TOOL DEVELOPMENT

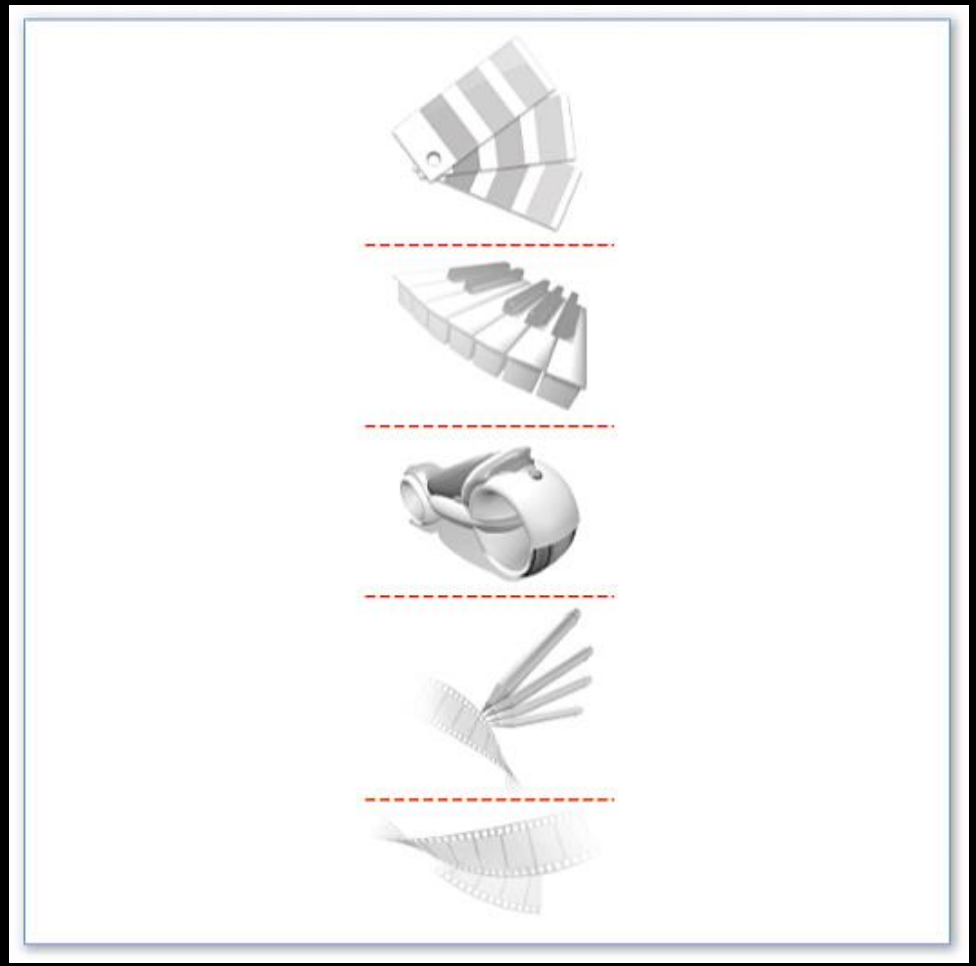
WPF – LAYOUT CONTAINERS

WHY LAYOUT CONTAINERS?

- Responsive GUI's

WPF LAYOUT CONTAINERS: OVERVIEW

ALL TYPES OF LAYOUT CONTAINERS + SPECIFICATIONS

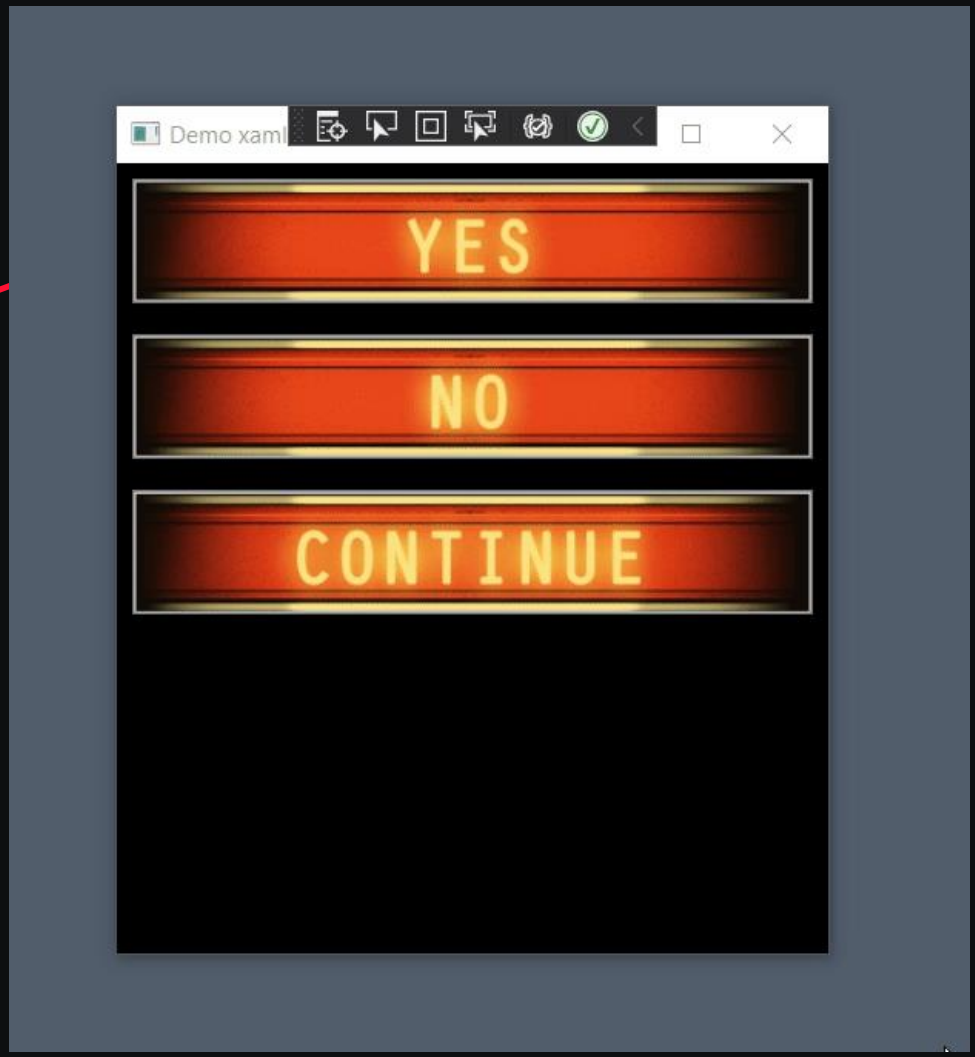


STACKPANEL

STACK ELEMENTS HORIZONTALLY OR VERTICALLY

layout containers
STACKPANEL

```
<StackPanel Background="Black">
    <Button x:Name="btnYes" Margin="8" >
        <Image Source="Resources/btnyes.png"/>
    </Button>
    <Button x:Name="btnNo" Margin="8" >
        <Image Source="Resources/btnno.png"/>
    </Button>
    <Button x:Name="btnContinue" Margin="8" >
        <Image Source="Resources/btncontinue.png"/>
    </Button>
</StackPanel>
```

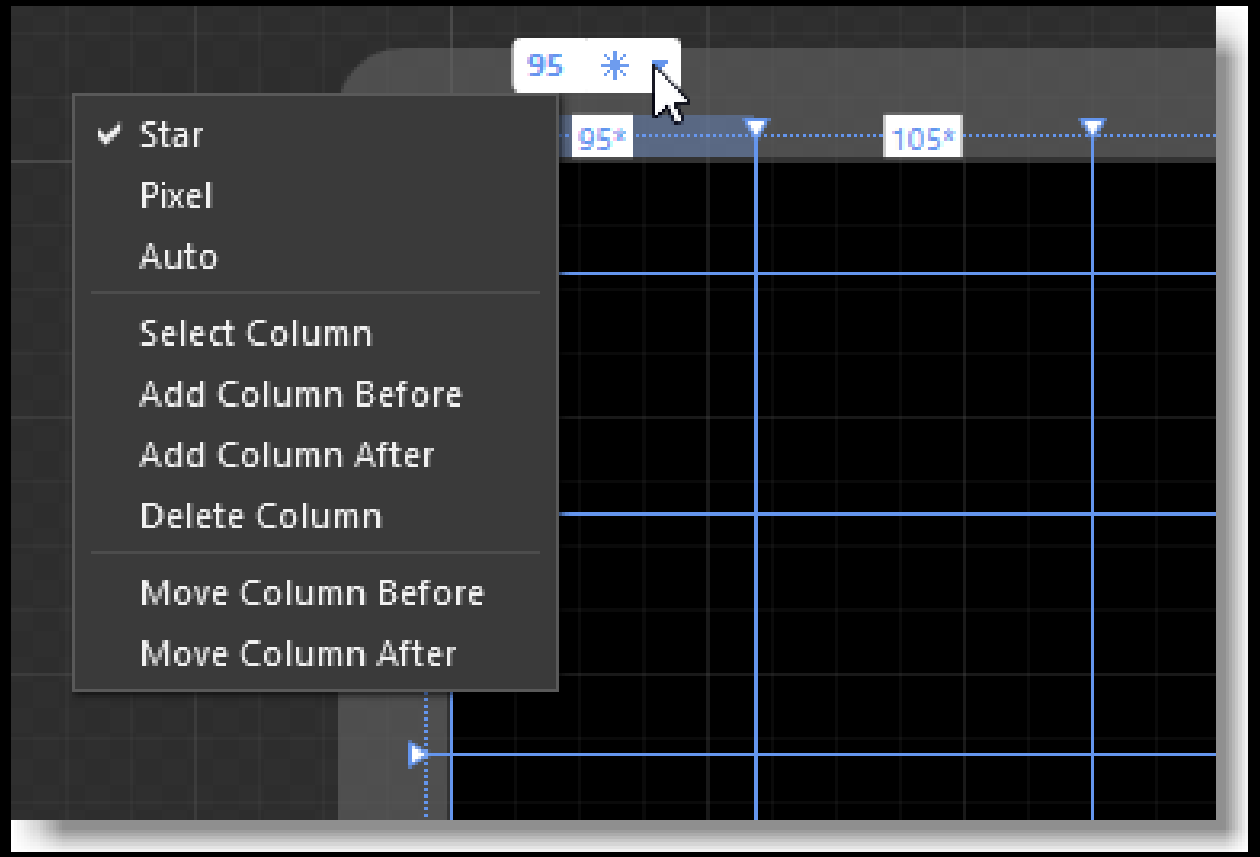


- order of controls in xaml = important!
- places every control right after (here: under) each other
- any space left (here: at the bottom) remains open
- no more space at the bottom? → out of window

STACKPANEL – LEFT TO RIGHT

```
<StackPanel Orientation="Horizontal" Background="Black">
    <Image Source="Resources/twitter.png"  Height="100" Width="100" Margin="8"/>
    <Image Source="Resources/facebook.png" Height="100" Width="100" Margin="8"/>
    <Image Source="Resources/youtube.png" Height="100" Width="100" Margin="8"/>
    <Image Source="Resources/winner.png"  Height="100" Width="100" Margin="8"/>

</StackPanel>
```

- left to right
- remaining space on the right



GRID

ARRANGE INTO ROWS AND COLUMS

layout containers

GRID LAYOUT

```
<Grid Background="Black">
    <Grid.RowDefinitions>
        <RowDefinition Height="2*"/>
        <RowDefinition Height="1*"/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition />
        <ColumnDefinition />
        <ColumnDefinition />
    </Grid.ColumnDefinitions>
    <Image Grid.Row="1" Source="Resources/circle1.png" Margin
    <Image Grid.Row="1" Grid.Column="1" Source="Resources/circle2.png" Margin="8" />
    <Image Grid.Row="1" Grid.Column="2" Source="Resources/circle3.png" Margin="8" />

    <Image Grid.ColumnSpan="3" Source="Resources/circle large.png"  Margin="8"/>
</Grid>
```

GRID LAYOUT – ROWS & COLS

```
<Grid Background="Black">
    <Grid.RowDefinitions>
        <RowDefinition Height="2*"/>
        <RowDefinition Height="1*"/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition />
        <ColumnDefinition />
        <ColumnDefinition />
    </Grid.ColumnDefinitions>
    <Image Grid.Row="1" Source="Resources/circle1.png" Margin
    <Image Grid.Row="1" Grid.Column="1" Source="Resources/circle2.png" Margin="8" />
    <Image Grid.Row="1" Grid.Column="2" Source="Resources/circle3.png" Margin="8" />

    <Image Grid.ColumnSpan="3" Source="Resources/circle large.png"  Margin="8"/>
</Grid>
```

sum: 2 + 1 = 3*
→ first row: 2/3
→ second row: 1/3

default width = 1*
→ sum: 1+1+1 = 3*
→ each col: 1/3

GRID LAYOUT – ATTACHED PROPERTIES

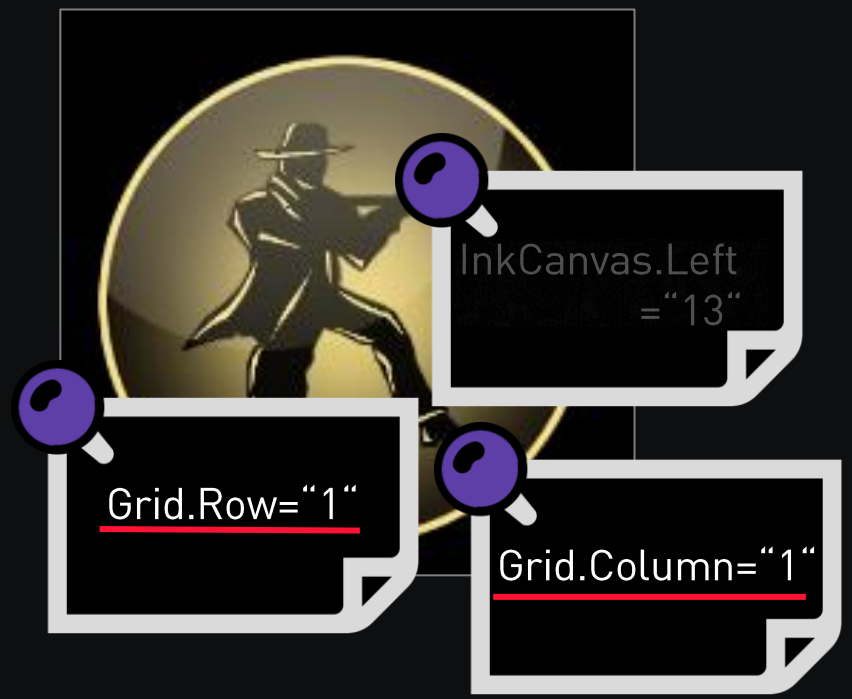
```
<Grid Background="Black">
    <Grid.RowDefinitions>
        <RowDefinition Height="2*"/>
        <RowDefinition Height="1*"/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition />
        <ColumnDefinition />
        <ColumnDefinition />
    </Grid.ColumnDefinitions>
    <Image Grid.Row="1" Source="Resources/circle1.png" Margin
    <Image Grid.Row="1" Grid.Column="1" Source="Resources/circle2.png" Margin="8" />
    <Image Grid.Row="1" Grid.Column="2" Source="Resources/circle3.png" Margin="8" />

    <Image Grid.ColumnSpan="3" Source="Resources/circle large.png"  Margin="8"/>
</Grid>
```

WHAT IS AN ATTACHED PROPERTY?

An attached property:

- Is defined in a certain class,
- but is being set on objects of another type

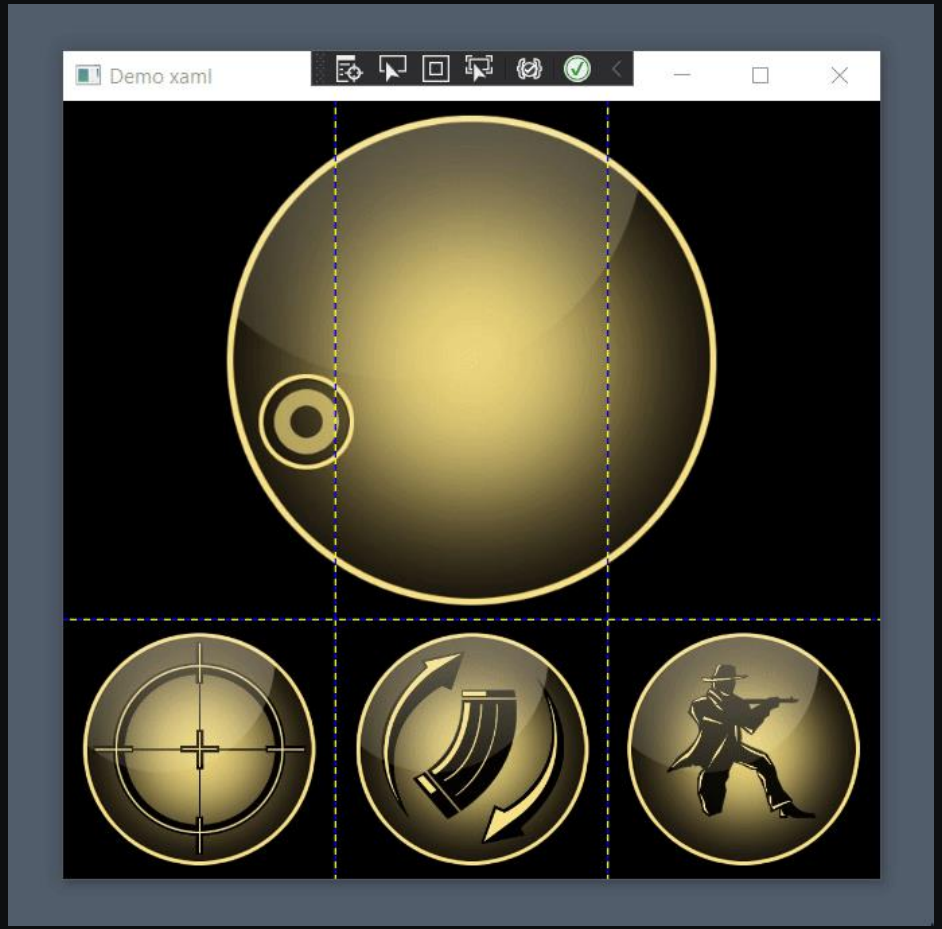
```
<Grid Background="Black">
    <Grid.RowDefinitions>
        <RowDefinition Height="2*"/>
        <RowDefinition Height="1*"/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition />
        <ColumnDefinition />
        <ColumnDefinition />
    </Grid.ColumnDefinitions>
    <Image Grid.Row="1" Source="Resource
    <Image Grid.Row="1" Grid.Column="1"
```



- Image: does not have Row / Col properties
- Grid class: has attached properties Row & Column
- These properties are attached to the Image element
- IF this image is in a grid container:
  - the grid will evaluate the attached properties in its children and change their position accordingly.

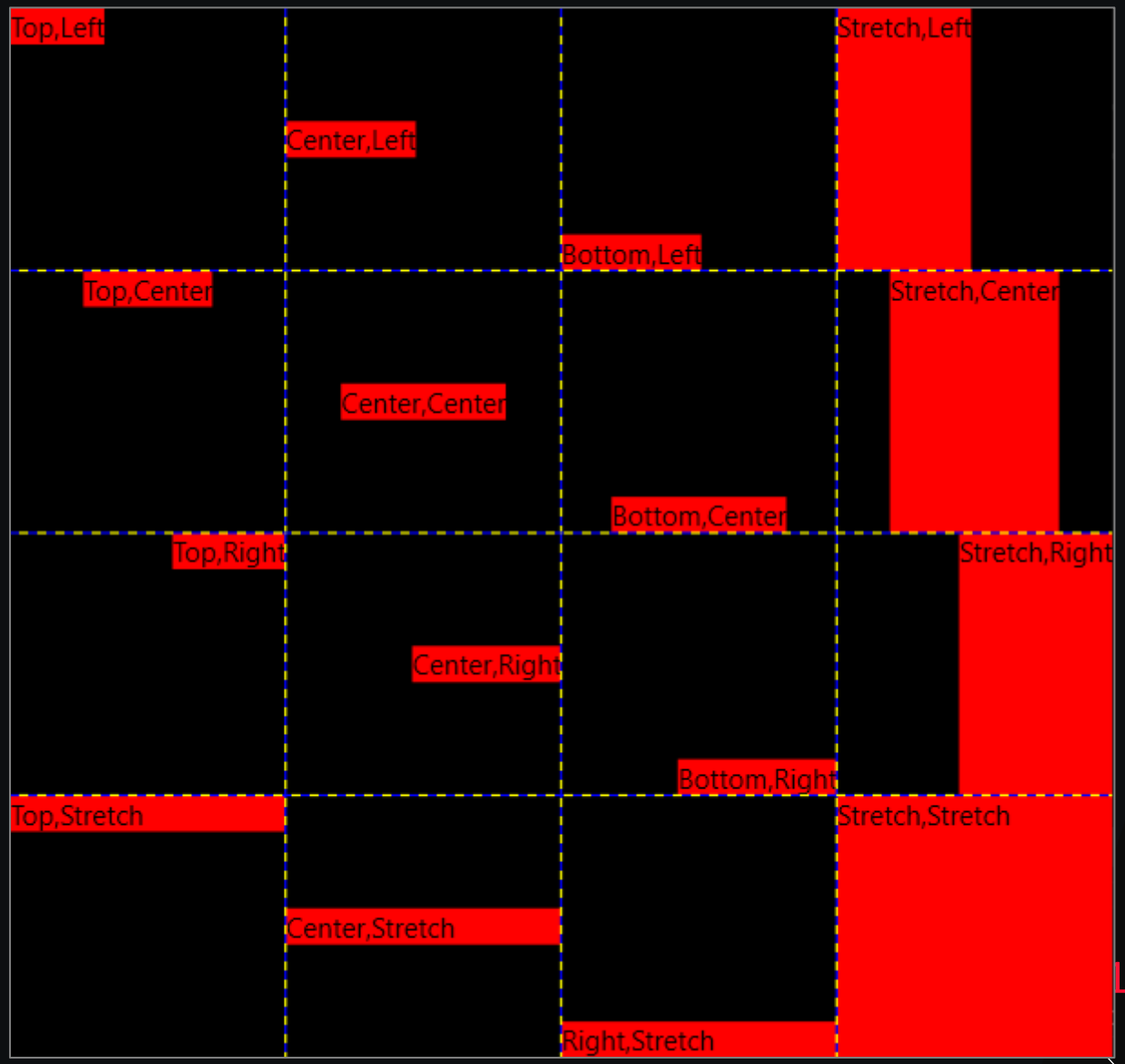
GRID LAYOUT - OVERVIEW

- order of controls in xaml is not important
  (except for overlapping elements)
- every element is placed in / stretched over a certain row / col
- when resizing the window, the layout structure remains the same
- by default elements fill the whole cell(s)



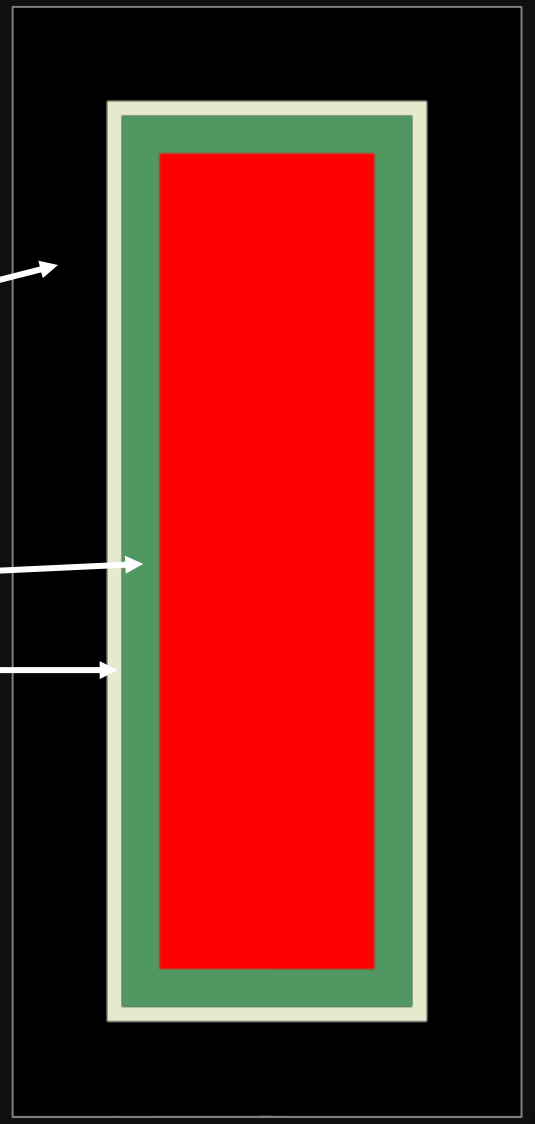
SETTING THE ALIGNMENT

- Properties VerticalAlignment and HorizontalAlignment can be set via the attributes
- This images shows an example of TextBlock controls being aligned in a cell.
- Each label shows:
  VerticalAlignment,HorizontalAlignment
- Default in grid:
  Stretch, Stretch



MARGINS AND PADDING (AND BORDER)

```
<Grid Background="Black">
    <Border Margin="50" Padding="20" BorderThickness="8"
            Background="#519761" BorderBrush="#E5EACF">
        <Rectangle Fill="Red"/>
    </Border>
</Grid>
```

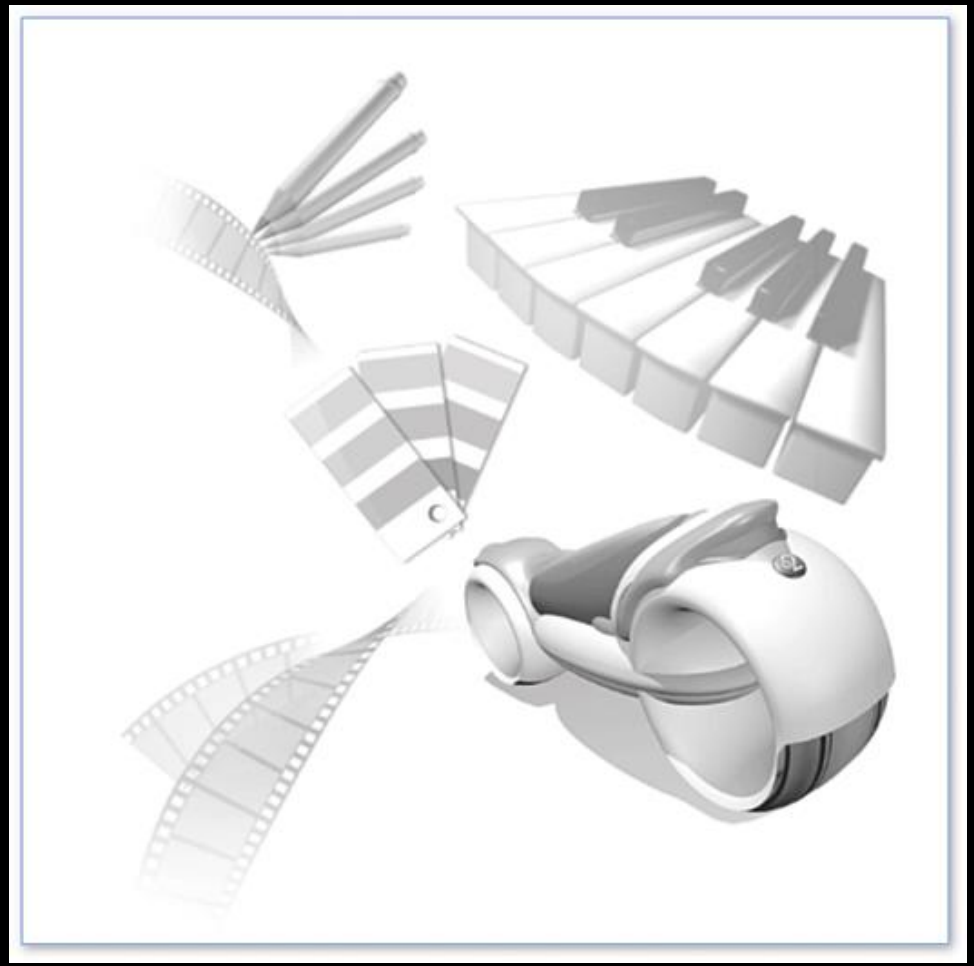


- **Margin**: space **around** the object
- **Padding**: **inner** space around **content of** the object
- **Border**: in between those two (not required)
- Thickness:
  - 1 value: left = top = right = bottom `margin="13"`
  - 2 values: left=right , top=bottom `margin="10,20"`
  - 4 values: left, top, right, bottom `margin="5,20,10,8"`

UNIFO MGRID

ARRANGE OBJ , OR UNIFORM, GRID REGIONS.

WPF ONLY



CANVAS

FIXED POSITION FOR EACH ELEMENT

WRAP PANEL

ARRANGE FRO , WRAP THE CONTENT

WPF ONLY

DOCKPANEL

DOCK ELEMEN

WPF ONLY

SUMMARY: LAYOUTS IN WPF

- Choose the correct layout container
  - Grid (most preferred)
  - StackPanel (in more specific cases)
  - Others: if the two above don't suffice
- Set the alignment within the container
  - HorizontalAlignment & VerticalAlignment
- Set a margin and padding if needed
  - optionally add a border control