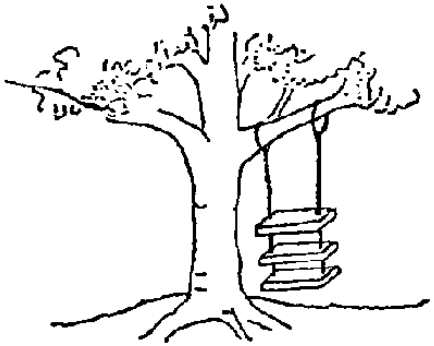
Windows Presentation Foundation (WPF)

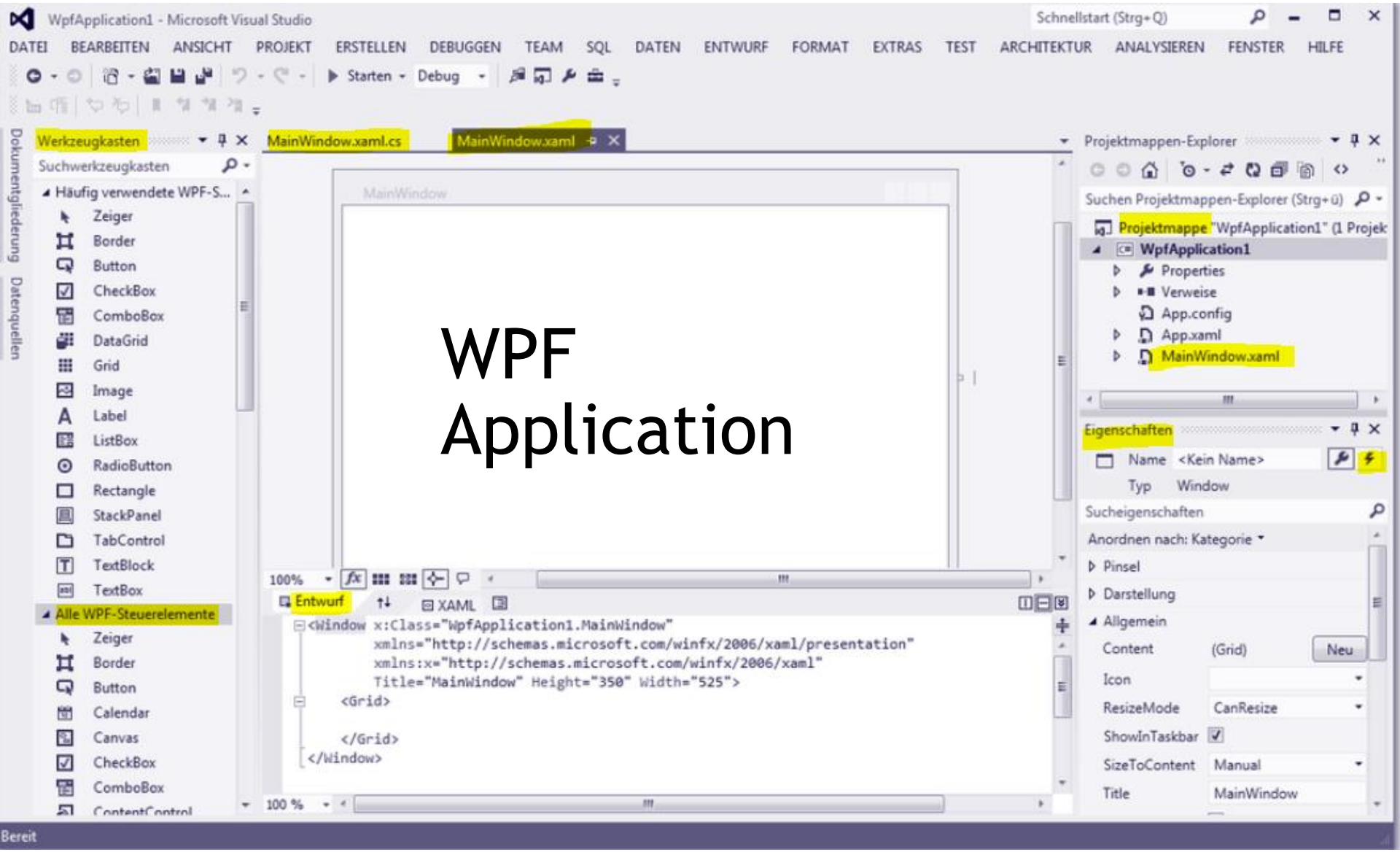
Software Entwicklung



Overview

- WPF Application
- WPF Panels

- Canvas
  - WPF Jumping Ball Example


WpfApplication1 - Microsoft Visual Studio
Schnellstart (Strg+Q)
DATEI BEARBEITEN ANSICHT PROJEKT ERSTELLEN DEBUGGEN TEAM SQL DATEN ENTWURF FORMAT EXTRAS TEST ARCHITEKTUR ANALYSIEREN FENSTER HILFE
Starten
Debug
Werkzeugkasten
Suchwerkzeugkasten
Häufig verwendete WPF-S...
Zeiger
Border
Button
CheckBox
ComboBox
DataGrid
Grid
Image
Label
ListBox
RadioButton
Rectangle
StackPanel
TabControl
TextBlock
TextBox
Alle WPF-Steuerelemente
Zeiger
Border
Button
Calendar
Canvas
CheckBox
ComboBox
MainWindow.xaml.cs
MainWindow.xaml
MainWindow
WPF
Application
100%
Entwurf
XAML
<Window x:Class="WpfApplication1.MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Title="MainWindow" Height="350" Width="525">
<Grid>
</Grid>
</Window>
100 %
Projektmappen-Explorer
Suchen Projektmappen-Explorer (Strg+ü)
Projektmappe "WpfApplication1" (1 Projek
WpfApplication1
Properties
Verweise
App.config
App.xaml
MainWindow.xaml
Eigenschaften
Name <Kein Name>
Typ Window
Sucheigenschaften
Anordnen nach: Kategorie
Pinsel
Darstellung
Allgemein
Content (Grid)
Neu
Icon
ResizeMode CanResize
ShowInTaskbar
SizeToContent Manual
Title MainWindow
Bereit

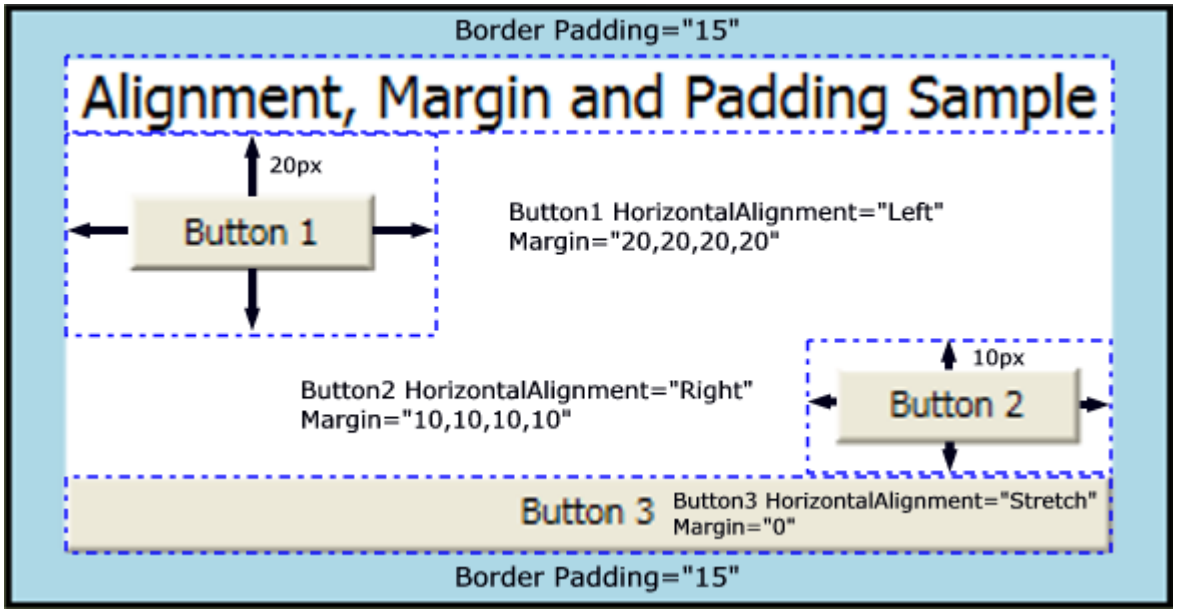
CS & XAML Files

- XAML is a declarative markup language
  - XAML simplifies creating a UI
  - create visible UI elements in the declarative XAML markup
  - separate the UI definition from the run-time logic by using code-behind files
  - joined to the markup through partial class definitions

https://docs.microsoft.com/en-us/dotnet/framework/wpf/advanced/xaml-in-wpf

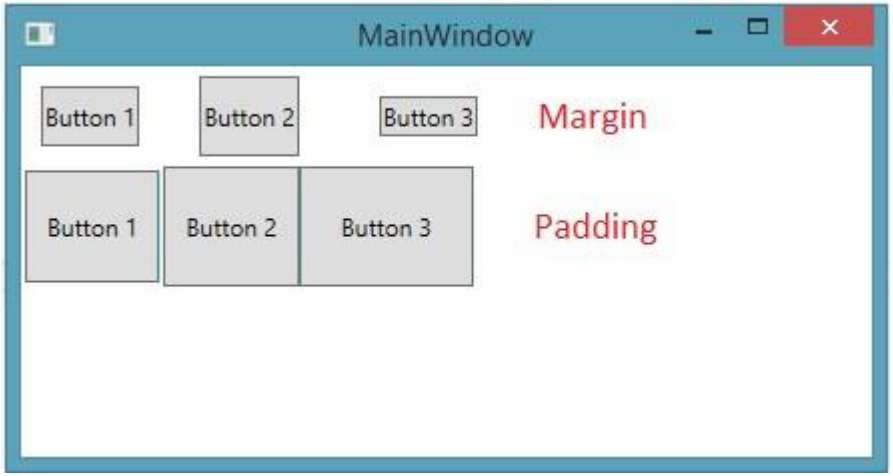
Margin & Allignment

- Margin: the distance between…
- Allignment: Left, Right, Top, Bottom

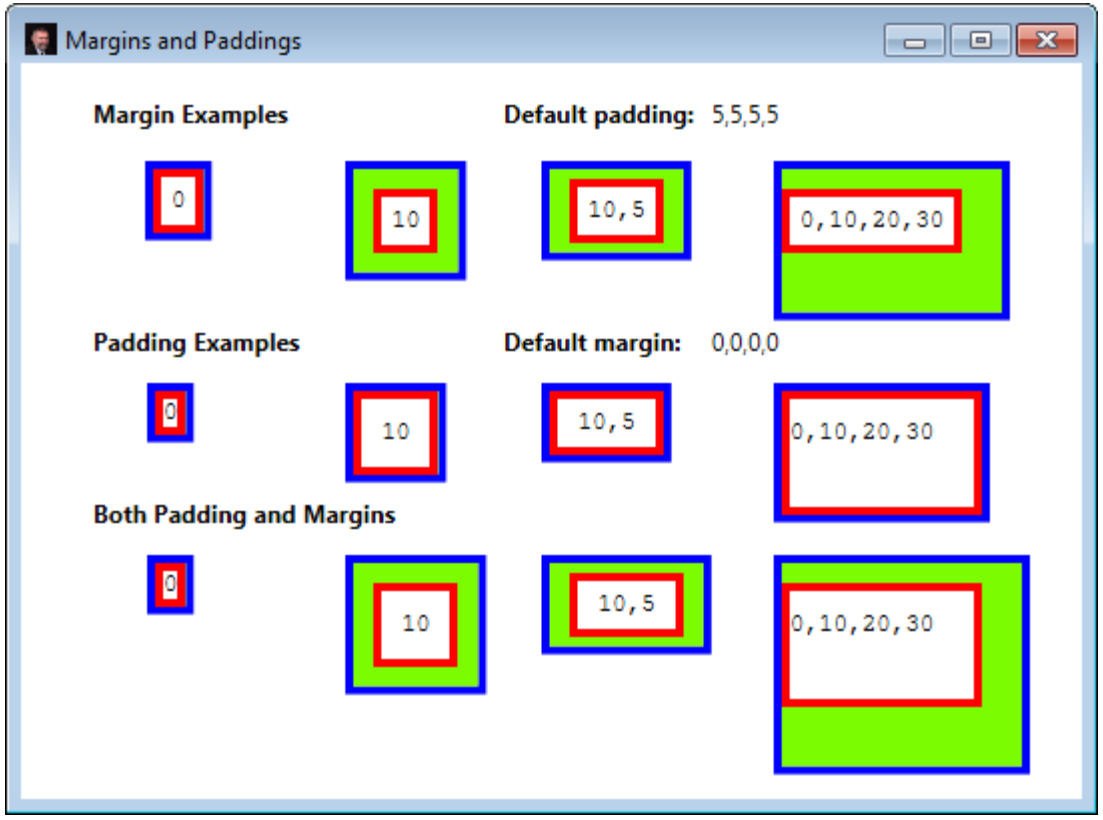


Margin vs Padding

- Margin
  - controls how much extra space gets placed around the outside edges of the element
- Padding
  - controls around the inside edges of the element



Example Padding & Margin

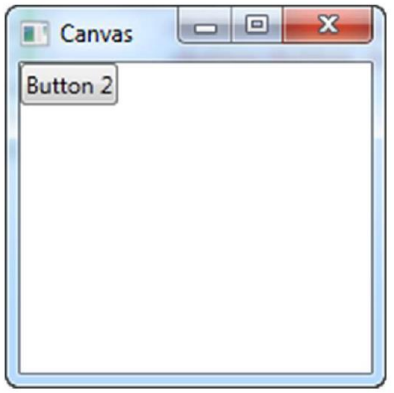


https://www.codeproject.com/Articles/321295/WPF-Controls-Tester

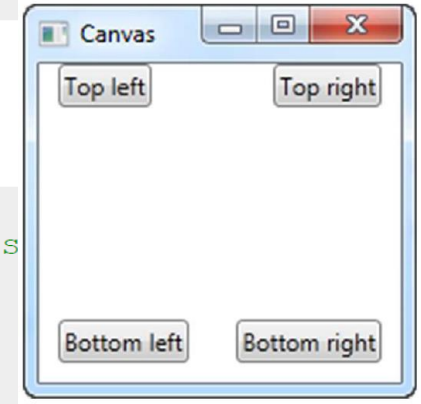
Canvas

https://wpf-tutorial.com/panels/canvas/

Canvas with Buttons

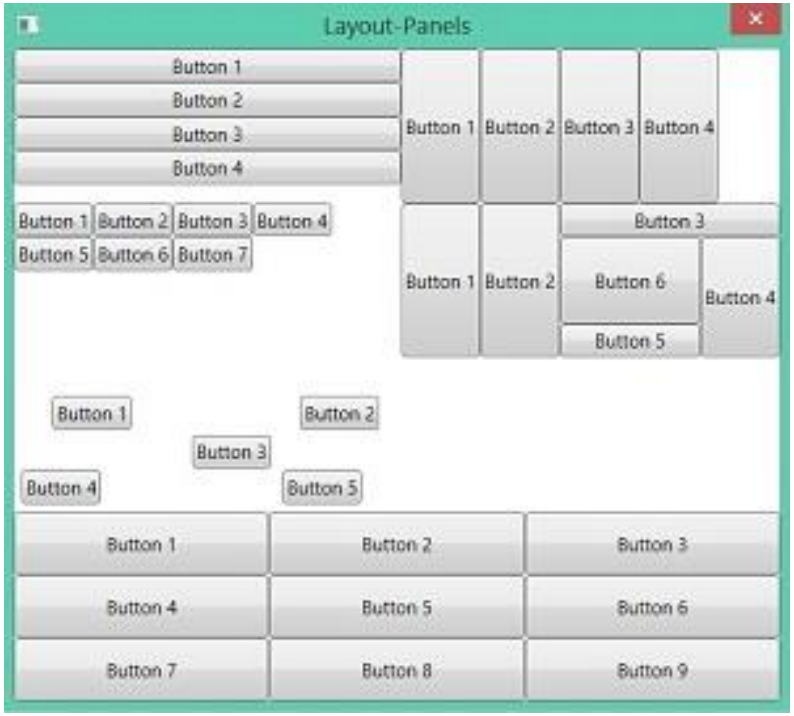



```
<Window x:Class="WpfTutorialSamples.Panels.Canvas"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xa
        Title="Canvas" Height="200" Width="200">
        <Canvas>
                <Button>Button 1</Button>
                <Button>Button 2</Button>
        </Canvas>
</Window>
```



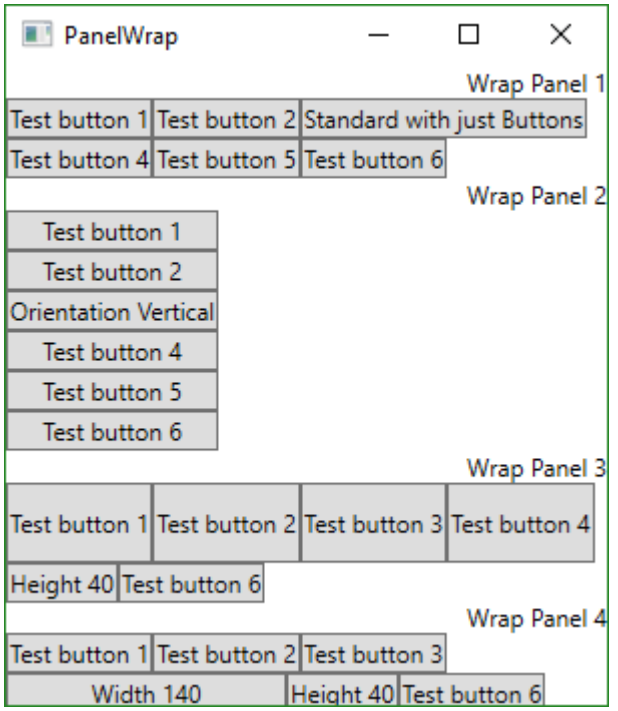

```
<Window x:Class="WpfTutorialSamples.Panels.Canvas"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/pres
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Canvas" Height="200" Width="200">
        <Canvas>
                <Button Canvas.Left="10">Top left</Button>
                <Button Canvas.Right="10">Top right</Button>
                <Button Canvas.Left="10" Canvas.Bottom="10">Bottom left</Button>
                <Button Canvas.Right="10" Canvas.Bottom="10">Bottom right</Button>
        </Canvas>
</Window>
```

The Windows Presentation Foundation (WPF) provides a number of predefined Panel elements:



Panels

Canvas	used for graphical Objects
WrapPanel	puts one element next to another
StackPanel	puts element after element (expanding)
DockPanel	docks on top, bottom, left & right
Grid	uses rows and columns

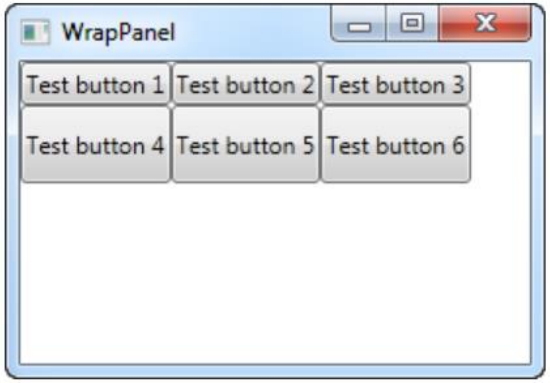


WrapPanel

positions each of its child controls next to the other horizontally (default)

https://wpf-tutorial.com/panels/wrappanel/

WrapPanel



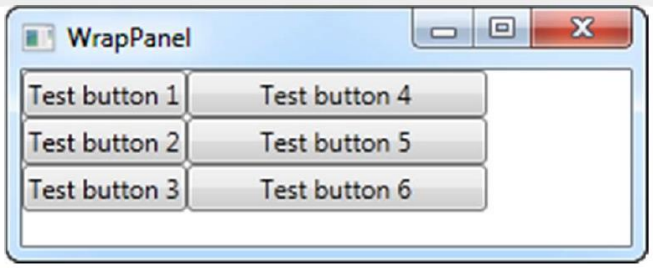
- position each of its child controls next to the other horizontally

  until there is no more room, where it will wrap to the next line and then continue

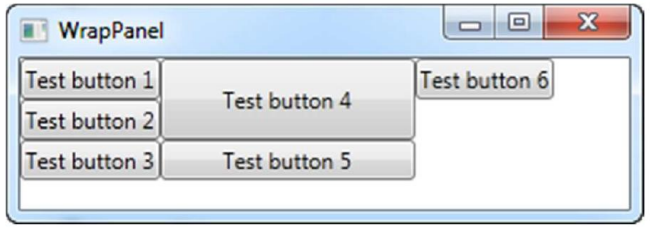
```
Title="WrapPanel" Height="300" Width="300">
<WrapPanel>
        <Button>Test button 1</Button>
        <Button>Test button 2</Button>
        <Button>Test button 3</Button>
        <Button Height="40">Test button 4</Button>
        <Button>Test button 5</Button>
        <Button>Test button 6</Button>
</WrapPanel>
```

WrapPanel

- Test the Wrap Panel
  - with different Width and Height



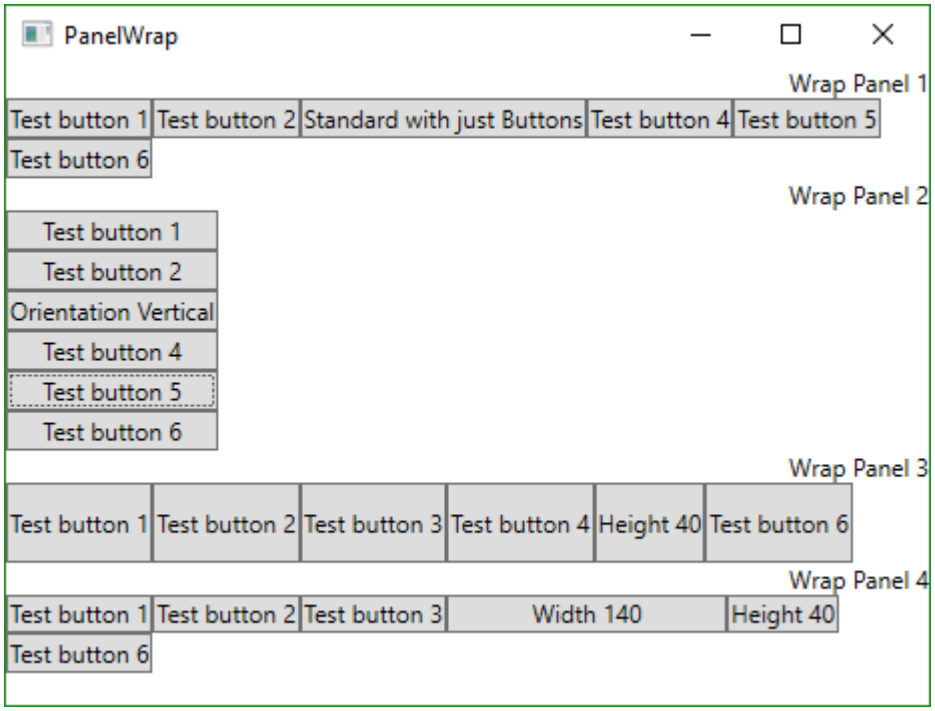

```
<Button Width="140">Test button 4</Button>
```

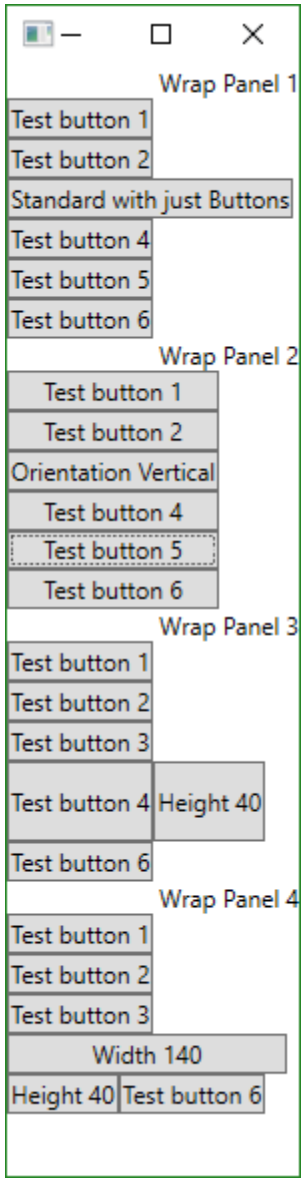


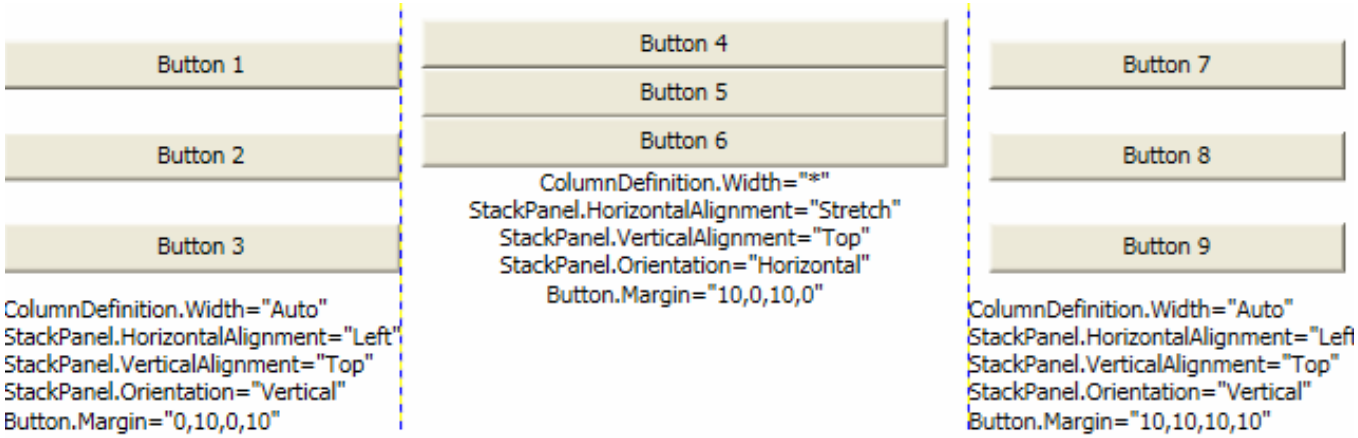

```
<Button Width="140" Height="44">Test button 4</Button>
```

Wrap Panel

- Resize your window during runtime







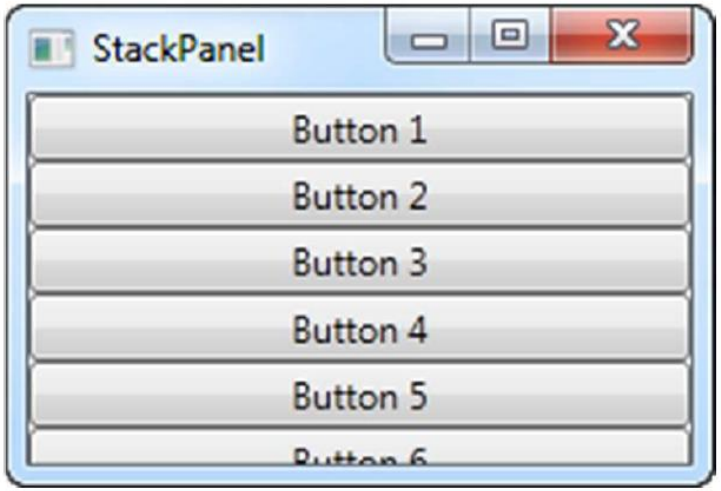
StackPanel

https://wpf-tutorial.com/panels/stackpanel/

StackPanel

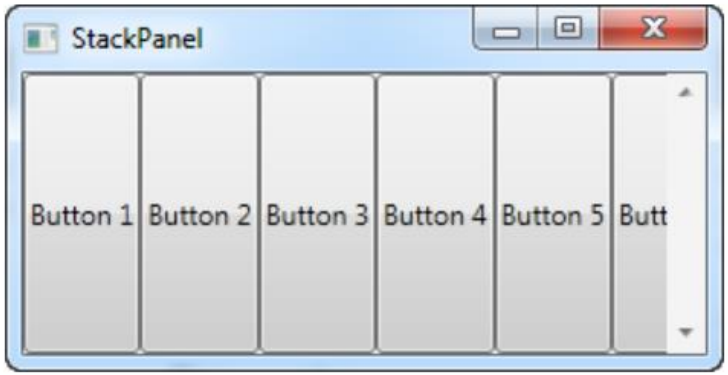
- very similar to the WrapPanel, but it doesn't wrap the content!
- it stretches it content in one direction, allowing you to stack item after item on top of each other

```
<StackPanel>
        <Button>Button 1</Button>
        <Button>Button 2</Button>
        <Button>Button 3</Button>
        <Button>Button 4</Button>
        <Button>Button 5</Button>
        <Button>Button 6</Button>
</StackPanel>
```



StackPanel

- Orientation Horizontal

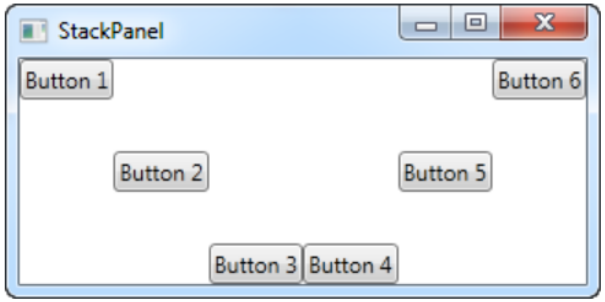



```
<StackPanel Orientation="Horizontal">
        <Button VerticalAlignment="Top">Button 1</Button>
        <Button VerticalAlignment="Center">Button 2</Button>
        <Button VerticalAlignment="Bottom">Button 3</Button>
        <Button VerticalAlignment="Bottom">Button 4</Button>
        <Button VerticalAlignment="Center">Button 5</Button>
        <Button VerticalAlignment="Top">Button 6</Button>
</StackPanel>
```

StackPanel

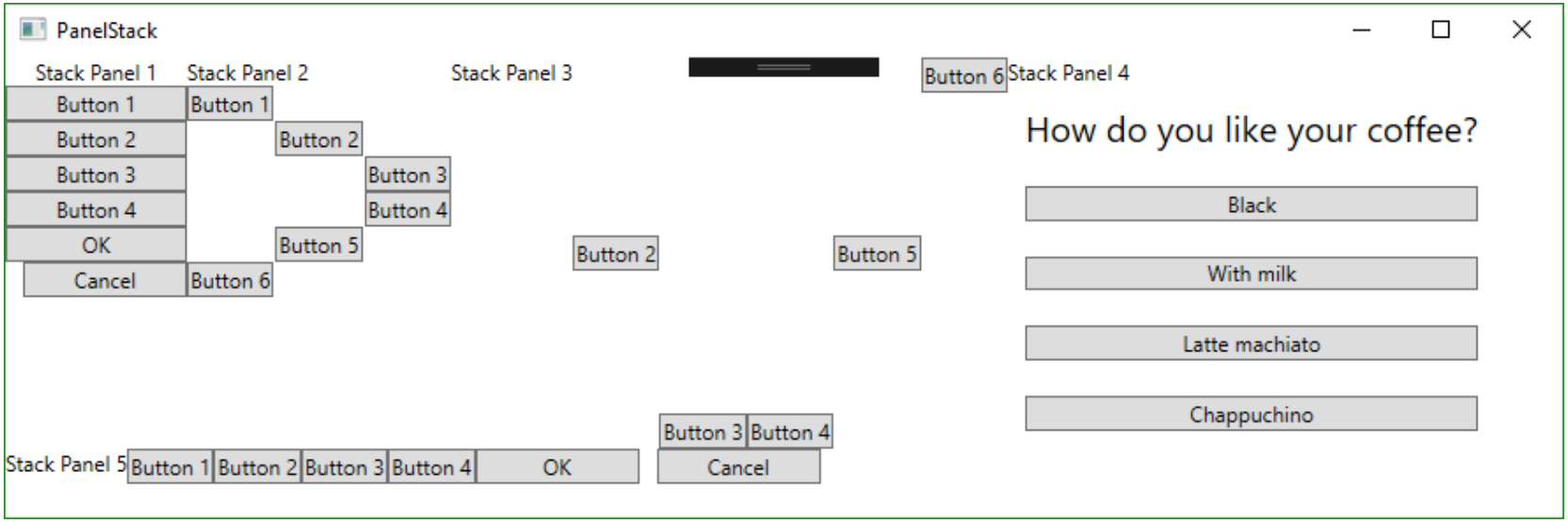
```
<StackPanel Orientation="Vertical">
        <Button HorizontalAlignment="Left">Button 1</Button>
        <Button HorizontalAlignment="Center">Button 2</Button>
        <Button HorizontalAlignment="Right">Button 3</Button>
        <Button HorizontalAlignment="Right">Button 4</Button>
        <Button HorizontalAlignment="Center">Button 5</Button>
        <Button HorizontalAlignment="Left">Button 6</Button>
</StackPanel>
```


```
<StackPanel Orientation="Horizontal">
        <Button VerticalAlignment="Top">Button 1</Button>
        <Button VerticalAlignment="Center">Button 2</Button>
        <Button VerticalAlignment="Bottom">Button 3</Button>
        <Button VerticalAlignment="Bottom">Button 4</Button>
        <Button VerticalAlignment="Center">Button 5</Button>
        <Button VerticalAlignment="Top">Button 6</Button>
</StackPanel>
```



Test Stack Panels

- 5xStackPanel in an WrapPanel:

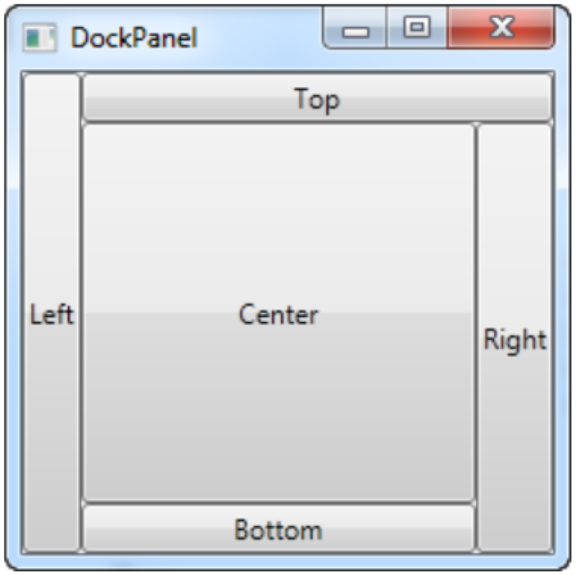


DockPanel

https://wpf-tutorial.com/panels/dockpanel/

DockPanel

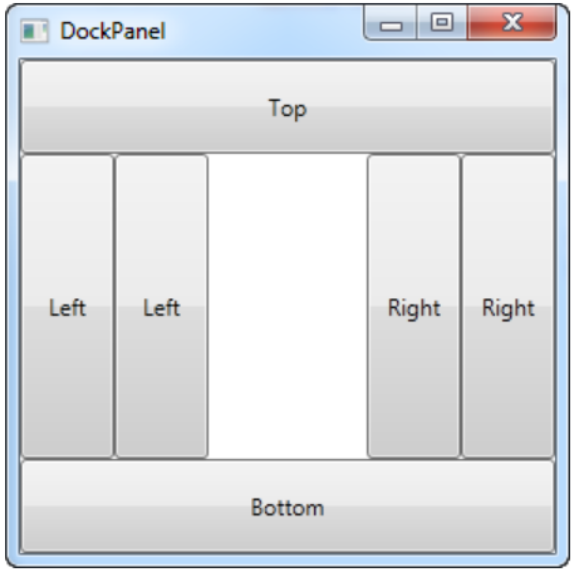
- makes it easy to dock content
  in all four directions:
  - top, bottom, left and right




```
<DockPanel>
        <Button DockPanel.Dock="Left">Left</Button>
        <Button DockPanel.Dock="Top">Top</Button>
        <Button DockPanel.Dock="Right">Right</Button>
        <Button DockPanel.Dock="Bottom">Bottom</Button>
        <Button>Center</Button>
</DockPanel>
```

DockPanel

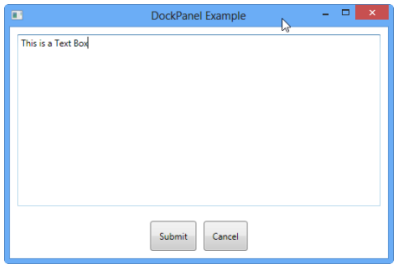
- Last Child Filled disabled

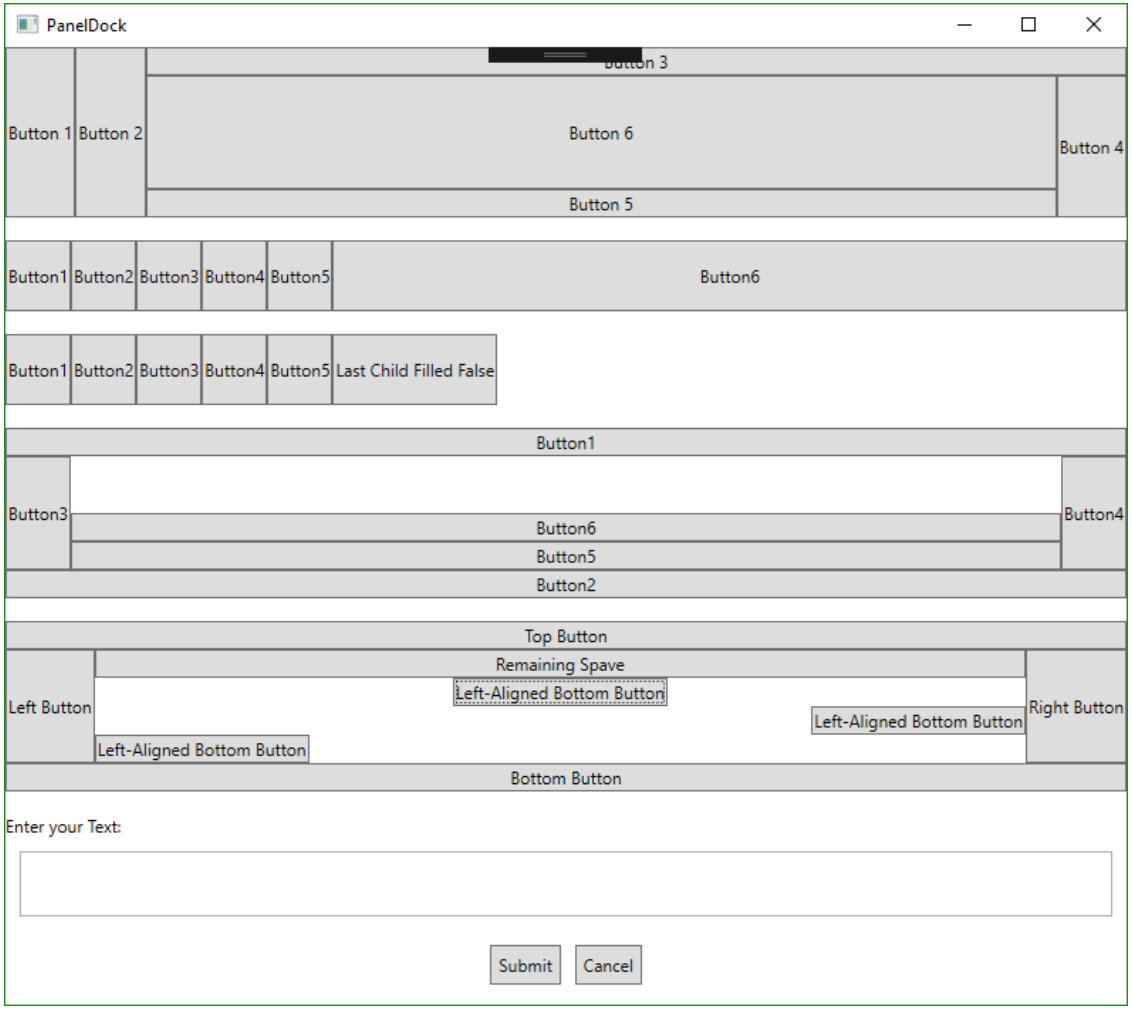



```
Title="DockPanel" Height="300" Width="300">
<DockPanel LastChildFill="False">
        <Button DockPanel.Dock="Top" Height="50">Top</Button>
        <Button DockPanel.Dock="Bottom" Height="50">Bottom</Button>
        <Button DockPanel.Dock="Left" Width="50">Left</Button>
        <Button DockPanel.Dock="Left" Width="50">Left</Button>
        <Button DockPanel.Dock="Right" Width="50">Right</Button>
        <Button DockPanel.Dock="Right" Width="50">Right</Button>
</DockPanel>
```

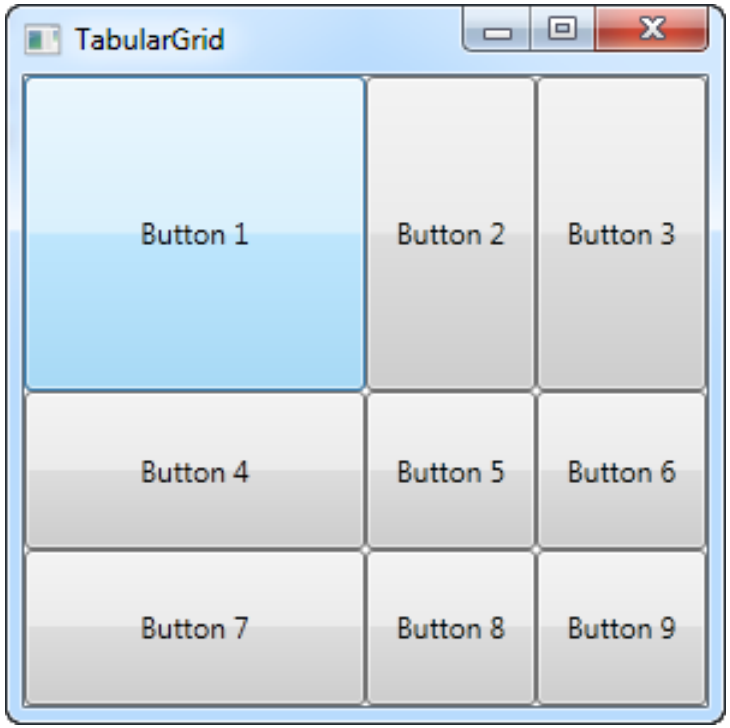
Dock Panel

- Different ways of using dock panels:





https://www.ittutorialswithexample.com/2014/08/dockpanel-in-wpf.html



Grid

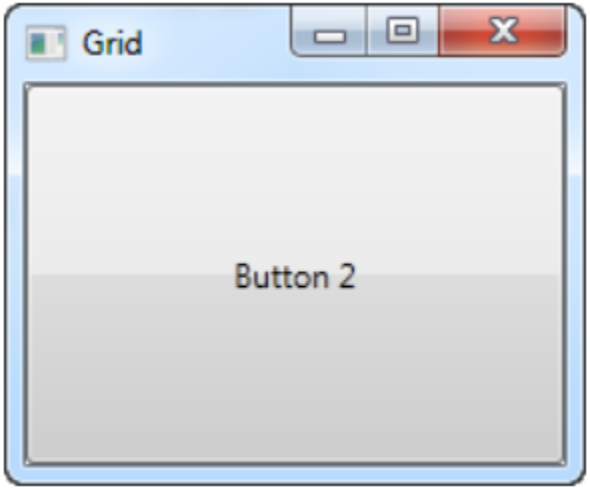
https://www.wpf-tutorial.com/panels/grid/

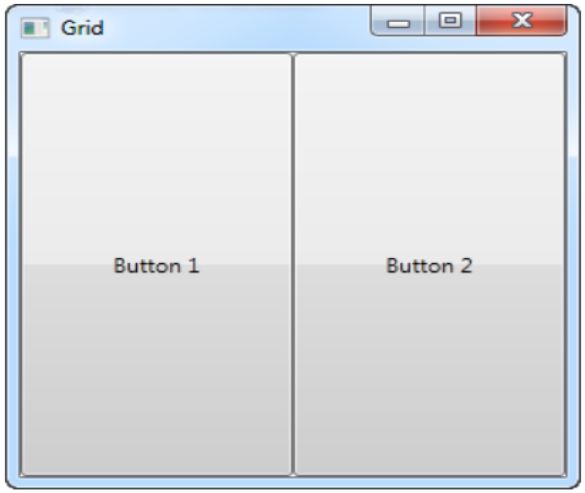
Grid

- most complex panel
- Can define
  - height for each of the rows
  - width for each of the columns
- Define the size
  - in absolute amount of pixels
  - in a percentage of the available space or
  - as auto where the row or column
    - will automatically adjust its size depending on the content

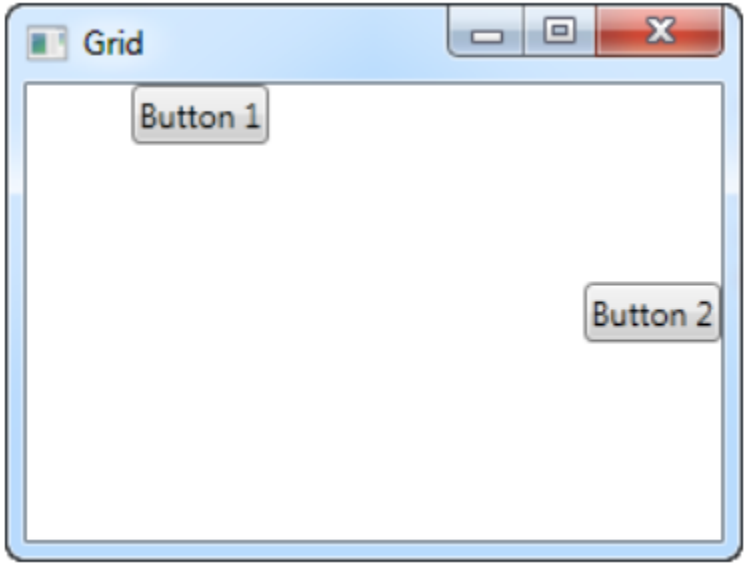
https://www.wpf-tutorial.com/panels/grid/

Try some examples



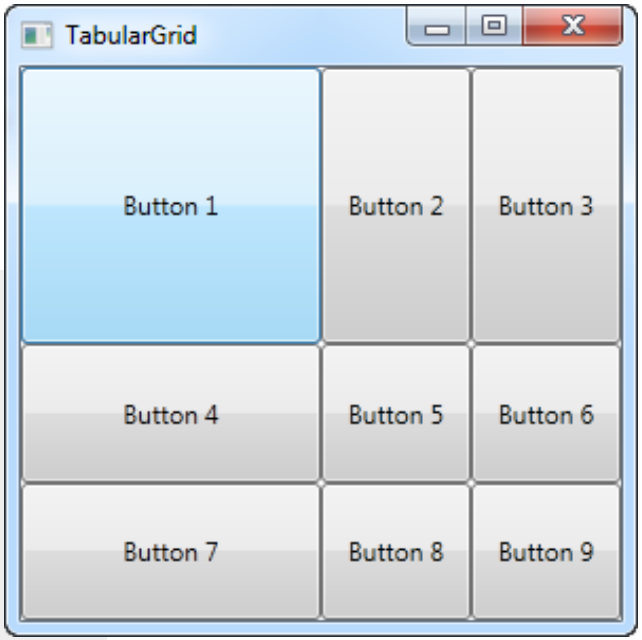


- Position your Buttons



Grid

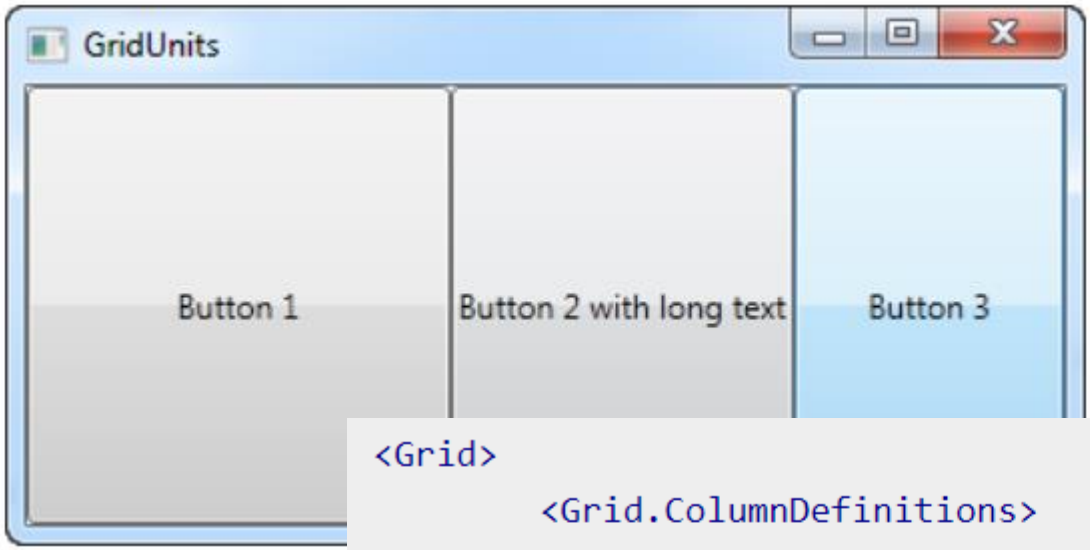
- with rows and colums

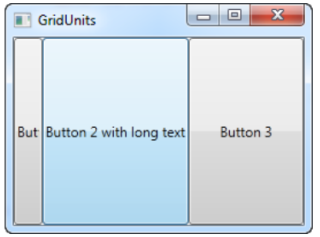


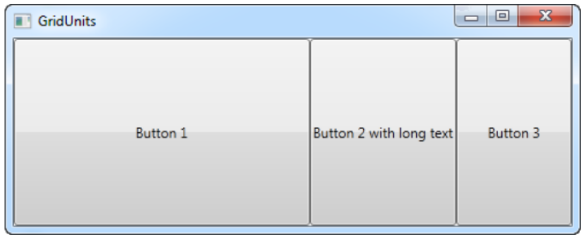

```
<Grid>
            <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="2*" />
                    <ColumnDefinition Width="1*" />
                    <ColumnDefinition Width="1*" />
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
                    <RowDefinition Height="2*" />
                    <RowDefinition Height="1*" />
                    <RowDefinition Height="1*" />
            </Grid.RowDefinitions>
            <Button>Button 1</Button>
            <Button Grid.Column="1">Button 2</Button>
            <Button Grid.Column="2">Button 3</Button>
            <Button Grid.Row="1">Button 4</Button>
            <Button Grid.Column="1" Grid.Row="1">Button 5</Button>
            <Button Grid.Column="2" Grid.Row="1">Button 6</Button>
            <Button Grid.Row="2">Button 7</Button>
            <Button Grid.Column="1" Grid.Row="2">Button 8</Button>
            <Button Grid.Column="2" Grid.Row="2">Button 9</Button>
    </Grid>
```

https://www.wpf-tutorial.com/panels/grid-rows-and-columns/

Grid: Units

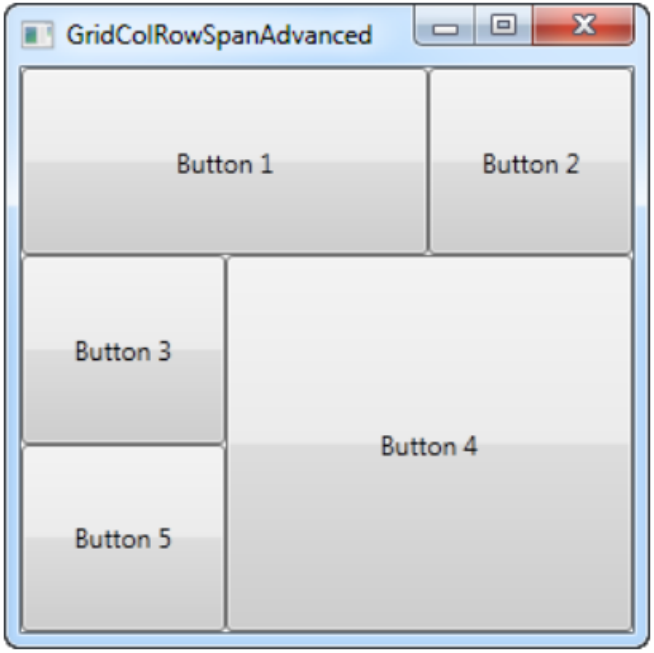







```
<Grid>
        <Grid.ColumnDefinitions>
                <ColumnDefinition Width="1*" />
                <ColumnDefinition Width="Auto" />
                <ColumnDefinition Width="100" />
        </Grid.ColumnDefinitions>
        <Button>Button 1</Button>
        <Button Grid.Column="1">Button 2 with long text</Button>
        <Button Grid.Column="2">Button 3</Button>
</Grid>
```

Grid - Spanning




```
<Grid.ColumnDefinitions>
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="*" />
        <ColumnDefinition Width="*" />
</Grid.ColumnDefinitions>
<Grid.RowDefinitions>
        <RowDefinition Height="*" />
        <RowDefinition Height="*" />
        <RowDefinition Height="*" />
</Grid.RowDefinitions>
<Button Grid.ColumnSpan="2">Button 1</Button>
<Button Grid.Column="3">Button 2</Button>
<Button Grid.Row="1">Button 3</Button>
<Button Grid.Column="1" Grid.Row="1" Grid.RowSpan="2" Grid.ColumnSpan="2">Button 4</Button>
<Button Grid.Column="0" Grid.Row="2">Button 5</Button>
```

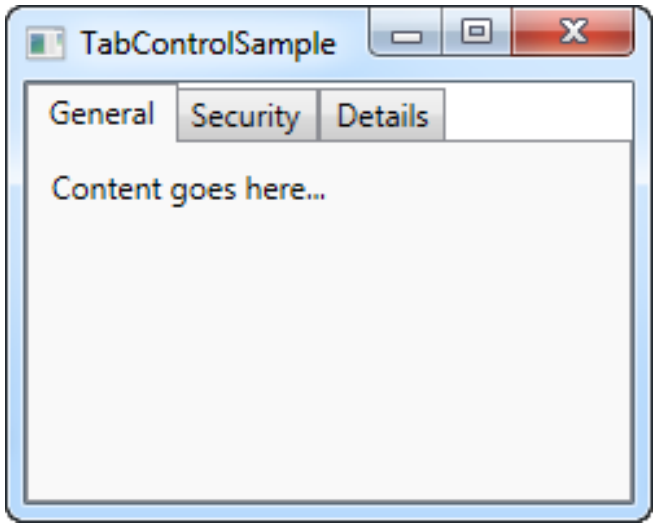
Tab Control

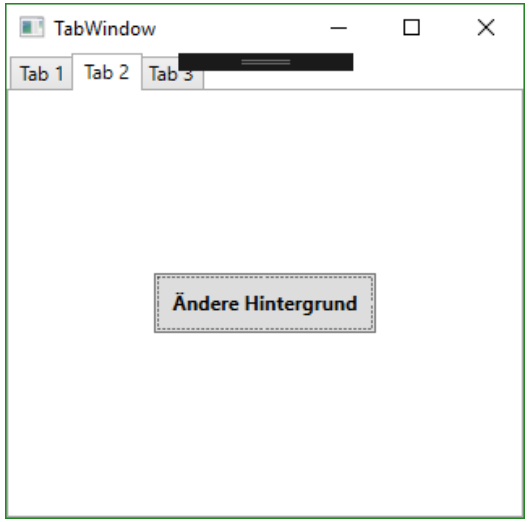
https://www.wpf-tutorial.com/tabcontrol/using-the-tabcontrol/

Tab Control

- split your interface up into different areas, each accessible by clicking on the tab header

```
<Grid>
    <TabControl>
        <TabItem Header="General">
            <Label Content="Content goes here..." />
        </TabItem>
        <TabItem Header="Security" />
        <TabItem Header="Details" />
    </TabControl>
</Grid>
```



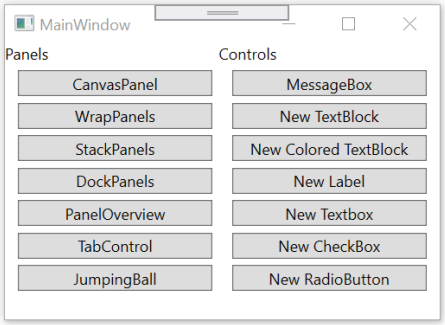


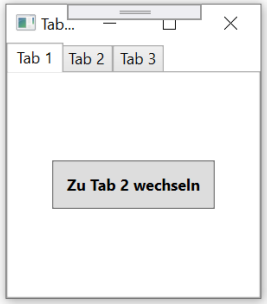
Tab Example

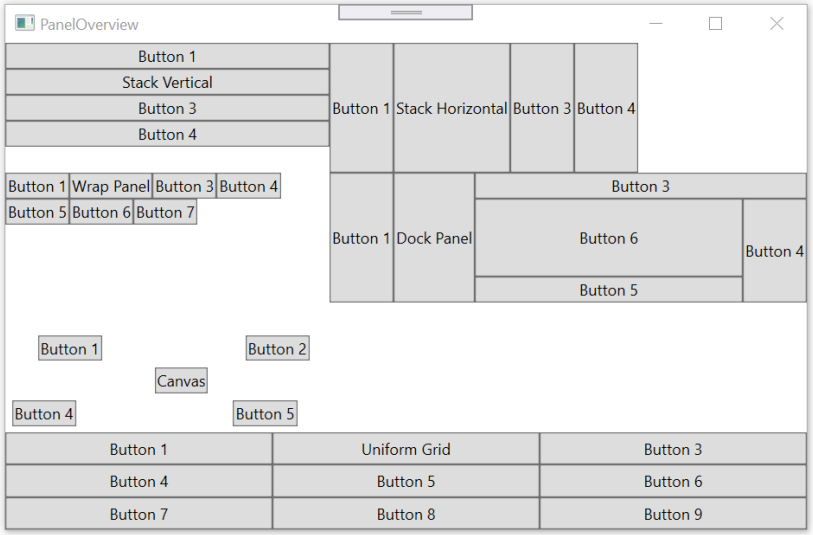
http://www.das-grosse-computer-abc.de/CSharp/index.php?chapter=12&topic=6

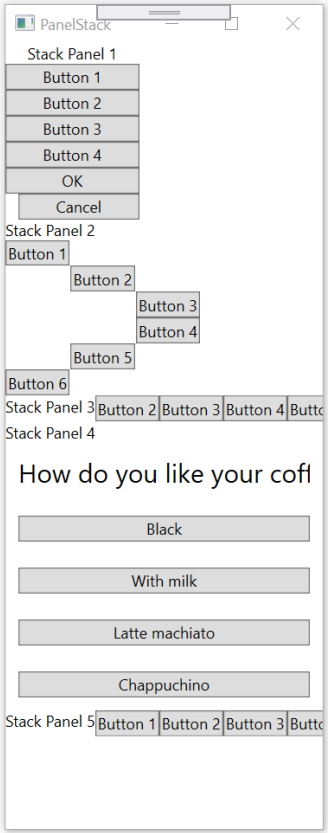
```
<TabControl Name="tabControlFenster">
    <TabItem Header="Tab 1">
        <Grid Background="White">
            <Button VerticalAlignment="Center" Hor
        </Grid>
    </TabItem>
    <TabItem Header="Tab 2">
        <Grid Background="White">
            <Button VerticalAlignment="Center" Hor
        </Grid>
    </TabItem>
    <TabItem Header="Tab 3">
        <Grid Background="White">
            <Button VerticalAlignment="Center" Hor
        </Grid>
    </TabItem>
</TabControl>
```

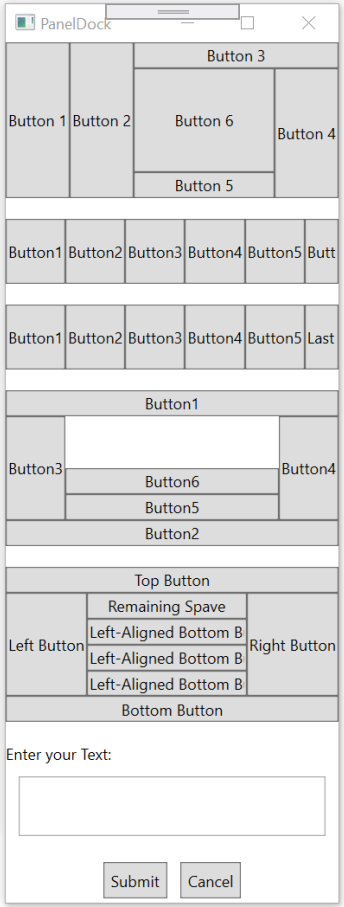
All together now… all Panels produced

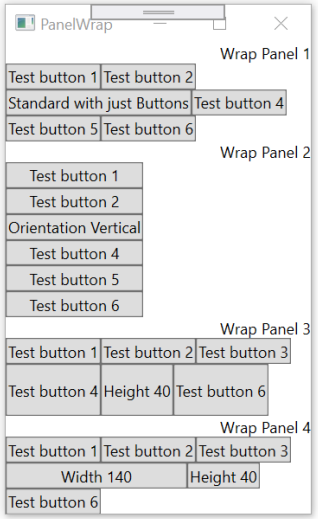


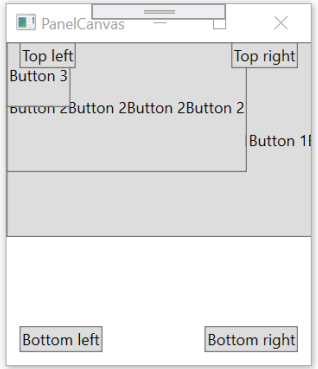








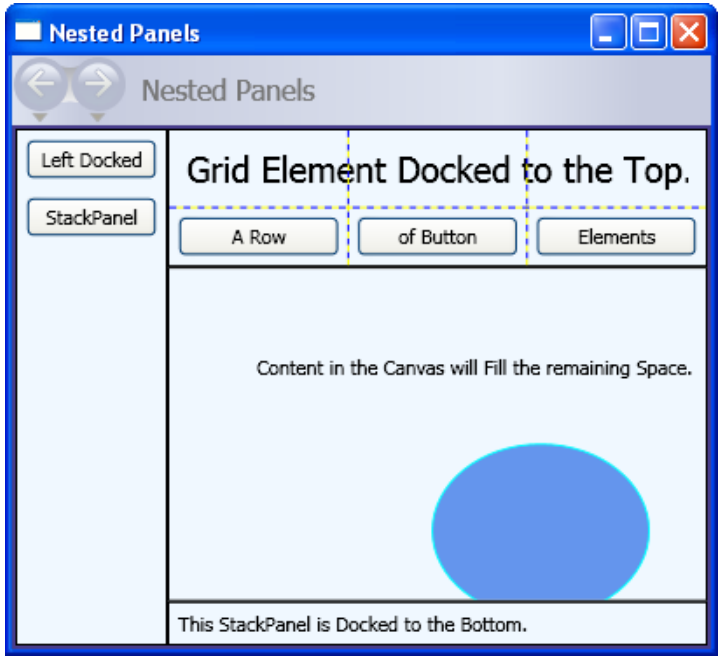




Panel Excercise

Panel Overview

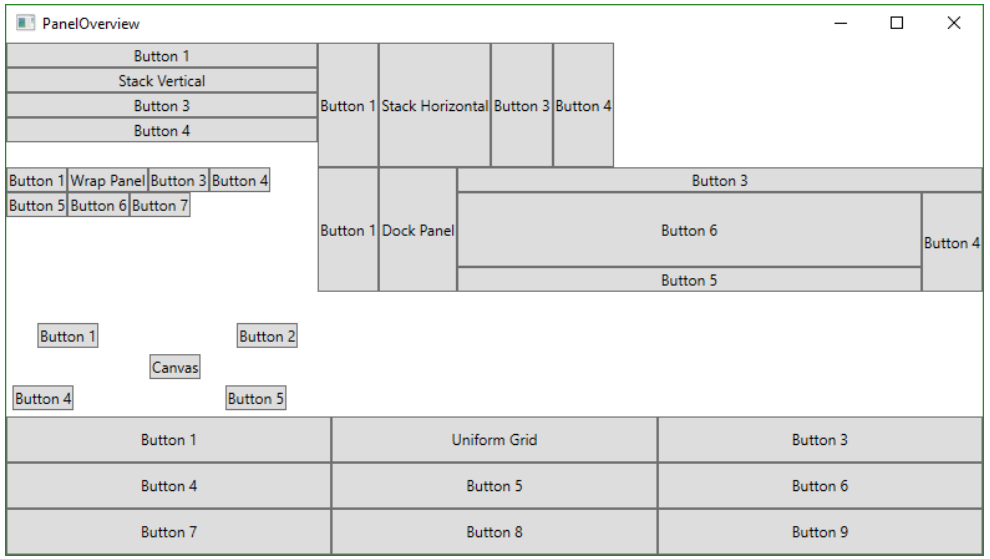
Canvas Jumping Ball



Panel & Code Behind

Code Behind – not XAML

https://docs.microsoft.com/de-de/dotnet/framework/wpf/controls/panels-overview



Example: Panel Overview

Use Grid: 3 Rows 2 Columns

- Add Stack Panel Vertical & Stack Panel Horizontal
- Wrap Panel und Dock Panel
- Canvas & Uniform Grid

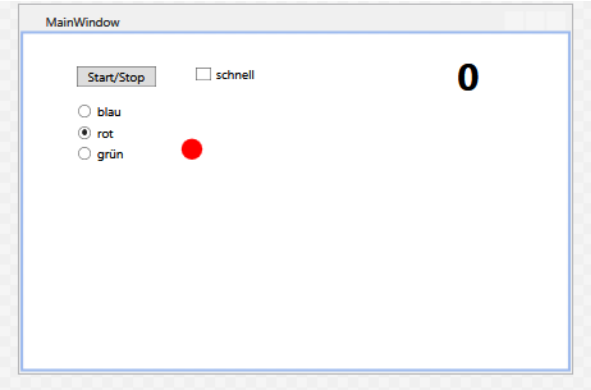
Grid Definition

```
    Title="PanelOverview" Height="450" Width="800">
<Grid> <!--3 rows 2 columns-->
    <Grid.RowDefinitions>
        <RowDefinition Height="100" />
        <RowDefinition Height="100" />
        <RowDefinition Height="100" />
        <RowDefinition />
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="250" />
        <ColumnDefinition />
    </Grid.ColumnDefinitions>
    <StackPanel Grid.Row="0" Grid.Column="0" Orientation="Vertical"...>
    <StackPanel Grid.Row="0" Grid.Column="1" Orientation="Horizontal"...>
    <WrapPanel Grid.Row="1" Grid.Column="0"...>
    <DockPanel Grid.Row="1" Grid.Column="1"...>
    <Canvas Grid.Row="2" Grid.Column="0"...>
    <UniformGrid Grid.Row="3" Grid.Column="0" Grid.ColumnSpan="2" Rows="3" Columns="3"...>
</Grid>
```

Add Panels to Grid

```
<Grid> <!--3 rows 2 columns-->
    <Grid.RowDefinitions...>
    <Grid.ColumnDefinitions...>
    <StackPanel Grid.Row="0" Grid.Column="0" Orientation="Vertical">
        <Button Content="Button 1" />
        <Button Content="Stack Vertical" />
        <Button Content="Button 3" />
        <Button Content="Button 4" />
    </StackPanel>
    <StackPanel Grid.Row="0" Grid.Column="1" Orientation="Horizontal">
        <Button Content="Button 1" />
        <Button Content="Stack Horizontal" />
        <Button Content="Button 3" />
        <Button Content="Button 4" />
    </StackPanel>
    <WrapPanel Grid.Row="1" Grid.Column="0">
        <Button Content="Button 1" />
        <Button Content="Wrap Panel" />
        <Button Content="Button 3" />
        <Button Content="Button 4" />
        <Button Content="Button 5" />
        <Button Content="Button 6" />
        <Button Content="Button 7" />
    </WrapPanel>
```

```
<Canvas Grid.Row="2" Grid.Column="0">
    <Button Canvas.Left="25" Canvas.Top="25" Content="Button 1" />
    <Button Canvas.Left="185" Canvas.Top="25" Content="Button 2" />
    <Button Canvas.Left="115" Canvas.Top="50" Content="Canvas" />
    <Button Canvas.Left="5" Canvas.Bottom="5" Content="Button 4" />
    <Button Canvas.Right="25" Canvas.Bottom="5" Content="Button 5" />
</Canvas>
<UniformGrid Grid.Row="3" Grid.Column="0"
             Grid.ColumnSpan="2" Rows="3" Columns="3">
    <Button Content="Button 1" />
    <Button Content="Uniform Grid" />
    <Button Content="Button 3" />
    <Button Content="Button 4" />
    <Button Content="Button 5" />
    <Button Content="Button 6" />
    <Button Content="Button 7" />
    <Button Content="Button 8" />
    <Button Content="Button 9" />
</UniformGrid>
```

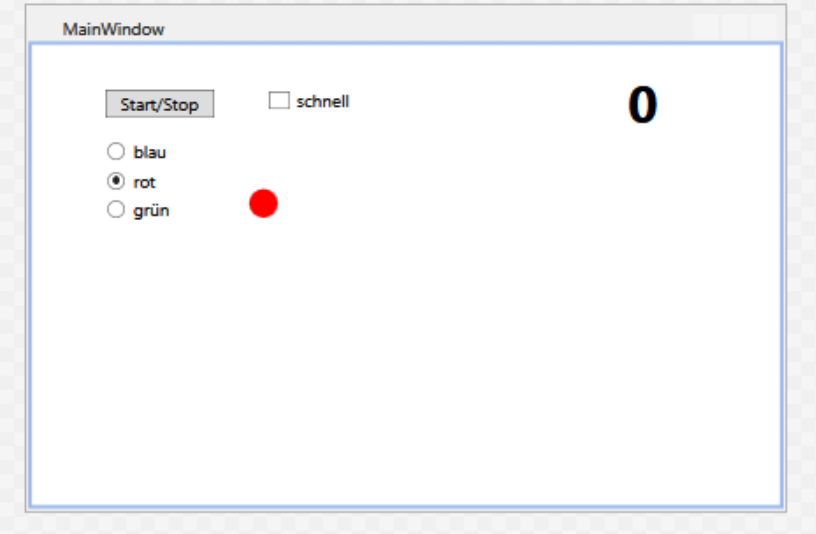


Jumping Ball with Canvas

Let a ball jump throw a canvas,
count how often the ball was clicked

Make it faster, change the colour, show the score.

GUI Designen

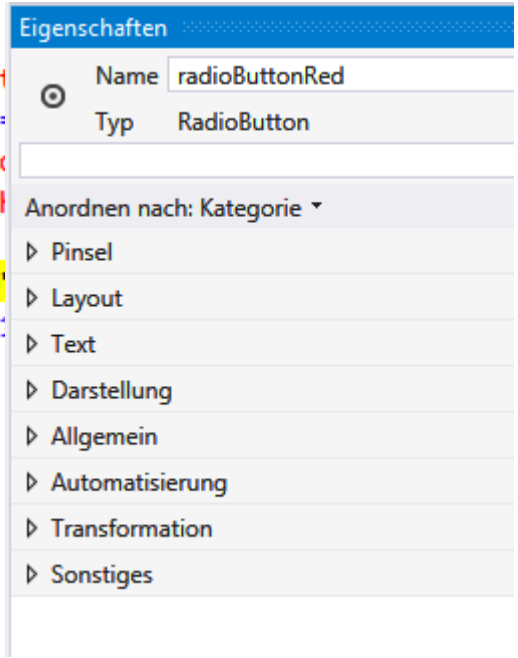


- Use
  - 1 Button
  - 3 RadioButton
  - 1 Checkbox
  - 1 Label
  - 1 Elippse (via XAML)

```
 <Ellipse Name="Ball"
MouseDown="Ball_MouseDown"
Height="20" Width="20"
Canvas.Left="150" Canvas.Top="100"
Fill="Red" />
```

XAML

```
<Window x:Class="_01_JumpingBall.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:_01_JumpingBall"
        mc:Ignorable="d"
        Title="MainWindow" Height="350" Width="525">
    <Canvas Name="TheCanvas">
        <Button Click="ButtonStartStop_Click" x:Name="ButtonStartStop" Content
        <CheckBox x:Name="CheckBoxFast" Content="schnell" HorizontalAlignment=
        <RadioButton x:Name="radioButtonBlue" Click="radioButtonBlue_Click" C
        <RadioButton x:Name="radioButtonRed" Click="radioButtonRed_Click" IsCh
        <RadioButton x:Name="radioButtonGreen" Click="radioButtonGreen_Click"
        <Ellipse Name="Ball" MouseDown="Ball_MouseDown" Height="20" Width="20
        <Label x:Name="labelScoue" Content="0" Canvas.Left="407" Canvas.Top="
    </Canvas>
</Window>
```




```
<Ellipse Name="Ball" MouseDown="Ball_MouseDown" Height="20"
Width="20" Canvas.Left="150" Canvas.Top="100" Fill="Red" />
```

XAML Datei

```
<Window x:Class="_01_JumpingBall.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:_01_JumpingBall"
        mc:Ignorable="d"
        Title="MainWindow" Height="350" Width="525">
    <Canvas Name="TheCanvas">
        <Button Click="ButtonStartStop_Click" x:Name="ButtonStartStop" Content="Start/Stop"
         HorizontalAlignment="Left" Margin="51,31,0,0" VerticalAlignment="Top" Width="75"/>
        <CheckBox x:Name="CheckBoxFast" Content="schnell" HorizontalAlignment="Left" Margin="163,31,0,0"
                  VerticalAlignment="Top"/>
        <RadioButton x:Name="radioButtonBlue" Click="radioButtonBlue_Click" Content="blau"
                  HorizontalAlignment="Left“ Margin="51,67,0,0" VerticalAlignment="Top"/>
        <RadioButton x:Name="radioButtonRed" Click="radioButtonRed_Click" IsChecked="True" Content="rot"
                  HorizontalAlignment="Left" Margin="51,87,0,0" VerticalAlignment="Top"/>
        <RadioButton x:Name="radioButtonGreen" Click="radioButtonGreen_Click" Content="grün"
                  HorizontalAlignment="Left" Margin="51,107,0,0" VerticalAlignment="Top"/>
        <Ellipse Name="Ball" MouseDown="Ball_MouseDown" Height="20" Width="20"
                  Canvas.Left="150" Canvas.Top="100" Fill="Red" />
        <Label x:Name="labelScoue" Content="0" Canvas.Left="407"
                  Canvas.Top="10" FontSize="36" FontWeight="Bold"/>
    </Canvas>
</Window>
```

MainWindow

```
public partial class MainWindow : Window
{
    DispatcherTimer timer = new DispatcherTimer();

    public MainWindow()
    {
        InitializeComponent();

        //Was soll geschehen alle 0,05 Sekunden (Eventhandler) - Funktion zuweisen
        timer.Tick += Physics;
        //Timer soll alle 0,05 Sekunden feuern
        timer.Interval = TimeSpan.FromSeconds(0.05);

    }

    private void radioButtonBlue_Click(object sender, RoutedEventArgs e)...

    private void radioButtonRed_Click(object sender, RoutedEventArgs e)...

    private void radioButtonGreen_Click(object sender, RoutedEventArgs e)
    {
        Ball.Fill = Brushes.Green;
    }

    private void ButtonStartStop_Click(object sender, RoutedEventArgs e)
    {
        //Gegenteil des aktuellen Zustands
        timer.IsEnabled = !timer.IsEnabled;
    }
    bool GoingRight = true;
    bool GoingDown = true;
```

Move Ball

```
bool GoingRight = true;
bool GoingDown = true;
private void Physics(object sender, EventArgs e)
{
    double speed = 3.0;

    if (CheckBoxFast.IsChecked.Value)
        speed = 10.0;
    MoveBallLeftRight(speed);
    MoveBallUpDown(speed);

}
private void MoveBallLeftRight(double speed)...
private void MoveBallUpDown(double speed)...

int score = 0;
private void Ball_MouseDown(object sender, MouseButtonEventArgs e)...
```

Move Ball

```
private void Physics(object sender, EventArgs e)...
private void MoveBallLeftRight(double speed)
{
    double x = Canvas.GetLeft(Ball);
    if (GoingRight)
        x += speed;
    else
        x -= speed;


    //wenn rechter Rand erreicht -> umdrehen
    if (x + Ball.Width > TheCanvas.ActualWidth)
    {
        GoingRight = false;
        //nicht über den Rand wandern lassen - sondern am Rand setzen
        x = TheCanvas.ActualWidth - Ball.Width;
        System.Media.SystemSounds.Asterisk.Play();
    }
    else if (x < 0.0)
    {
        GoingRight = true;
        //nicht über den Rand wandern lassen - sondern auf 0 (am Rand setzen)
        x = 0.0;
        System.Media.SystemSounds.Beep.Play();
    }

    Canvas.SetLeft(Ball, x);
}
private void MoveBallUpDown(double speed)...
```

```
private void MoveBallUpDown(double speed)
{
    double y = Canvas.GetTop(Ball);
    if (GoingDown)
        y += speed;
    else
        y -= speed;


    //wenn rechter Rand erreicht -> umdrehen
    if (y + Ball.Height > TheCanvas.ActualHeight)
    {
        GoingDown = false;
        //nicht über den Rand wandern lassen - sondern am Rand setzen
        y = TheCanvas.ActualHeight - Ball.Height;
        System.Media.SystemSounds.Exclamation.Play();
    }
    else if (y < 0.0)
    {
        GoingDown = true;
        //nicht über den Rand wandern lassen - sondern auf 0 (am Rand setzen)
        y = 0.0;
        System.Media.SystemSounds.Hand.Play();
    }

    Canvas.SetTop(Ball, y);
}

int score = 0;
private void Ball_MouseDown(object sender, MouseButtonEventArgs e)...
```

Count Score

```
bool GoingRight = true;
bool GoingDown = true;
private void Physics(object sender, EventArgs e)...
private void MoveBallLeftRight(double speed)...
private void MoveBallUpDown(double speed)...

int score = 0;
private void Ball_MouseDown(object sender, MouseButtonEventArgs e)
{
    if(timer.IsEnabled) {
        score++;
        labelScoue.Content = score;
    }
}
```

Next

Open WPF Data Binding & Command Binding

Solve the Examples in WPF Examples