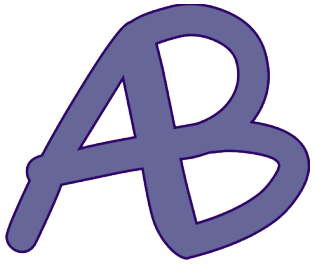
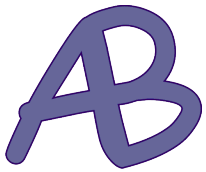


Wykład
WPF – Panel „Grid”



Panel „Grid”

- Może zawierać wiele rzędów i kolumn.
- Kolumnom można zdefiniować szerokość, a wierszom wysokość i to na kilka różnych sposobów: bezpośrednio w liczbie pikseli, w procencie dostępnego miejsca jak i w sposób automatyczny - gdzie rozmiar zostanie dostosowany w zależności od zawartości.

Panele: Grid

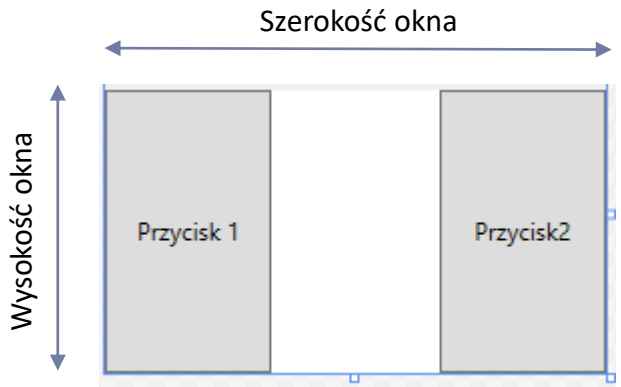
Podział na kolumny i przypisanie kontrolek do kolumn

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition />
        <ColumnDefinition />
        <ColumnDefinition />
    </Grid.ColumnDefinitions>
    <Button>Przycisk 1</Button>
    <Button Grid.Column="2">Przycisk2</Button>
</Grid>
```

Definicja kolumn (w tym przykładzie trzech)

Przypisanie kontrolki do kolumny.

Kolumny numerowane od 0 (0 jest wartością domyślną)



Panele: Grid

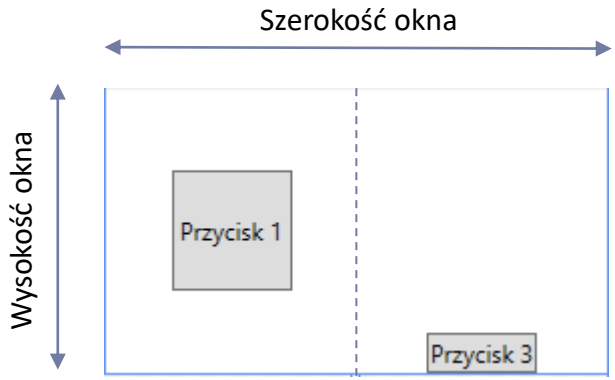
Rozmiar kontrolek i pozycjonowanie wewnątrz komórek

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition />
        <ColumnDefinition />
    </Grid.ColumnDefinitions>
    <Button Width="60px"
            Height="60px">
        Przycisk 1
    </Button>
    <Button Grid.Column="1"
            HorizontalAlignment="Center"
            VerticalAlignment="Bottom"
            >Przycisk 3
    </Button>
</Grid>
```

Kontrolka nie musi zajmować całej kolumny.

Możemy to osiągnąć poprzez:

- ustalenie rozmiaru kontrolki,
- pozycjonowanie kontrolki (rozmiar zostanie dopasowany automatycznie do zawartości)



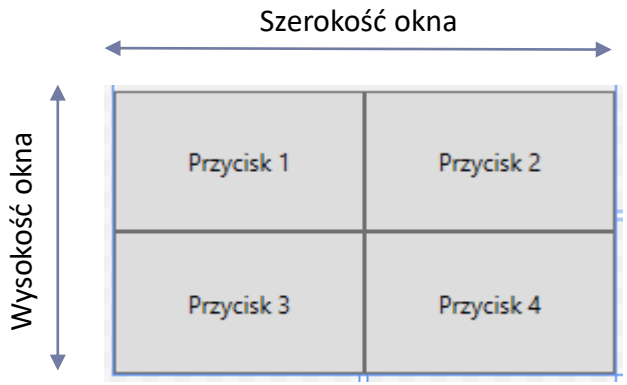
Panele: Grid

Komórki i wiersze (siatka)

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition />
        <ColumnDefinition />
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition />
        <RowDefinition />
    </Grid.RowDefinitions>
    <Button>Przycisk 1</Button>
    <Button Grid.Column="1" Grid.Row="0">
        Przycisk 2
    </Button>
    <Button Grid.Column="0" Grid.Row="1">
        Przycisk 3
    </Button>
    <Button Grid.Column="1" Grid.Row="1">
        Przycisk 4
    </Button>
</Grid>
```

Definiujemy liczbę kolumn oraz osobno liczbę wierszy.

Każdą kontrolkę przypisać należy do konkretnego wiersza i kolumny

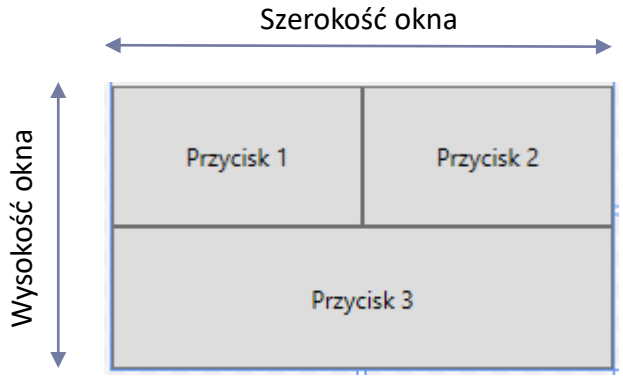


Panele: Grid

„Scalanie komórek” to właściwie rozciąganie kontrolki na sąsiadujące komórki

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition />
        <ColumnDefinition />
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition />
        <RowDefinition />
    </Grid.RowDefinitions>
    <Button>Przycisk 1</Button>
    <Button Grid.Column="1" Grid.Row="0">
        Przycisk 2
    </Button>
    <Button Grid.Column="0" Grid.Row="1" Grid.ColumnSpan="2">
        Przycisk 3
    </Button>
```

Własność `Grid.ColumnSpan="2”` oraz `Grid.RowSpan="2"` służą do rozciągania kontrolki na sąsiadujące ze sobą pola (odpowiednio na 2 kolumny oraz na 2 wiersze).



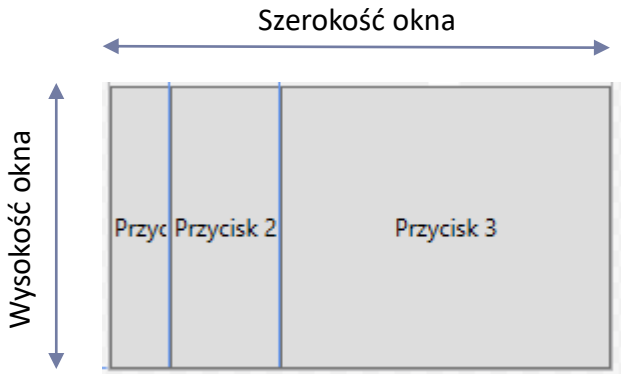
Panele: Grid

Szerokość komórek

Mamy do dyspozycji trzy metody definiowania szerokości komórek (kolumn)

1. Szerokość stała - podawana w pikselach
2. Automatyczne dopasowanie szerokości do zawartości komórki (Auto)
3. Szerokość proporcjonalna „system gwiazdek”

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="30px"/>
        <ColumnDefinition Width="Auto"/>
        <ColumnDefinition Width="*"/>
    </Grid.ColumnDefinitions>
    <Button>Przycisk 1</Button>
    <Button Grid.Column="1" >
        Przycisk 2
    </Button>
    <Button Grid.Column="2"  >
        Przycisk 3
    </Button>
</Grid>
```



Panele: Grid

Szerokość komórek – system gwiazdek

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="3*"/>
        <ColumnDefinition Width="2*"/>
        <ColumnDefinition Width="*"/>
    </Grid.ColumnDefinitions>
    <Button>Przycisk 1</Button>
    <Button Grid.Column="1" >
        Przycisk 2
    </Button>
    <Button Grid.Column="2"  >
        Przycisk 3
    </Button>
</Grid>
```

System gwiazdek pozwala na przydzielanie dla kolumn odpowiedniej „wagi”.

W tym przykładzie szerokość okna dzielona jest na 6 (bo tyle jest łącznie gwiazdek)

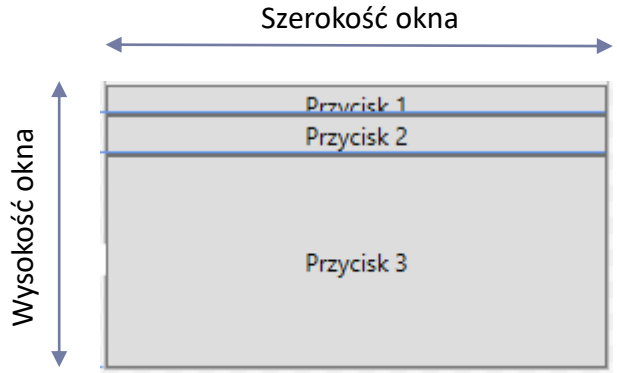
- pierwszej kolumnie przyznane jest 3/6 szerokości okna (trzy gwiazdki)
- Drugiej kolumnie 2/6 (dwie gwiazdki)
- Trzeciej kolumnie 1/6 (jedna gwiazdka)

Panele: Grid

Wysokość wierszy

Sterowanie wysokością wierszy działa analogicznie, z tym że używamy atrybutu `Height=""`

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="15px"/>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="*"/>
    </Grid.RowDefinitions>
    <Button>Przycisk 1</Button>
    <Button Grid.Row="1" >
        Przycisk 2
    </Button>
    <Button Grid.Row="2"  >
        Przycisk 3
    </Button>
</Grid>
```



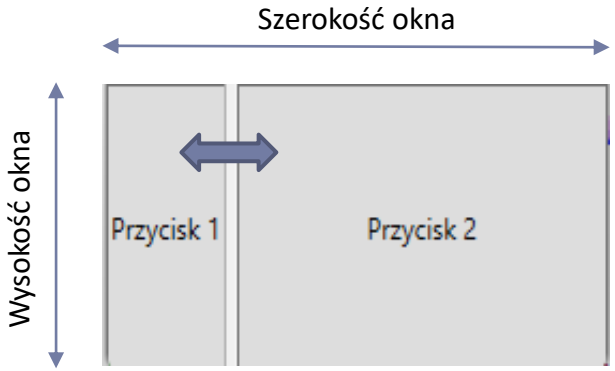
Układzie siatki możemy sterować zarówno szerokością kolumn jak i wysokością wierszy

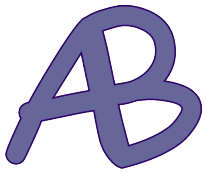
Panele: GridSplitter

GridSplitter – Kontrolka pozwalająca użytkownikowi zmieniać szerokości kolumn i wysokości wierszy w trakcie działania programu

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="*"/>
        <ColumnDefinition Width="3" />
        <ColumnDefinition Width="*"/>
    </Grid.ColumnDefinitions>
    <Button>Przycisk 1</Button>
    <GridSplitter Grid.Column="1" Width="3" HorizontalAlignment="Stretch" />
    <Button Grid.Column="2" >
        Przycisk 2
    </Button>
</Grid>
```

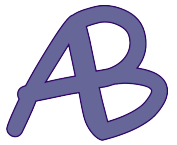
GrydSplitter jest kontrolką którą umieszczamy w pustej, wąskiej kolumnie lub wierszu.
Dzięki niej użytkownik może „chwycić” za krawędź i przeciągnąć ją na odpowiednią pozycję





Grafika w WPF – cz. 1
kolory i pędzle

Grafika: Color i Brush



Do tworzenia grafiki WPF potrzebne są klasy Color oraz Brush

Klasa kolor (Color) służy do składania kolorów i operacji na nich.

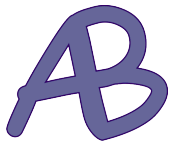
Klasa pędzel (Brush) służy do wypełniania obszaru wcześniej zdefiniowanym kolorem lub wzorem. Niektóre pędzle malują obszar pełnymi kolorami, inne gradientem, wzorcem lub obrazem.

Dodatkowo dostępne są klasy Colors oraz Brushes – są to kolekcje predefiniowanych kolorów oraz pędzli (klasy statyczne)

AliceBlue	#FFF0F8FF	DarkTurquoise	#FF00CED1	LightSeaGreen	#FF20B2AA	PapayaWhip	#FFFFEFD5
AntiqueWhite	#FFFAEBD7	DarkViolet	#FF9400D3	LightSkyBlue	#FF87CEFA	PeachPuff	#FFFFDAB9
Aqua	#FF00FFFF	DeepPink	#FFFF1493	LightSlateGray	#FF778899	Peru	#FFCD853F
Aquamarine	#FF7FFFD4	DeepSkyBlue	#FF00BFFF	LightSteelBlue	#FFB0C4DE	Pink	#FFFFC0CB
Azure	#FFF0FFFF	DimGray	#FF696969	LightYellow	#FFFFFFE0	Plum	#FFDDA0DD
Beige	#FFF5F5DC	DodgerBlue	#FF1E90FF	Lime	#FF00FF00	PowderBlue	#FFB0E0E6
Bisque	#FFFFE4C4	Firebrick	#FFB22222	LimeGreen	#FF32CD32	Purple	#FF800080
Black	#FF000000	FloralWhite	#FFFFFAF0	Linen	#FFFAF0E6	Red	#FFFF0000
BlanchedAlmond	#FFFFEBCD	ForestGreen	#FF228B22	Magenta	#FFFF00FF	RosyBrown	#FFBC8F8F
Blue	#FF0000FF	Fuchsia	#FFFF00FF	Maroon	#FF800000	RoyalBlue	#FF4169E1
BlueViolet	#FF8A2BE2	Gainsboro	#FFDCDCDC	MediumAquamarine	#FF66CDAA	SaddleBrown	#FF8B4513
Brown	#FFA52A2A	GhostWhite	#FFF8F8FF	MediumBlue	#FF0000CD	Salmon	#FFFA8072
BurlyWood	#FFDEB887	Gold	#FFFFD700	MediumOrchid	#FFBA55D3	SandyBrown	#FFF4A460
CadetBlue	#FF5F9EA0	Goldenrod	#FFDAA520	MediumPurple	#FF9370DB	SeaGreen	#FF2E8B57
Chartreuse	#FF7FFF00	Gray	#FF808080	MediumSeaGreen	#FF3CB371	SeaShell	#FFFFF5EE
Chocolate	#FFD2691E	Green	#FF008000	MediumSlateBlue	#FF7B68EE	Sienna	#FFA0522D
Coral	#FFFF7F50	GreenYellow	#FFADFF2F	MediumSpringGreen	#FF00FA9A	Silver	#FFC0C0C0
CornflowerBlue	#FF6495ED	Honeydew	#FFF0FFF0	MediumTurquoise	#FF48D1CC	SkyBlue	#FF87CEEB
Cornsilk	#FFFFF8DC	HotPink	#FFFF69B4	MediumVioletRed	#FFC71585	SlateBlue	#FF6A5ACD
Crimson	#FFDC143C	IndianRed	#FFCD5C5C	MidnightBlue	#FF191970	SlateGray	#FF708090
Cyan	#FF00FFFF	Indigo	#FF4B0082	MintCream	#FFF5FFFA	Snow	#FFFFFAFA
DarkBlue	#FF00008B	Ivory	#FFFFFFF0	MistyRose	#FFFFE4E1	SpringGreen	#FF00FF7F
DarkCyan	#FF008B8B	Khaki	#FFF0E68C	Moccasin	#FFFFE4B5	SteelBlue	#FF4682B4
DarkGoldenrod	#FFB8860B	Lavender	#FFE6E6FA	NavajoWhite	#FFFFDEAD	Tan	#FFD2B48C
DarkGray	#FFA9A9A9	LavenderBlush	#FFFFF0F5	Navy	#FF000080	Teal	#FF008080
DarkGreen	#FF006400	LawnGreen	#FF7CFC00	OldLace	#FFFDF5E6	Thistle	#FFD8BFD8
DarkKhaki	#FFBDB76B	LemonChiffon	#FFFFFACD	Olive	#FF808000	Tomato	#FFFF6347
DarkMagenta	#FF8B008B	LightBlue	#FFADD8E6	OliveDrab	#FF6B8E23	Transparent	#00FFFFFF
DarkOliveGreen	#FF556B2F	LightCoral	#FFF08080	Orange	#FFFFA500	Turquoise	#FF40E0D0
DarkOrange	#FFFF8C00	LightCyan	#FFE0FFFF	OrangeRed	#FFFF4500	Violet	#FFEE82EE
DarkOrchid	#FF9932CC	LightGoldenrodYellow	#FFFAFAD2	Orchid	#FFDA70D6	Wheat	#FFF5DEB3
DarkRed	#FF8B0000	LightGray	#FFD3D3D3	PaleGoldenrod	#FFEEE8AA	White	#FFFFFFFF
DarkSalmon	#FFE9967A	LightGreen	#FF90EE90	PaleGreen	#FF98FB98	WhiteSmoke	#FFF5F5F5
DarkSeaGreen	#FF8FBC8F	LightPink	#FFFFB6C1	PaleTurquoise	#FFAFEEEE	Yellow	#FFFFFF00
DarkSlateBlue	#FF483D8B	LightSalmon	#FFFFA07A	PaleVioletRed	#FFDB7093	YellowGreen	#FF9ACD32
DarkSlateGray	#FF2F4F4F						

Źródło: https://docs.microsoft.com

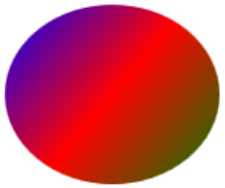
Grafika: Color i Brush



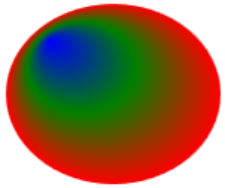
Przykłady tworzenia różnych pędzli

```
SolidColorBrush brush_00 = new SolidColorBrush(Colors.AliceBlue);
brush_00.Color = Color.FromRgb(255, 0, 0);
```

```
LinearGradientBrush brush_01 = new LinearGradientBrush();
brush_01.GradientStops.Add(new GradientStop(Colors.Blue, 0.0));
brush_01.GradientStops.Add(new GradientStop(Colors.Red, 0.5));
brush_01.GradientStops.Add(new GradientStop(Colors.Green, 1.0));
```



```
RadialGradientBrush brush_02 = new RadialGradientBrush();
brush_02.GradientOrigin = new Point(0.2, 0.2);
brush_02.GradientStops.Add(new GradientStop(Colors.Blue, 0.0));
brush_02.GradientStops.Add(new GradientStop(Colors.Green, 0.5));
brush_02.GradientStops.Add(new GradientStop(Colors.Red, 1.0));
```



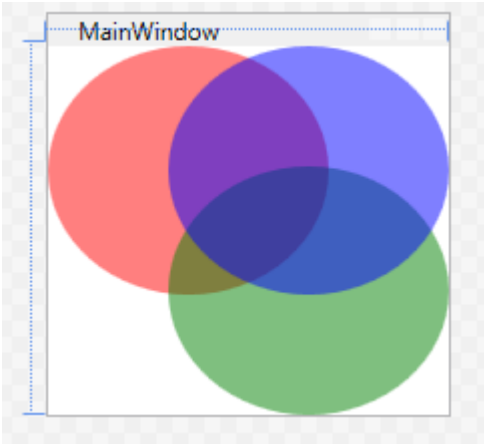
```
elipsa.Fill = brush_00;
```

Wypełnianie kontrolki przygotowanym pędzlem

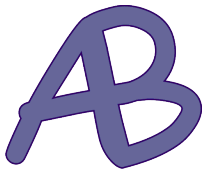
Grafika: Color i Brush

Wypełnienia możemy definiować także bezpośrednio z poziomu kodu XAML. Wypełnienie definiujemy j jako znacznik podrzędny kontrolki, a w nim kolejny znacznik podrzędny - pędzel który używamy do wypełnienia wraz z jego właściwościami (w tym przykładzie kolor i przezroczystość)

```
<Ellipse
    x:Name="elipsa" Margin="0,0,60,60">
    <Ellipse.Fill>
        <SolidColorBrush Color="Red" Opacity="0.50" />
    </Ellipse.Fill>
</Ellipse>
<Ellipse  x:Name="elipsa_2" Margin="60,60,0,0">
    <Ellipse.Fill>
        <SolidColorBrush Color="Green" Opacity="0.50" />
    </Ellipse.Fill>
</Ellipse>
<Ellipse x:Name="elipsa_3" Margin="60,0,0,60">
    <Ellipse.Fill>
        <SolidColorBrush Color="Blue" Opacity="0.50" />
    </Ellipse.Fill>
</Ellipse>
```

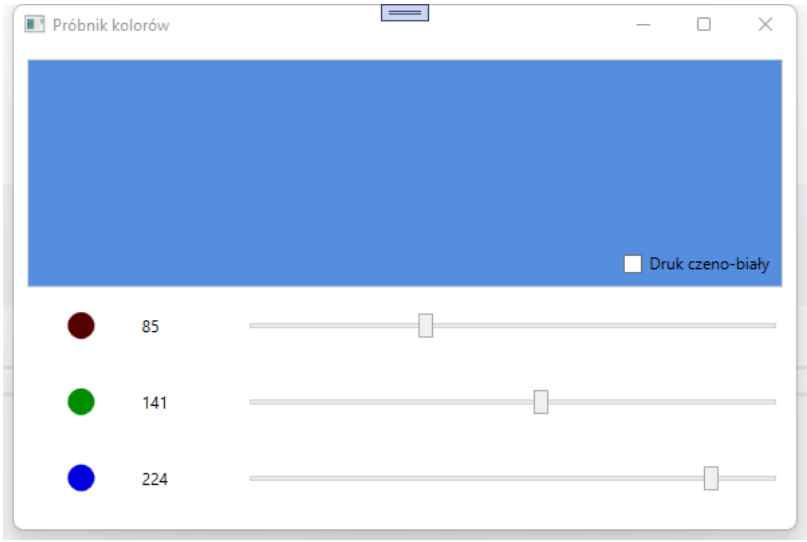


Dla spostrzegawczych:
Tak, macie rację. Coś tu się nie zgadza – to NIE jest model RGB. Przezroczystość to nie to samo co addytywne mieszanie kolorów (sumowanie światła).



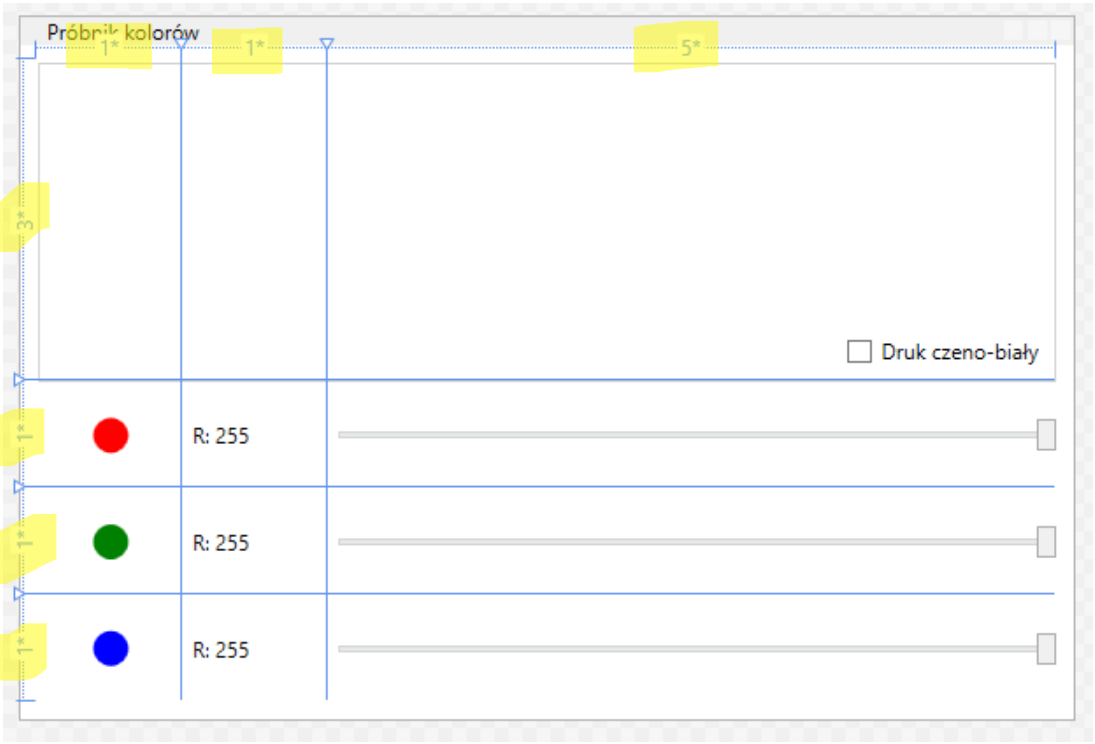
Przykład: Próbnik kolorów

Jeszcze raz przykład z próbnikiem kolorów RGB - tym razem dla odmiany wykonany w technologii WPF



Przykład: Próbnik kolorów

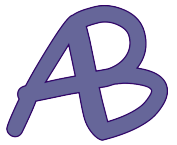
Próbnik wykonany będzie na siatce (panel Grid) siatka posiadać będzie 4 wiersze i 3 kolumny.



Szerokość kolumn i wysokość wierszy najprościej dostosować przy użyciu systemu gwiazdek.

Pierwszemu wierszowi nadano wagę 3 gwiazdek, a ostatniej kolumnie 5-ciu gwiazdek. Wszystkie pozostałe wiersze i kolumny mają po jednej gwiazdce.

Przykład: Próbnik kolorów

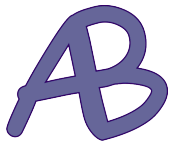


Maksymalne i minimalne wymiary okna.

```
Title="Próbnik kolorów" MinHeight="300" MaxHeight="500" Height="400"
                                MinWidth="400" MaxWidth="800" Width="600"
                                              Activated="cre_Activated">
    <Grid Margin="10px">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="*"/>
            <ColumnDefinition Width="5*"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="3*"/>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
        </Grid.RowDefinitions>
    </Grid>
```

Kod definiujący siatkę

Przykład: Próbnik kolorów



Prostokąt znajduje się domyślnie w polu (0,0) lecz zajmuje wszystkie trzy pola pierwszego wiersza (polecenie `Grid.ColumnSpan="3"`)

Kontrolka prostokąt na której wyświetlany będzie wynikowy kolor.

```
<Rectangle x:Name="probnik"
           Stroke="LightGray„ Grid.ColumnSpan="3" />

<CheckBox x:Name="druk" Content="Druk czeno-biały„
      Grid.Column="2" HorizontalAlignment="Right"
      VerticalAlignment="Bottom"
      Margin="10px"
      Click="druk_Click"/>
```

Kontrolka CheckBox, która przełączać będzie w tryb podglądu wydruku czarno-białego.
Osadzona w zerowym wierszu (domyślnie) i kolumnie o indeksie 2

Przykład: Próbnik kolorów

```
<Ellipse x:Name="p1"
    Grid.Column="0" Grid.Row="1"
    Width="20px" Height="20px"
    Fill="Red"/>

<Ellipse x:Name="p2"...>

<Ellipse x:Name="p3"...>
```

Znaczniki które prezentowały będą natężenia poszczególnych składowych wykonano za pomocą elips.

Kontrolki Label na których wyświetlane będą wartości poszczególnych składowych

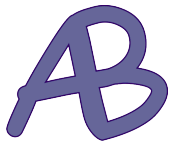
```
<Label x:Name="labelR" Content="R: 255"
       Grid.Column="1" Grid.Row="1"
       VerticalAlignment="Center"/>

<Label x:Name="labelG" Content="R: 255"...>

<Label x:Name="labelB" Content="R: 255"...>
```

Przykład: Próbnik kolorów

```
<Slider x:Name="suwakR"
        Grid.Column="2" Grid.Row="1"
        VerticalAlignment="Center"
        Maximum="255" Minimum="0"
        Value="255" ValueChanged="suwakR_ValueChanged"
        />

<Slider x:Name="suwakG".../>

<Slider x:Name="suwakB".../>
```

Poszczególne składowe kontrolowane będą za pomocą suwaków do których przypisano zdarzenie `ValueChanged="suwakR_ValueChanged"`

Przykład: Próbnik kolorów



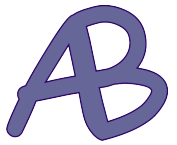
Metoda obsługi zdarzenia zmiany wartości suwaka.
Wywołana za pomocą: `ValueChanged="suwakR_ValueChanged"`

```
private void suwakR_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)
{
    p1.Fill = new SolidColorBrush(Color.FromRgb((byte)e.NewValue, 0,0));
    labelR.Content = Math.Round(e.NewValue).ToString();
    if (suwakR != null && suwakG != null && suwakB != null)
        zmien_kolor(); // wywołanie metody kolorującej
}
```

W WPF metoda zmiany wartości suwaka wywoływana jest także w momencie jego tworzenia.

Jeżeli odnosimy się w niej do elementów które mogą jeszcze nie być utworzone (są zdefiniowane niżej w kodzie XAML) to musimy upewnić się, że elementy takie istnieją (ich referencje są różne od NULL) inaczej wywołamy błąd.

Przykład: Próbnik kolorów



Metoda kolorująca prostokąt.

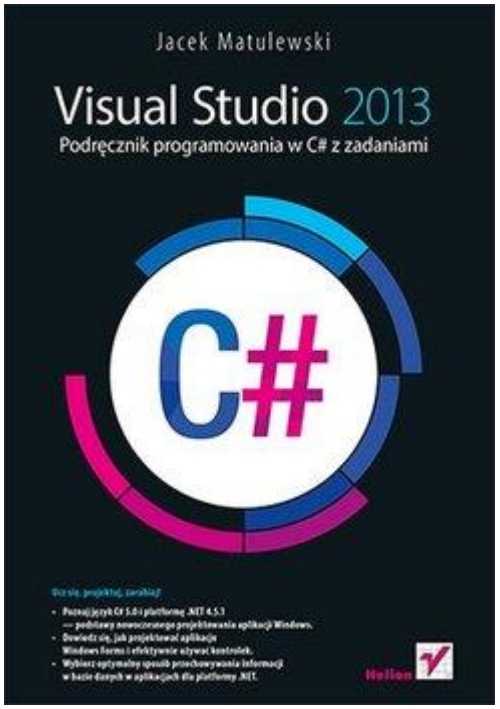
```
private void zmien_kolor()
{
    byte sredniaJasnosc = (byte)((suwakR.Value + suwakG.Value + suwakB.Value) / 3);
    if (sredniaJasnosc < 100)
        druk.Foreground = new SolidColorBrush(Colors.White);
    else
        druk.Foreground = new SolidColorBrush(Colors.Black);

    if (!(bool)druk.IsChecked)
        probnik.Fill = new SolidColorBrush(Color.FromRgb((byte)suwakR.Value,
                                                         (byte)suwakG.Value,
                                                         (byte)suwakB.Value));
    else
        probnik.Fill = new SolidColorBrush(Color.FromRgb(sredniaJasnosc,
                                                         sredniaJasnosc,
                                                         sredniaJasnosc));
}
```

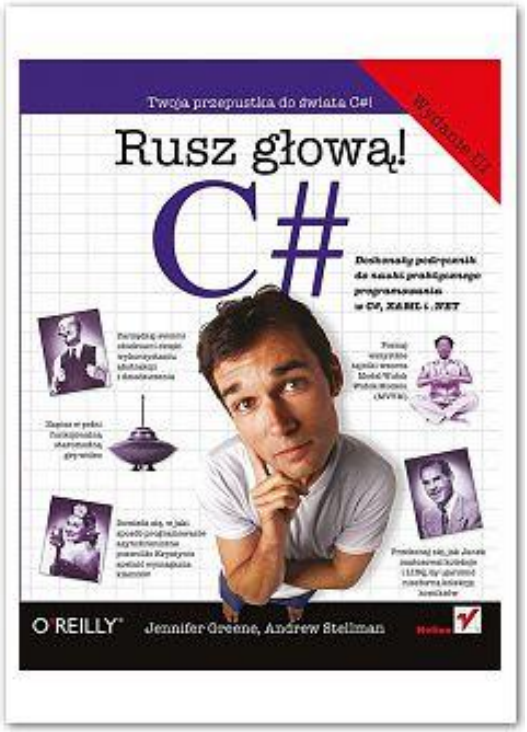
Przykład: Próbnik kolorów

Nie można zapomnieć o wywołaniu metody kolorującej prostokąt po kliknięciu na CheckBox.

```
private void druk_Click(object sender, RoutedEventArgs e)
{
    zmien_kolor();
}
```

Literatura:





Użyte w tej prezentacji tabelki pochodzą z książki:Visual Studio 2013. Podręcznik programowania w C# z zadaniami
Autor: Matulewski Jacek, Helion