тексту?

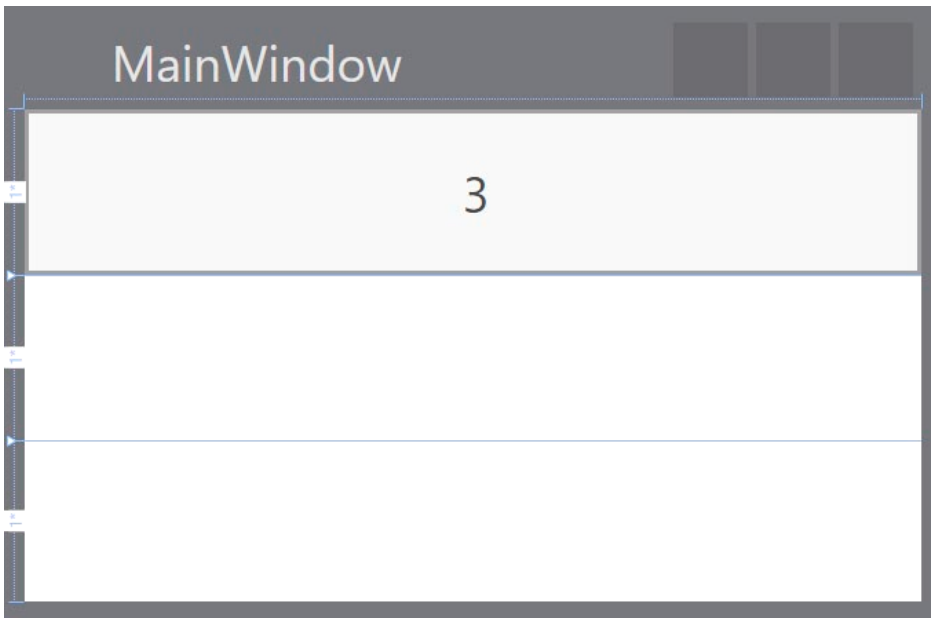
Для цього є окрема вкладка Text:



А як додати кілька кнопок, щоб вони стояли одна під одною?

Ага, ми маємо поділити нашу сітку на три частини:

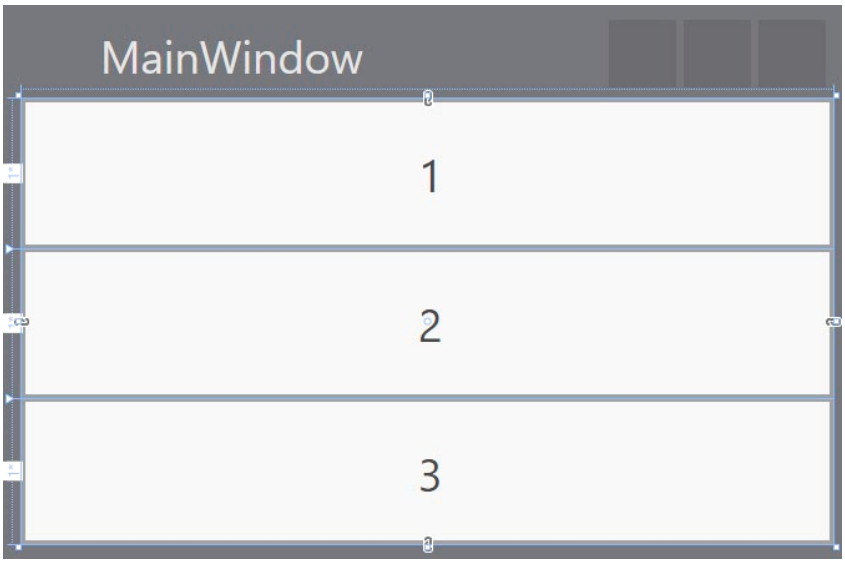
```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition />
        <RowDefinition />
        <RowDefinition />
    </Grid.RowDefinitions>

    <Button Content="1" />
    <Button Content="2" />
    <Button Content="3" />
</Grid>
```



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

Три RowDefinition в розмітці вказують на те, що в нас тепер є три рядки. Але оскільки ми не вказали нашим кнопкам,  в якому рядку кожна з них знаходиться – всі три кнопки автоматично попали в перший рядок.

```
<Button Content="1" Grid.Row="0" />
<Button Content="2" Grid.Row="1" />
<Button Content="3" Grid.Row="2" />
```

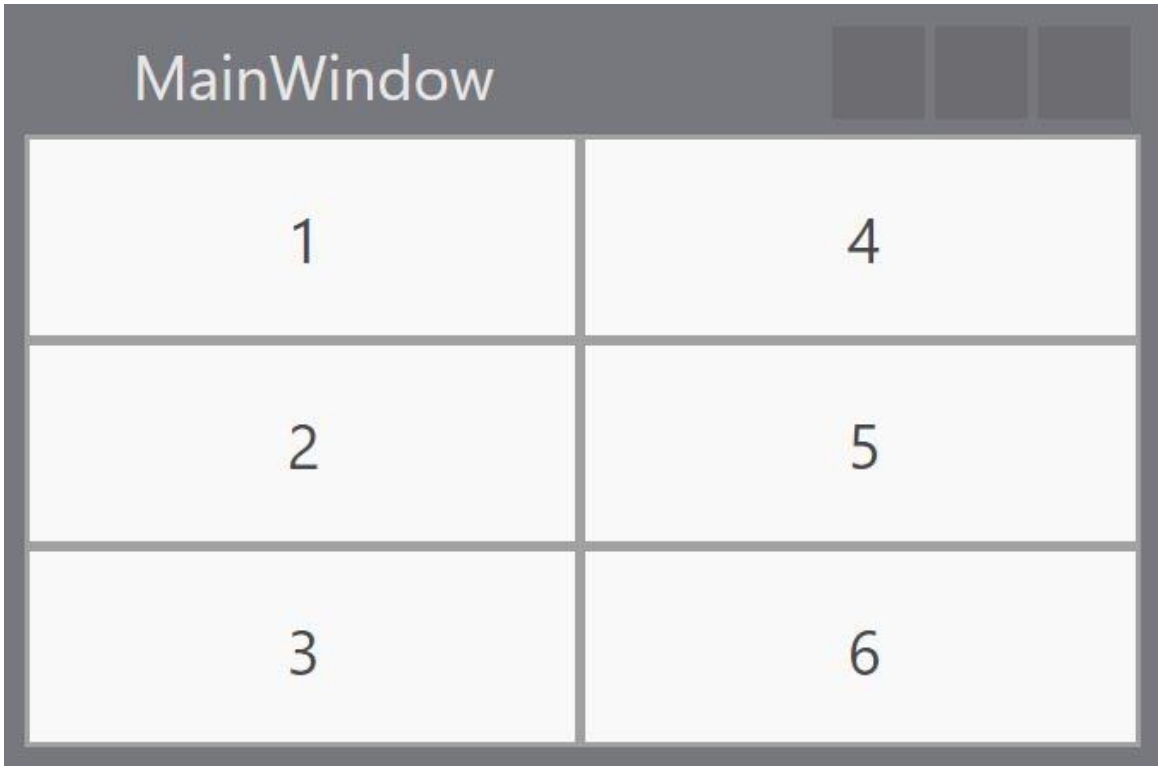


Командою `Grid.Row="1"` ми кажемо, що ця компонента буде знаходитися в другому рядку, бо нумерація в сітці починається з нуля, `0` – перший рядок

А як тоді додати стовпчики в сітку?

Невже існують команди `<Grid.ColumnDefinitions>` і `Grid.Column="1"`

Звісно ж існують!



# Тут 3 рядки і 2 стовпчики

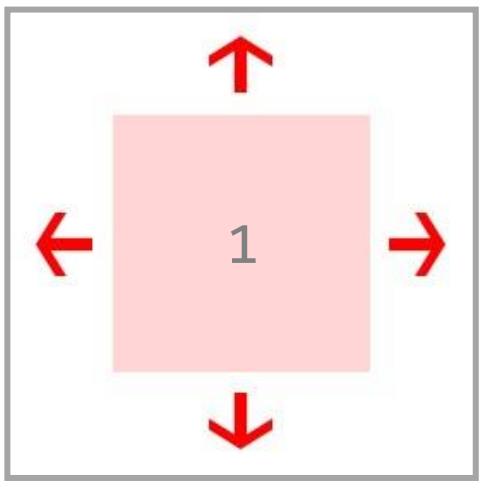
```
<Grid>
      <Grid.RowDefinitions>
          <RowDefinition />
          <RowDefinition />
          <RowDefinition />
      </Grid.RowDefinitions>
      <Grid.ColumnDefinitions>
          <ColumnDefinition />
          <ColumnDefinition />
      </Grid.ColumnDefinitions>

<Button Content="1" Grid.Row="0"
                    Grid.Column="0" />
<Button Content="2" Grid.Row="1"
                    Grid.Column="0" />
<Button Content="3" Grid.Row="2"
                    Grid.Column="0" />
<Button Content="4" Grid.Row="0"
                    Grid.Column="1" />
<Button Content="5" Grid.Row="1"
                    Grid.Column="1" />
<Button Content="6" Grid.Row="2"
                    Grid.Column="1" />

</Grid>
```

Якщо ми хочемо щось закоментувати, то в XAML це можна зробити так:

```
<!—
<Button Content="1" Margin="20" />
-->
```

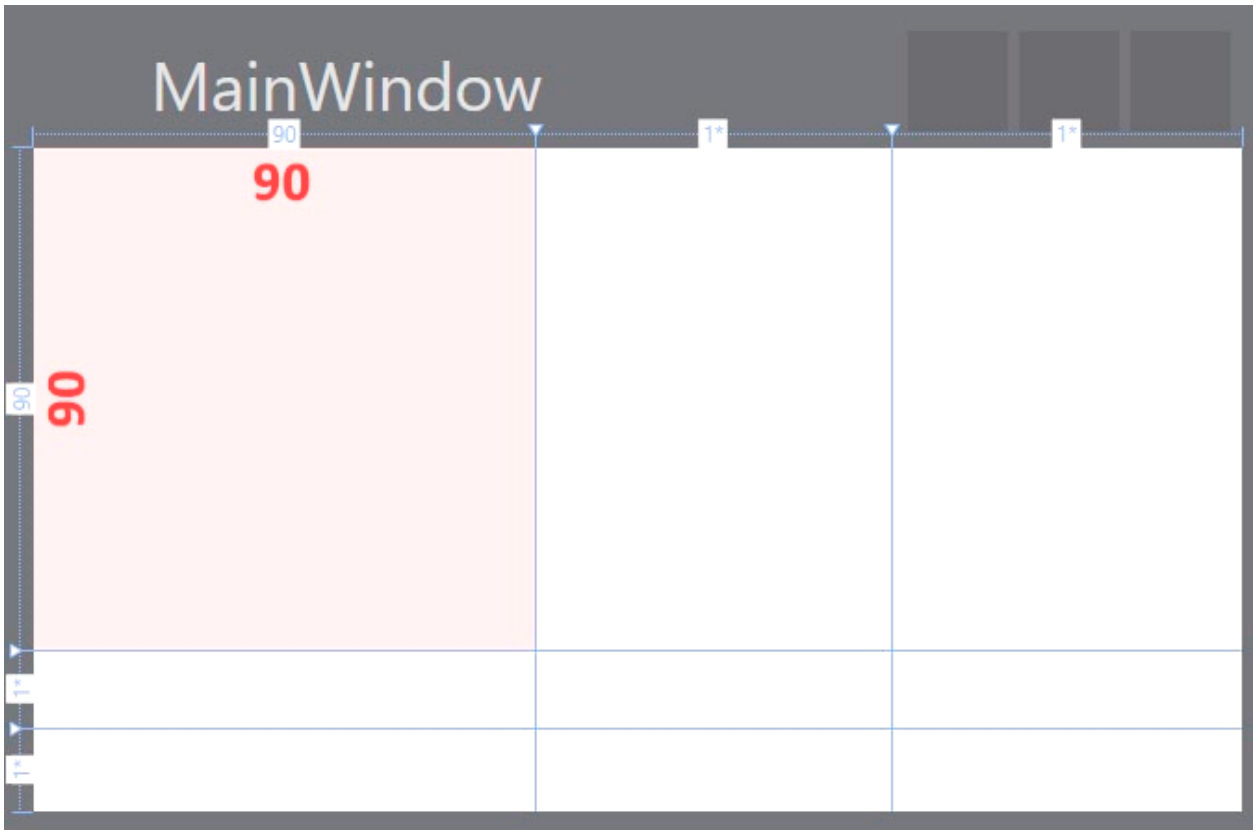
Цей елемент тепер не буде відображатися, наче ви його видалили.

А ще можна задати відштовхування компоненти від стінок сітки



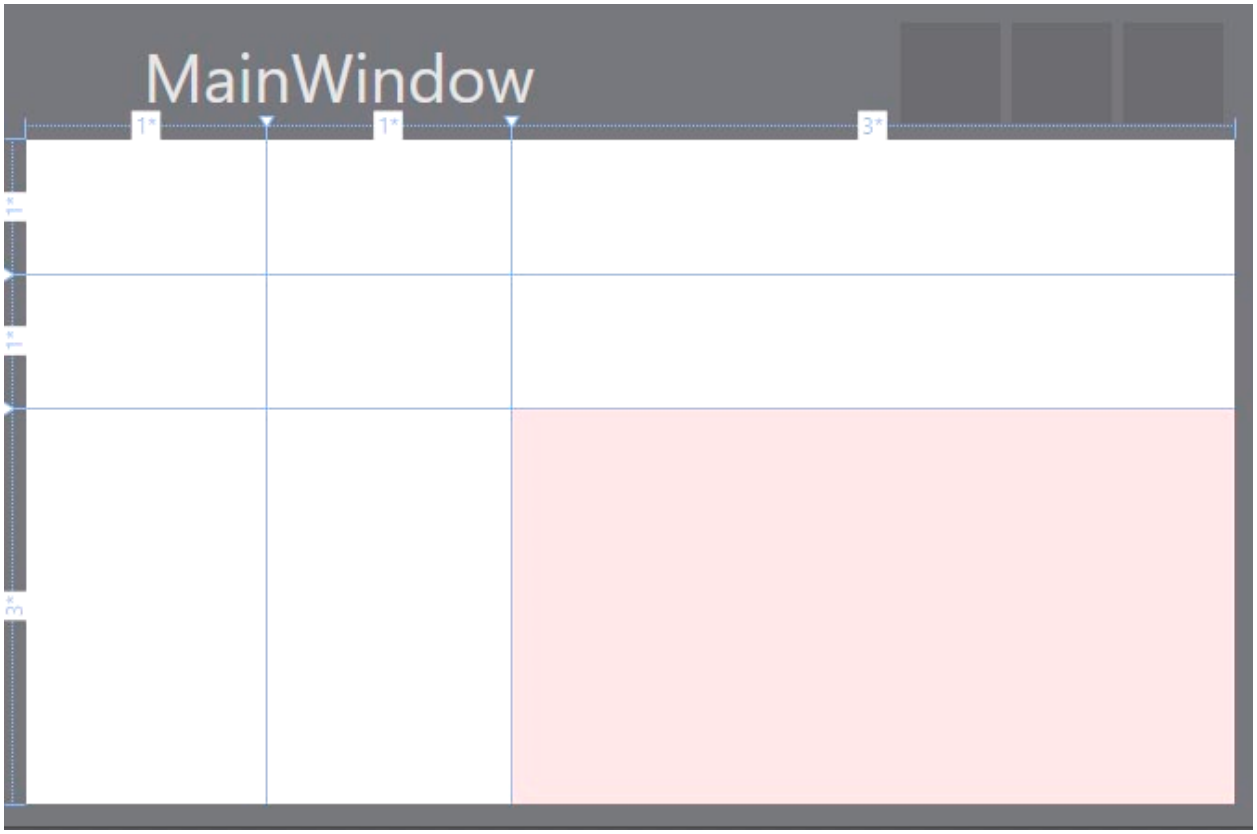

Margin="20"

# Також ми можемо задати висоту і ширину клітинок сітки




```
<Grid.RowDefinitions>
      <RowDefinition Height="90" />
      <RowDefinition />
      <RowDefinition />
</Grid.RowDefinitions>
<Grid.ColumnDefinitions>
      <ColumnDefinition Width="90" />
      <ColumnDefinition />
      <ColumnDefinition />
</Grid.ColumnDefinitions>
```

# Або задати висоту та ширину клітинок відносно сусідів:




```
<Grid.RowDefinitions>
      <RowDefinition Height="*" />
      <RowDefinition Height="*" />
      <RowDefinition Height="3*" />
</Grid.RowDefinitions>
<Grid.ColumnDefinitions>
      <ColumnDefinition Width="*" />
      <ColumnDefinition Width="*" />
      <ColumnDefinition Width="3*" />
</Grid.ColumnDefinitions>
```

Окрім кнопок можна в сітку вставляти текстовий блок:

```
<TextBlock Text="Мій текст" />
```

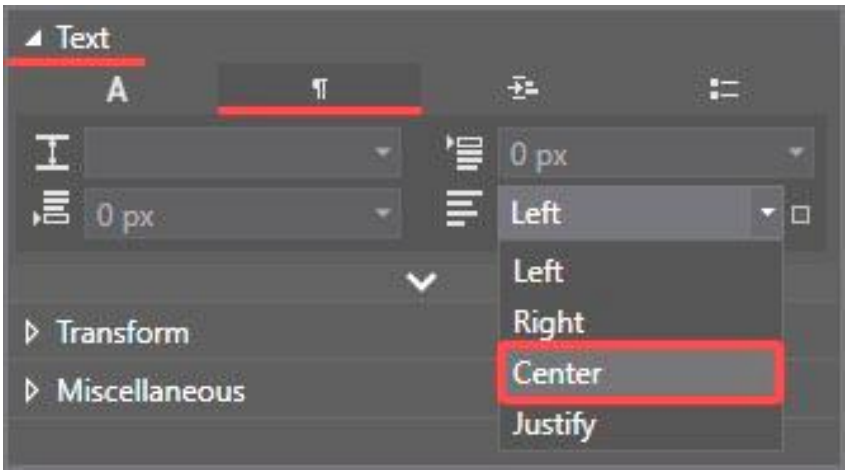
чи поле для вводу тексту:

```
<TextBox />
```

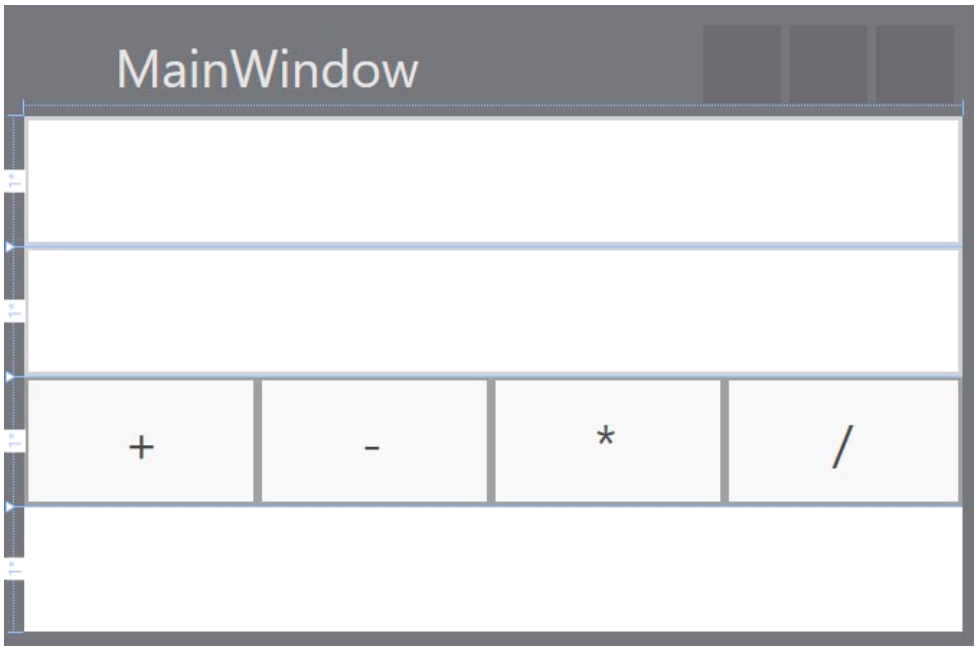
А текст всередині можна вирівняти

```
<TextBox Text="1" TextAlignment="Center" />
```

або ж зробити це в меню справа



# Час написати калькулятор! :)




Перші два рядки будуть текстовими полями для вводу цифр, а в четвертий рядок ми вставимо текстове поле для виводу результату.

Але як в третій рядок вставити чотири кнопки одна за одною?

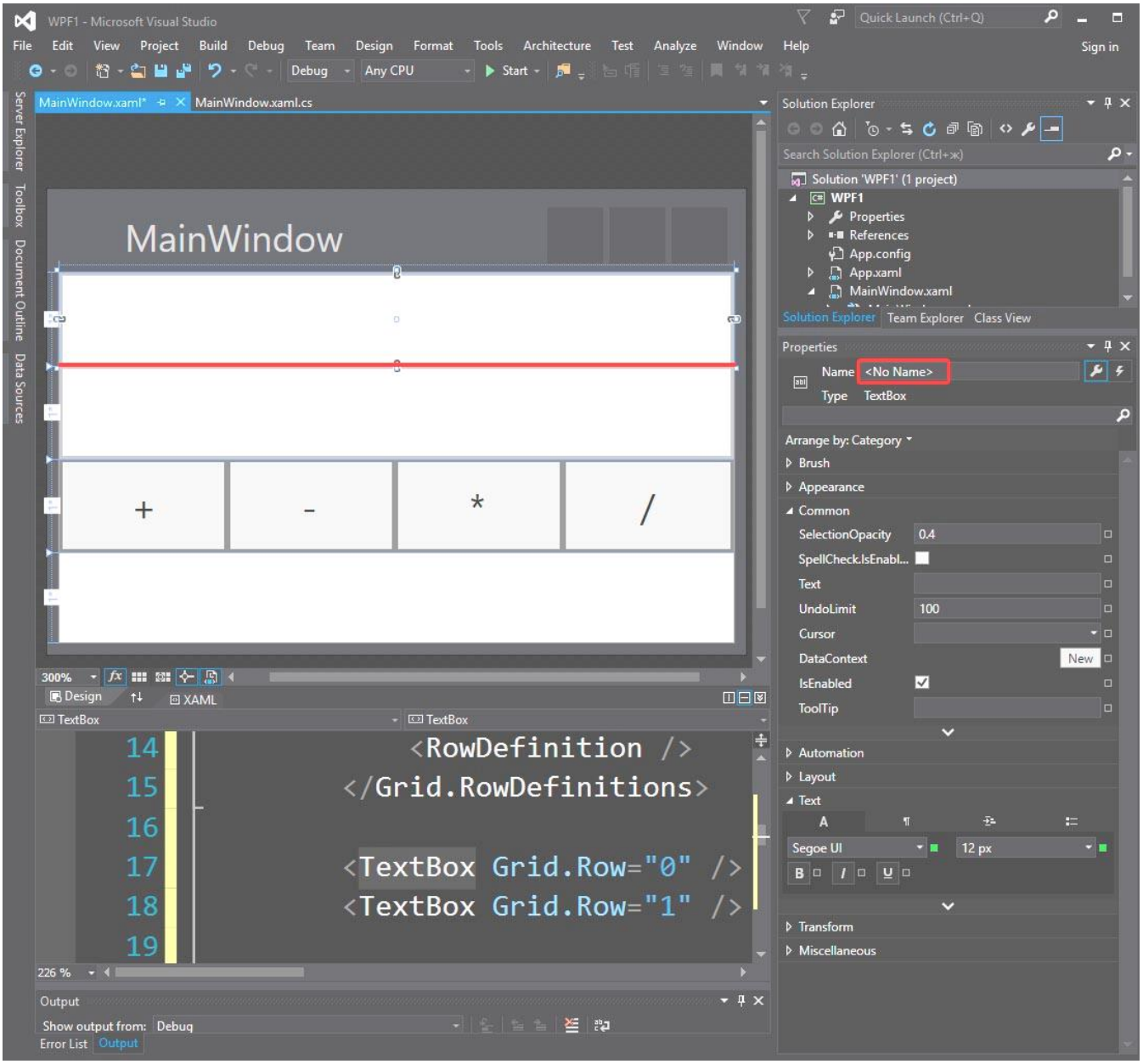
Дуже просто! Ми можемо в середину сітки вставити іншу сітку

```
<Grid>

    <Grid.RowDefinitions>
        <RowDefinition />
        <RowDefinition />
        <RowDefinition />
        <RowDefinition />
    </Grid.RowDefinitions>

    <TextBox Grid.Row="0" />
    <TextBox Grid.Row="1" />
    <Grid Grid.Row="2">
        <Grid.ColumnDefinitions>
            <ColumnDefinition />
            <ColumnDefinition />
            <ColumnDefinition />
            <ColumnDefinition />
        </Grid.ColumnDefinitions>

        <Button Content="+" Grid.Column="0" />
        <Button Content="-" Grid.Column="1" />
        <Button Content="*" Grid.Column="2" />
        <Button Content="/" Grid.Column="3" />
    </Grid>
    <TextBlock Grid.Row="3" />

</Grid>
```

Ок! Дизайн в нас є.

А як написати код C# для цих кнопок?

Спершу ми маємо дати назви компонентам, до яких ми будемо звертатися у коді



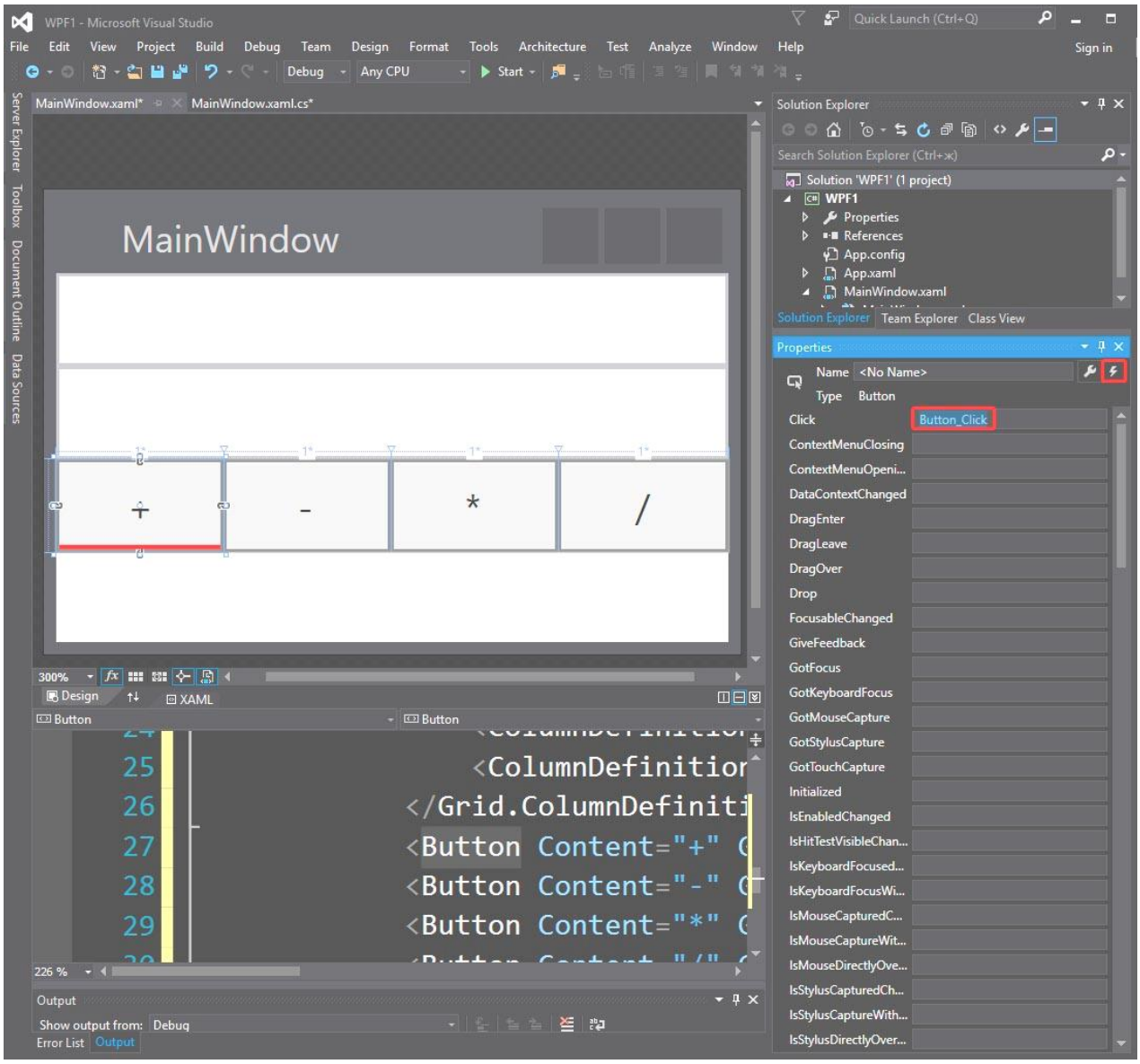

В полі Name першого поля для вводу пишемо tb1 (наприклад)

Аналогічно для другого поля вводу – tb2, і для останнього текстового поля пишемо tb3.

Потім в коді ми будемо використовувати ці назви, аби отримати значення в полях вводу і вивести результат обчислення в текстове поле знизу.

Ага, і даємо назви кнопкам:
b1, b2, b3, b4

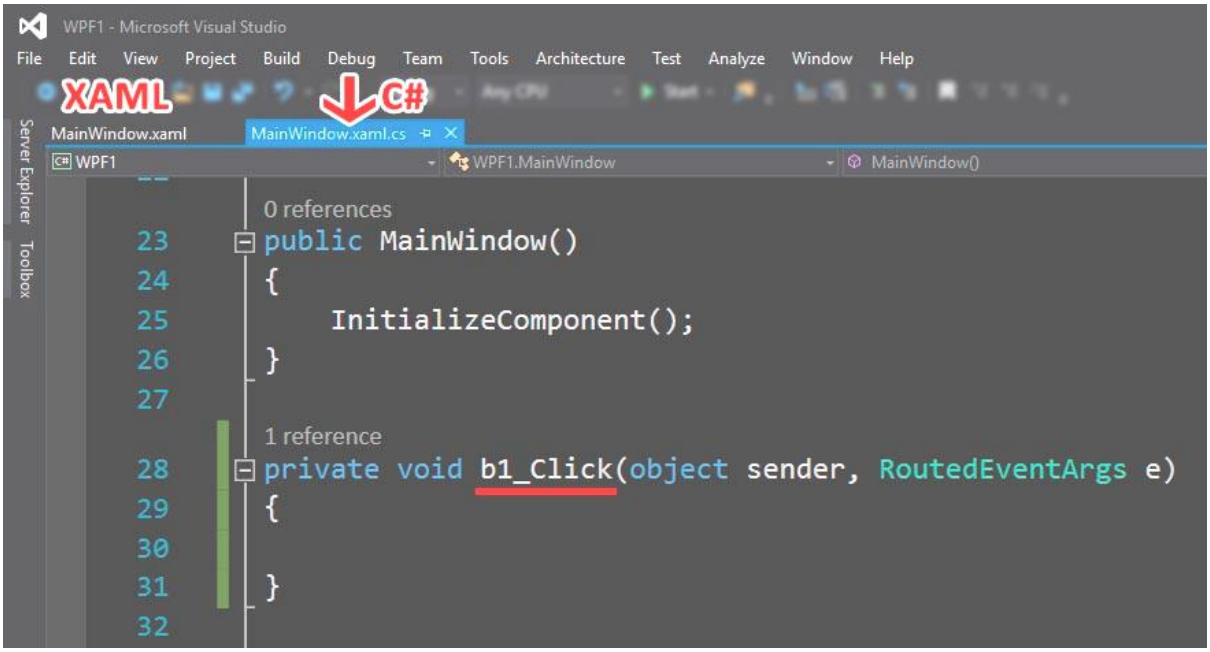

Назви кнопок потім допоможуть нам згенерувати красиві назви методів, які будуть до них прив'язані.




Беремо першу кнопку + і натискаємо на блискавку в меню справа.

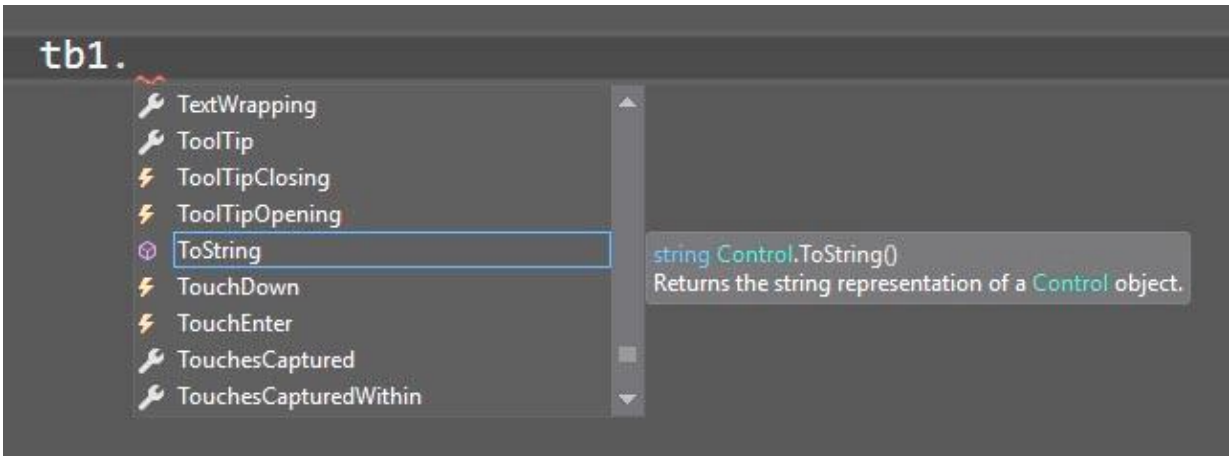
Шукаємо поле Click і робимо подвійний клік по ньому.

Вас має перекинути на вкладку з кодом.




Ви завжди можете побачити обидві вкладки вгорі, та за необхідності переключатися з дизайну в код і навпаки.

WPF зроблений так, щоб дизайн і код знаходилися окремо і це дуже зручно. Так ви можете скопіювати дизайн (XAML розмітку) з іншої програми і вставити собі в проект :)




Щоб взяти щось з якоїсь компоненти, ми просто пишемо її назву і ставимо крапу. Тоді з'являється список усіх її властивостей і ми обираємо ту, що нам потрібна. Наприклад tb1.Text – надає доступ до тексту, що знаходиться в текстовому полі.

От тепер ми можемо написати код для нашого методу b1_Click()

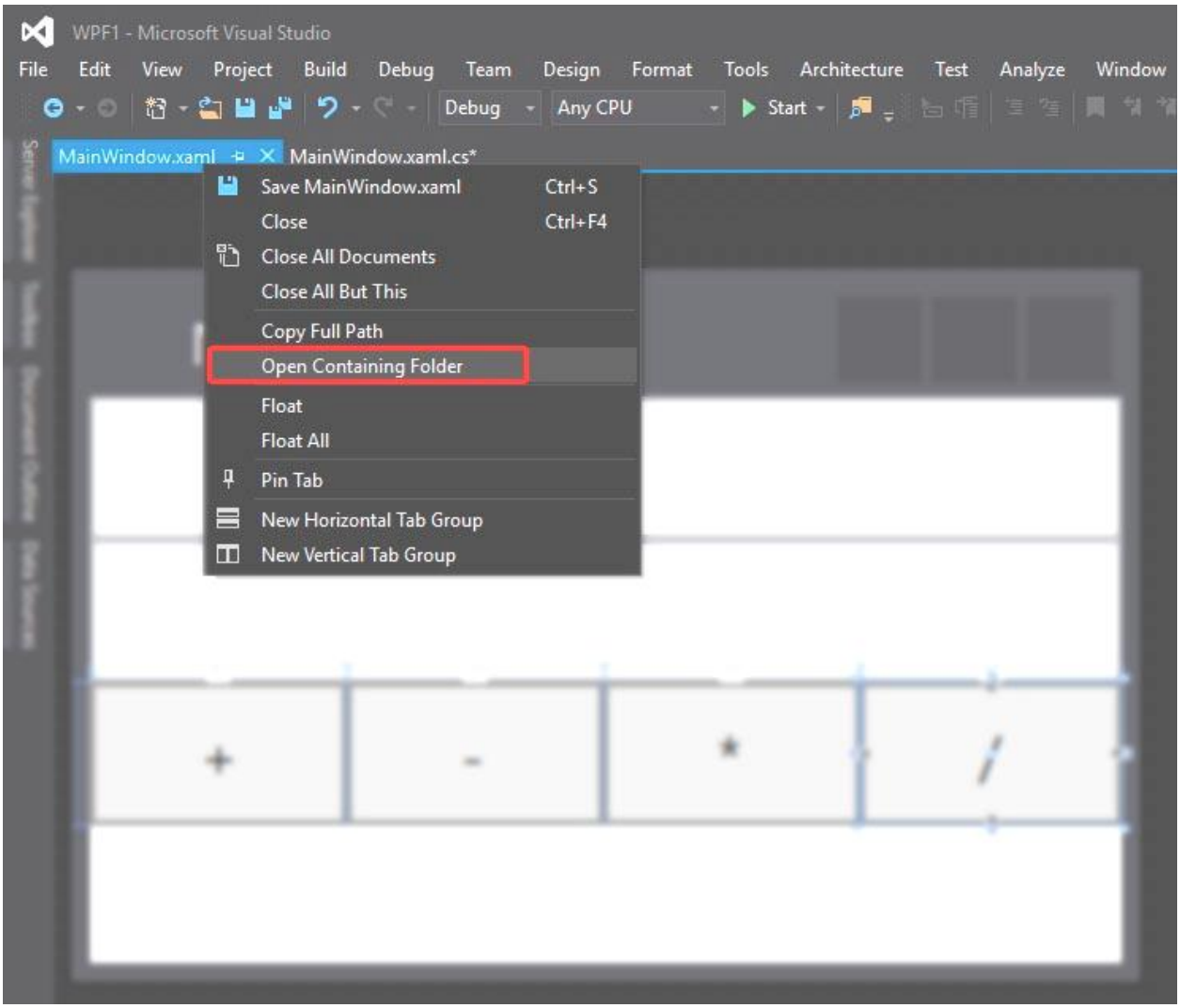
```
double a = Convert.ToDouble(tb1.Text);
double b = Convert.ToDouble(tb2.Text);
double c = a + b;
tb3.Text = c.ToString();
```

Аналогічно, використовуючи блискавку, ми генеруємо інші методи b2_Click(), b3_Click(), b4_Click():

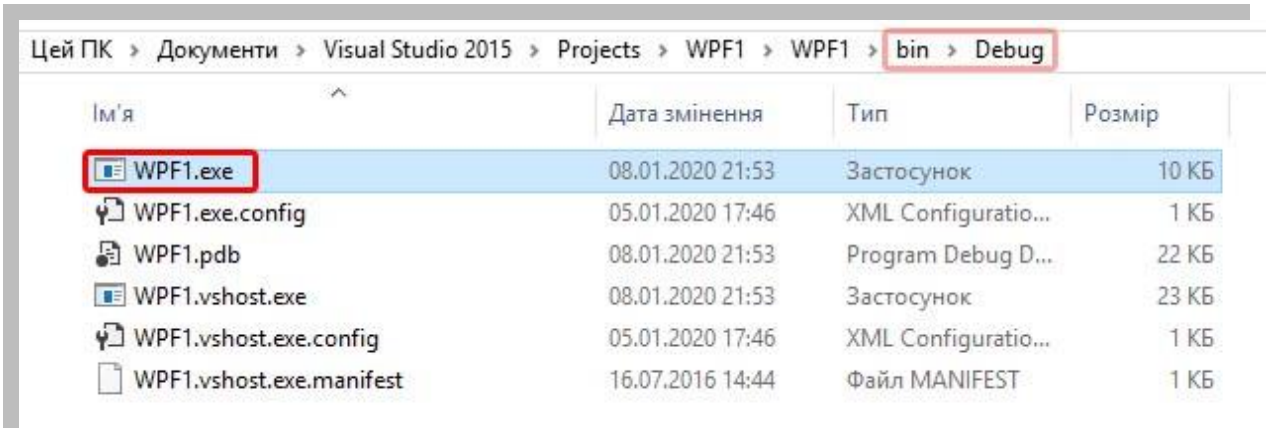
```
private void b2_Click(object sender, RoutedEventArgs e)
{
      double a = Convert.ToDouble(tb1.Text);
      double b = Convert.ToDouble(tb2.Text);
      double c = a - b;
      tb3.Text = c.ToString();
}
private void b3_Click(object sender, RoutedEventArgs e)
{
      double a = Convert.ToDouble(tb1.Text);
      double b = Convert.ToDouble(tb2.Text);
      double c = a * b;
      tb3.Text = c.ToString();
}
private void b4_Click(object sender, RoutedEventArgs e)
{
      double a = Convert.ToDouble(tb1.Text);
      double b = Convert.ToDouble(tb2.Text);
      double c = a / b;
      tb3.Text = c.ToString();
}
```

Запускаємо програму 


Visual Studio створює exe-файл і запускає його. Сам файл ви можете знайти, натиснувши правою кнопкою миші на одну з вкладок вгорі та обравши пункт "Відкрити папку з файлом"



Тепер треба зайти в папку `Bin`
а потім `Debug`




WPF1.exe - ваш exe-файл!

Інші файли допоміжні, вони були потрібні під час створення вашого файлу, тобто ми копіюємо собі на флешку лише WPF1.exe

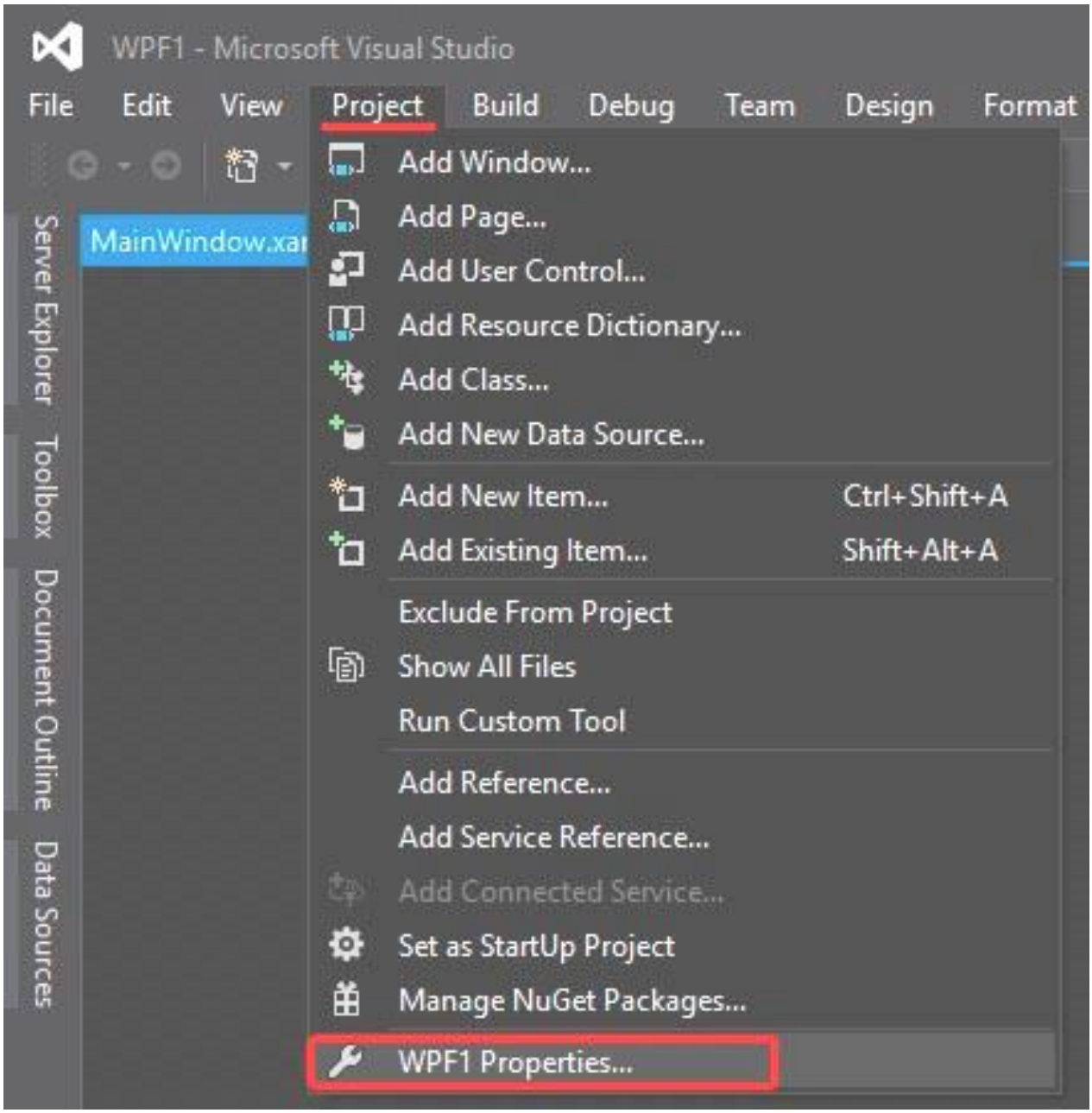
Все ок! Але наш файл без іконки.

Іконка, це маленька картинка
в форматі .ico її можна намалювати
прямо в інтернеті, написавши
в Гуглі "favicon":

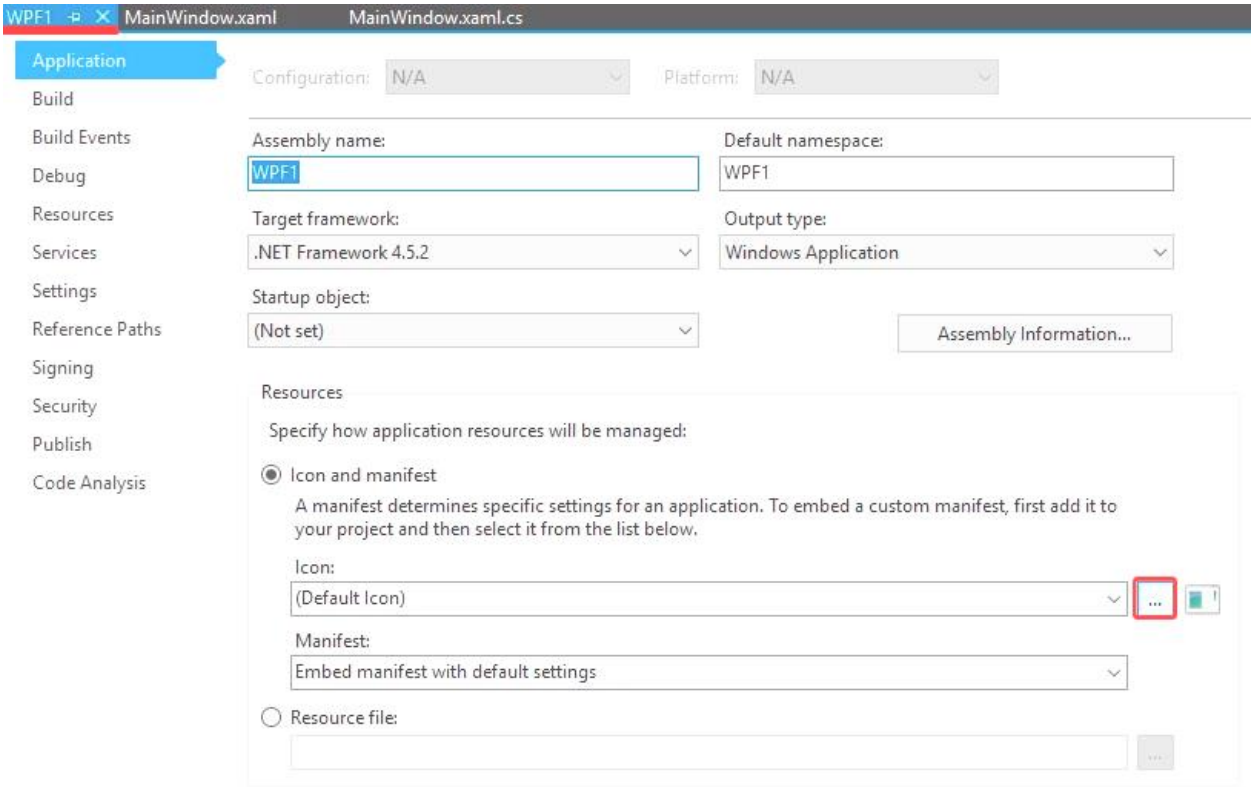
І підібрати сайт де можна намалювати
іконку, чи конвертувати картинку в
іконку, чи вибрати якусь з вже готових.

Для того, аби додати іконку на exe-файл, треба зайти в:

Проект –> Налаштування



Відкриється ще одна вкладка з налаштуваннями, тут треба натиснути на три крапки і обрати вашу іконку

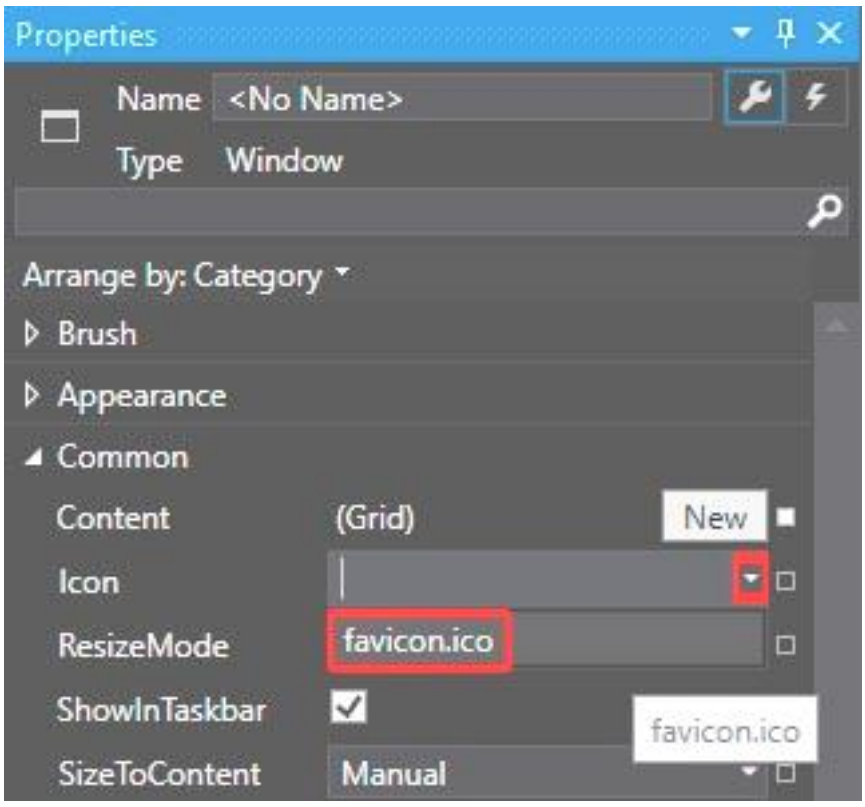



Час закрити цю вкладку, натиснувши на хрестик і знову запустити програму

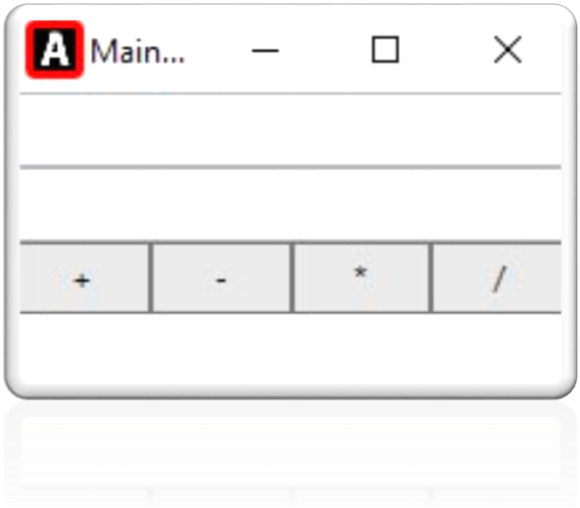

Тепер ваш файл має іконку

| | | | |
|---|---|---|---|
| WPF1.exe | 10.01.2020 4:59 | Застосунок | 11 КБ |
| WPF1.exe.config | 05.01.2020 17:46 | XML Configuratio... | 1 КБ |
| WPF1.pdb | 10.01.2020 4:59 | Program Debug D... | 22 КБ |
| WPF1.vshost.exe | 10.01.2020 4:59 | Застосунок | 23 КБ |
| WPF1.vshost.exe.config | 05.01.2020 17:46 | XML Configuratio... | 1 КБ |
| WPF1.vshost.exe.manifest | 16.07.2016 14:44 | Файл MANIFEST | 1 КБ |

Також можна зайти у вкладку XAML, обрати вікно <Windows …> і в загальних налаштуваннях Common натиснути на маленьку стрілочку вниз та вибрати вашу іконку.



Тепер у вікні програми завжди буде іконка




І на останок. Ви спитаєте, а як додати зображення на форму?

Просто перетягніть зображення на форму, видаліть лишні атрибути в XAML розмітці та вставте зображення у потрібну клітинку:

```
<Image Source="1.jpg" Grid.Row="3"/>
```

І ще одна річ, яку ми не розглянули – Глобальні змінні.

Давайте напишемо **Клікер**

Ми натискаємо на кнопку **+1**,
а в текстовому полі число збільшується на одиницю.

```
<Grid>
  <Grid.RowDefinitions>
    <RowDefinition Height="2*" />
    <RowDefinition Height="*" />
  </Grid.RowDefinitions>

    <TextBlock x:Name="tb1" Grid.Row="0"
      HorizontalAlignment="Center"
      VerticalAlignment="Center">1</TextBlock>

    <Button Grid.Row="1" Content="+1"
      Margin="5" Background="#FF00FF00"
      Click="Button_Click"/>

</Grid>
```

Ці команди розмітки роблять наступне:

`Height="2*"` – вдвічі більша клітинка

`x:Name="tb1"` – назва текстового блоку

`Margin="5"` – відступ з усіх боків

`Background="#FF00FF00"` – зелений

`Click="Button_Click"` – метод

Ось код програми:

```
int a = 1;

private void Button_Click(object
sender, RoutedEventArgs e)
{
      a++;
      tb1.Text = a.ToString();
}
```

Тут змінна a об'явлення над методом Button_Click(), що робить цю змінну глобальною.

Глобальна зміна доступна усім методам, а ще вона не втрачає своє значення після завершення роботи методу Button_Click()

# Тепер ви можете створювати програми з інтерфейсом