**Layout Panel**

The layout of controls is very important and critical for application usability. It is used to arrange a group of GUI elements in your application.

Proper layout and positioning are vital parts of interactive, high-performance and user-friendly Windows applications

**Panels:**

All Panel controls are defined in the System.Windows.Controls namespace that resides in presentationframework.dll assembly.

Besides the root Window, a Panel is the base control that works as a parent control for other child controls.

WPF comes with the following five built-in panels:

* Canvas
* Dock Panel
* Grid
* Stack Panel
* Wrap Panel

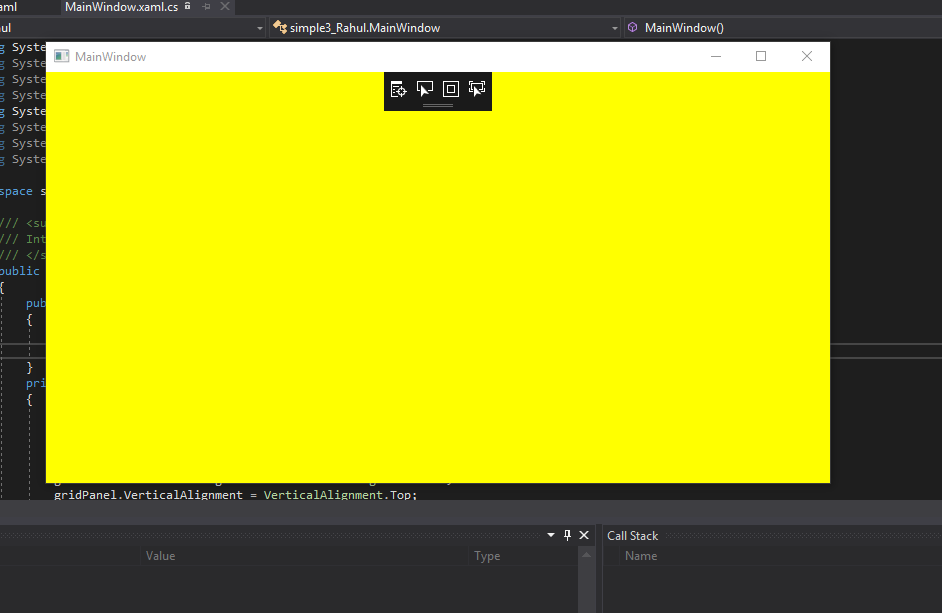
The purpose and use of these panels is different.

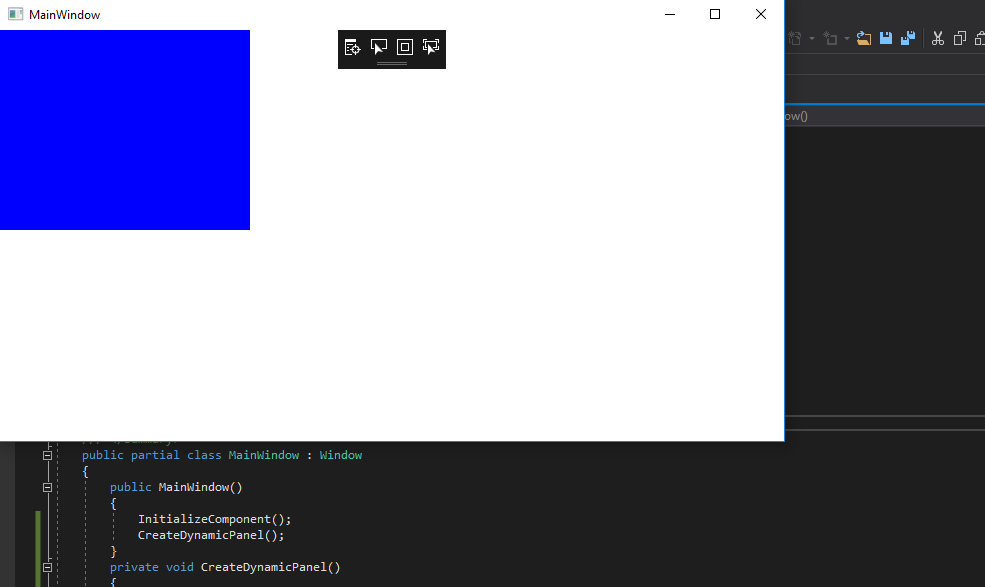
Each panel has a different way to position and reposition child controls placed within that panel.

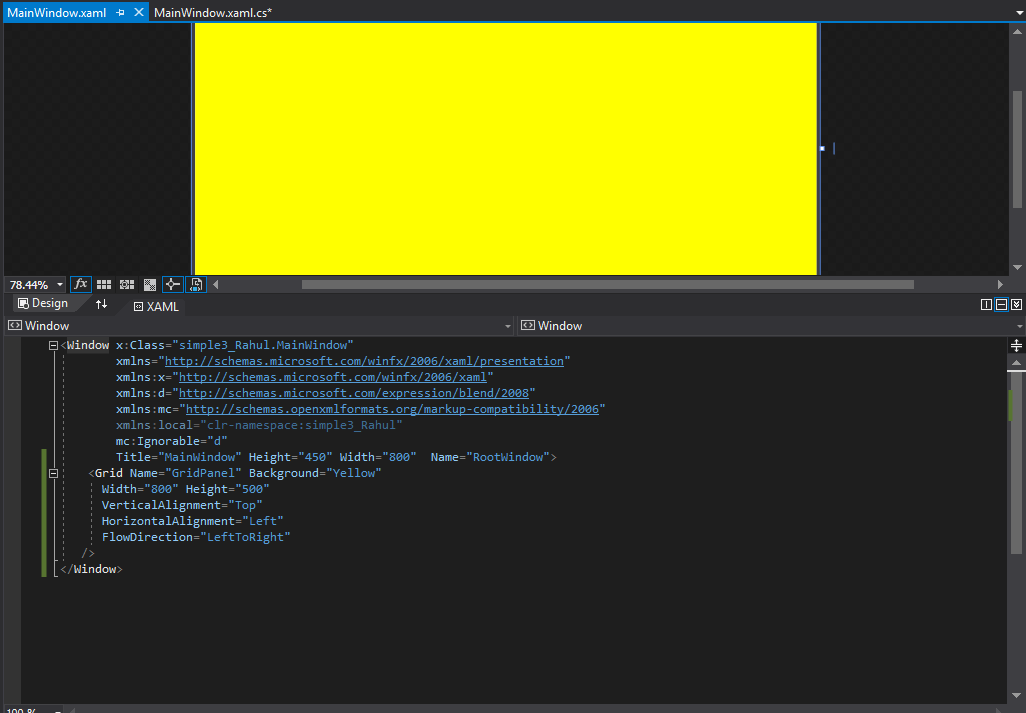
A Panel control may be represented in two ways.

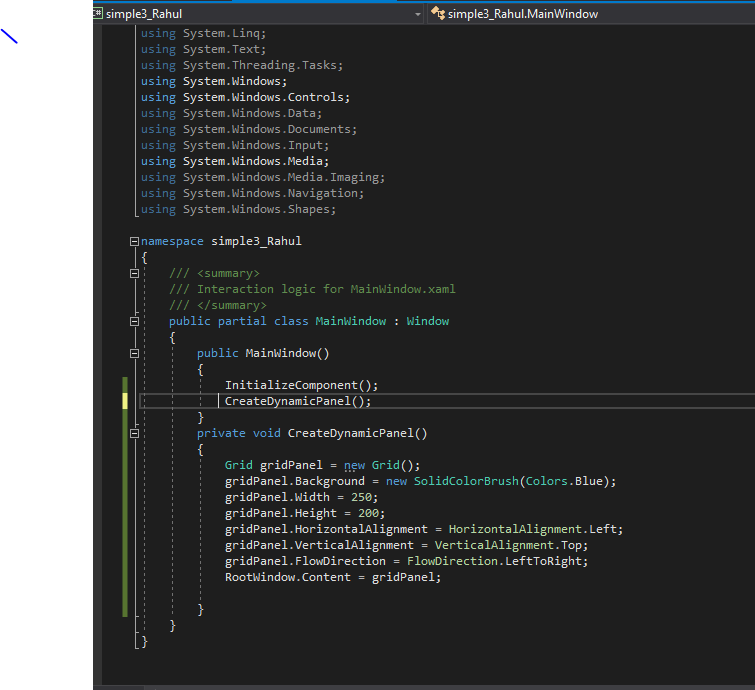
1. First, at design-time using XAML elements and attributes.
2. At run-time, using a WPF class and its properties.

**Code Snippet**

****

****

****

****

**Canvas Panel**

A Canvas panel is used to position child elements using coordinates relative to the canvas area. Here are some of the properties of Canvas panels.

The default values of the Height and Width properties of a Canvas are 0.

If you do not set these values, you will not see a canvas unless child elements are automatically resizable.

Child elements on a Canvas are never resized.

The vertical and horizontal alignments on child elements do not work. Child elements are placed on positions set by the Canvas Left, Top, Right and Bottom properties.

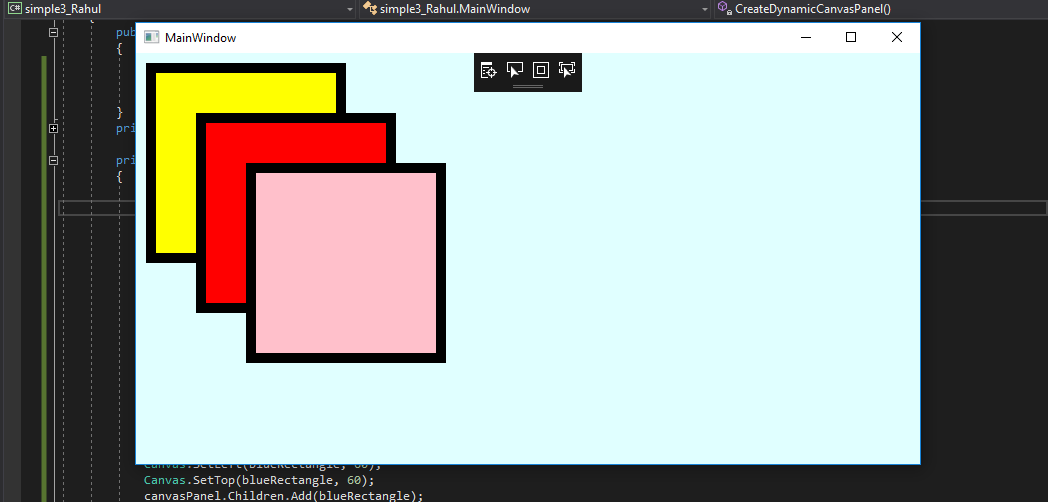
Margin does work partially. If a Left property of a Canvas is set, Right property does not work. If a Top property of a Canvas is set, the Bottom property does not work.

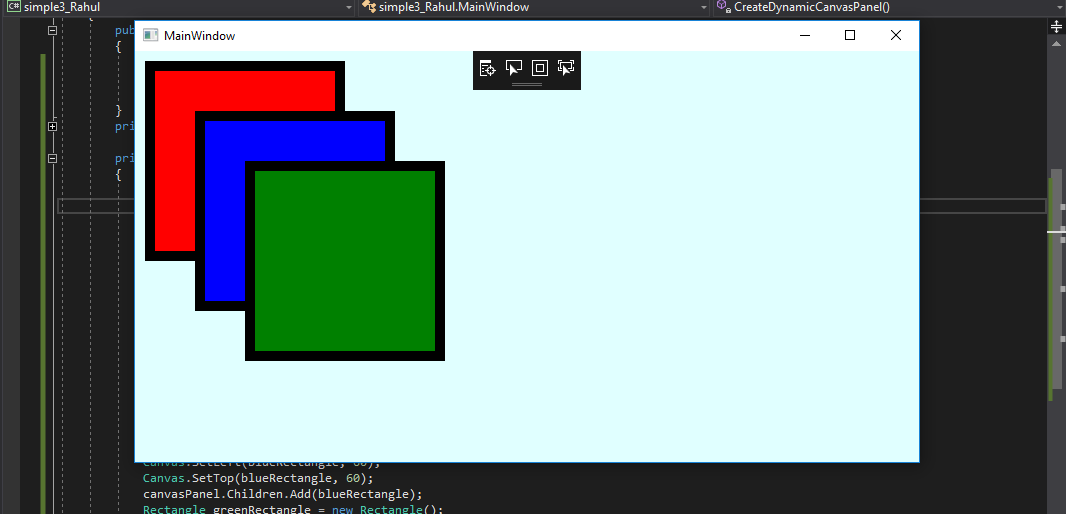
The Canvas control has three properties.

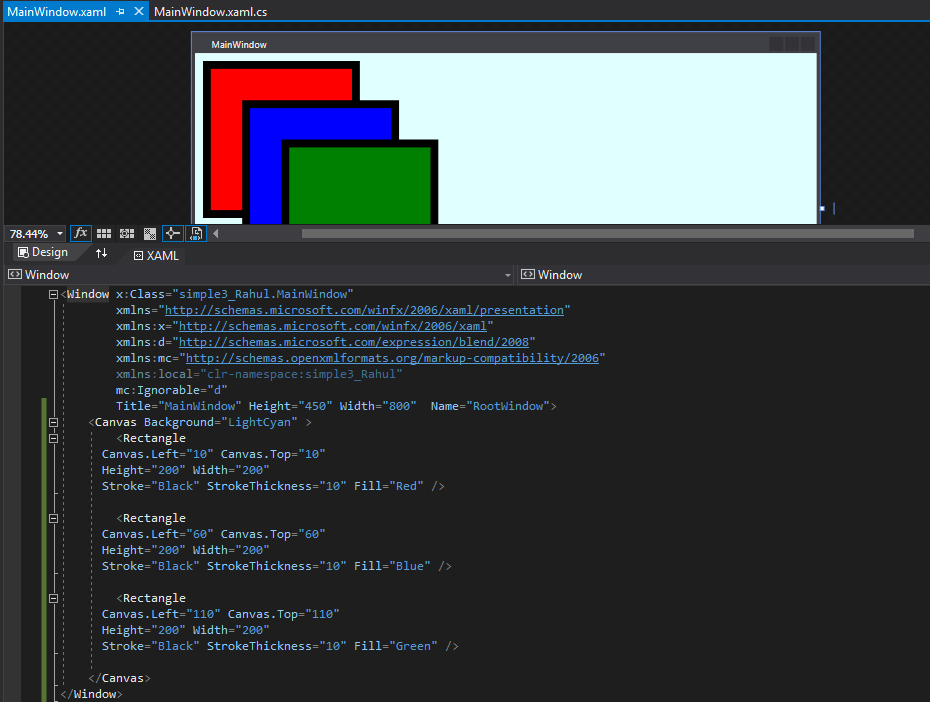
The Left property represents the distance between the left side of a control and its parent container Canvas.

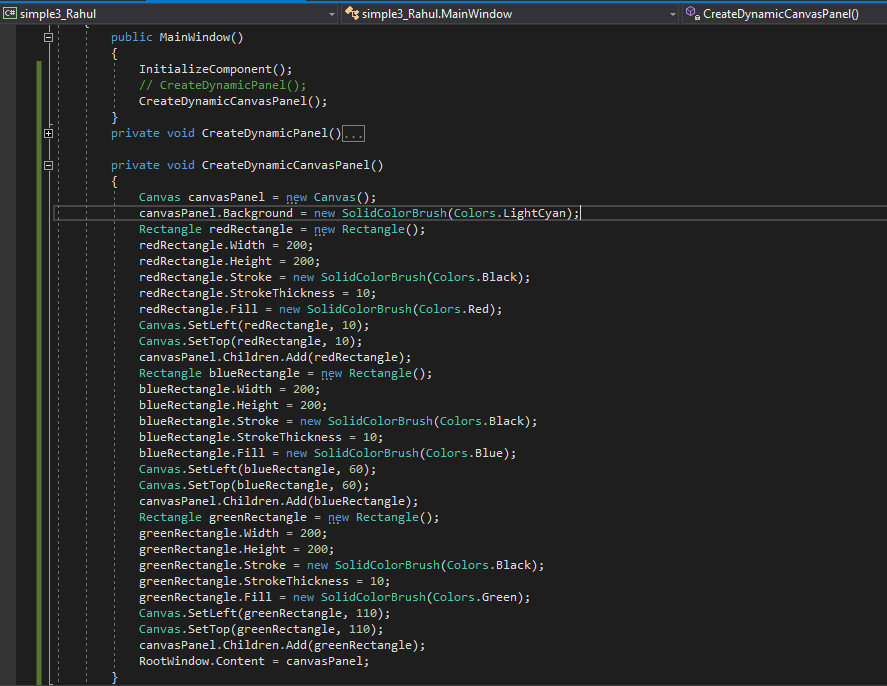
The Top property represents the distance between the top of a control and its parent container Canvas.

**Code Snippet and image**

****

****

****

****

**Dock Panel:**

A Dock Panel is used to dock child elements in the left, right, top and bottom positions of the relative elements.

The position of child elements is determined by the Dock property of the respective child elements and the relative order of those child elements.

The default value of the Dock property is left. The Dock property is of the type Dock enumeration that has Left, Right, Top and Bottom values.

The Last Child Fill property represents whether the last child element within a Dock Panel stretches to fill the remaining available space.

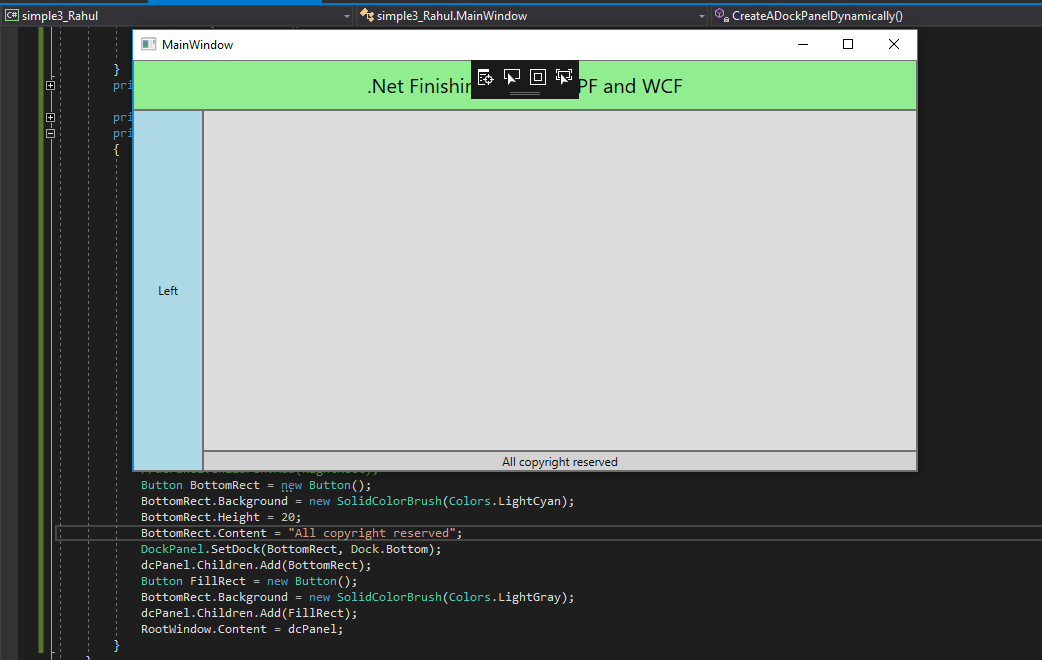
If the Last Child Fill property is set to true, the Dock property of the last element will be ignored.

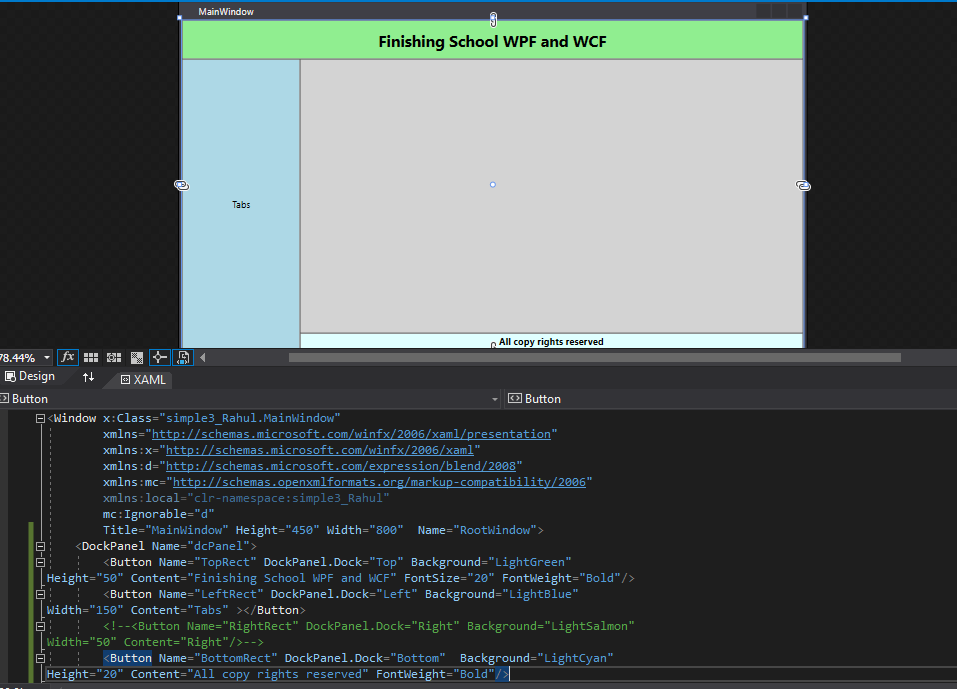
The Margin is applicable on Dock Panels.

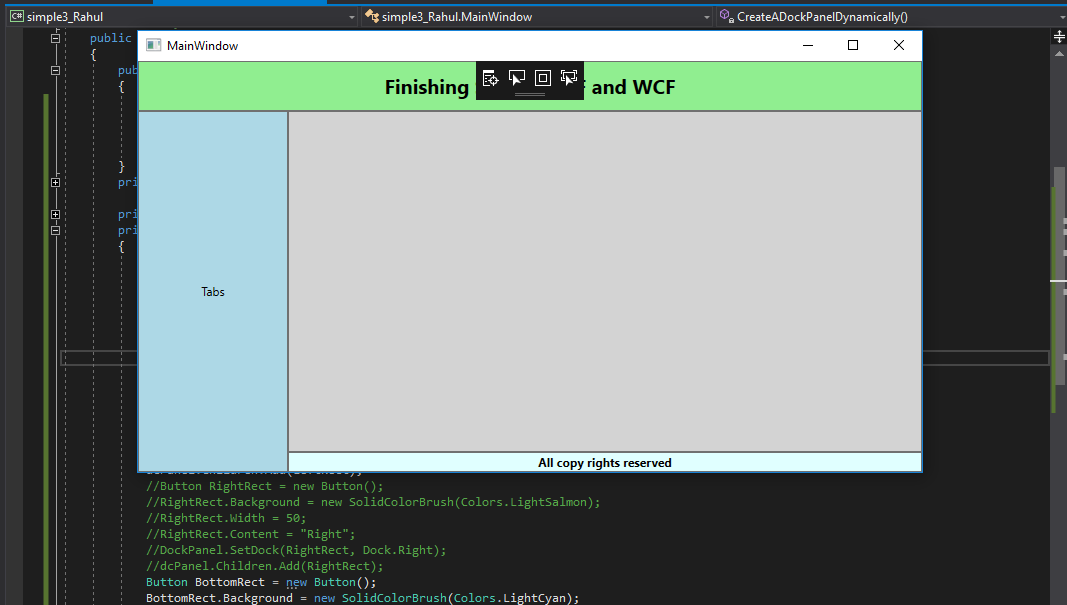
The Get Dock and Set Dock methods are used to get and set the dock value of an element.

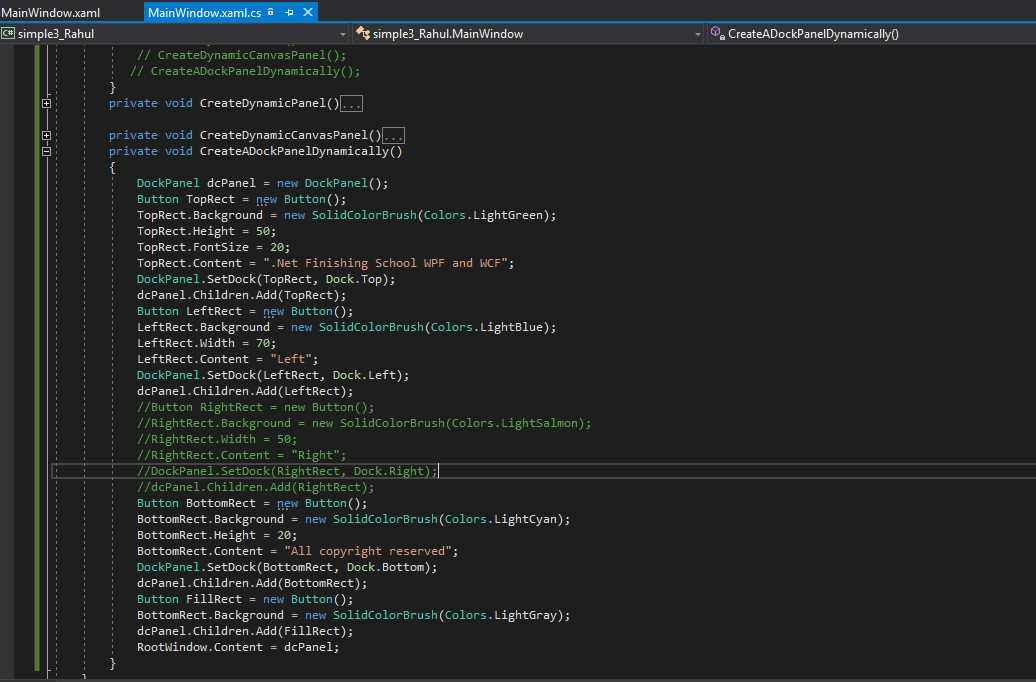
The Dock Panel class in WPF represents a Dock Panel control

**Code Snippet and images:**

****

****

****

****

**Grid**

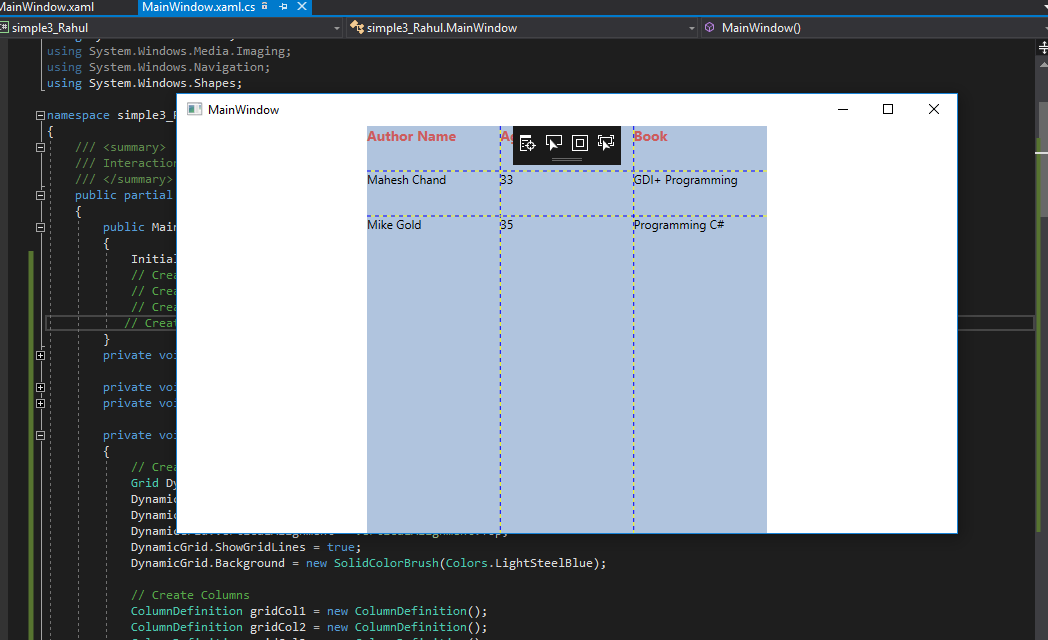
The Grid Panel is the most complicated but versatile panel among all panels.

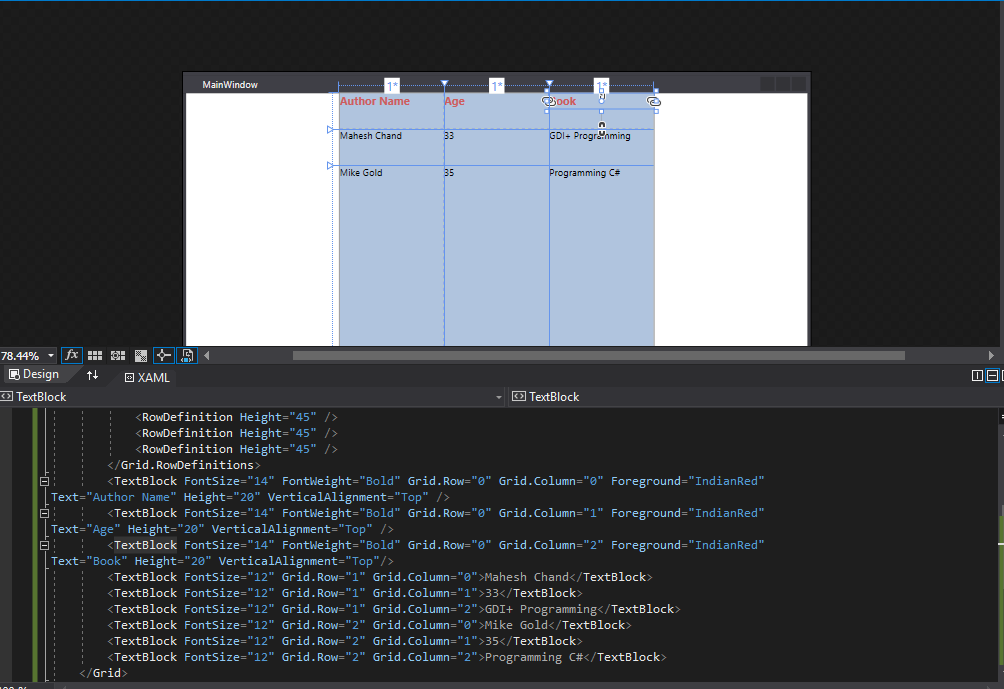
A Grid panel can be used to design complicated user interfaces where we need to place multiple elements in a tabular format of rows and columns.  
  
The Grid element in XAML represents a Grid panel.

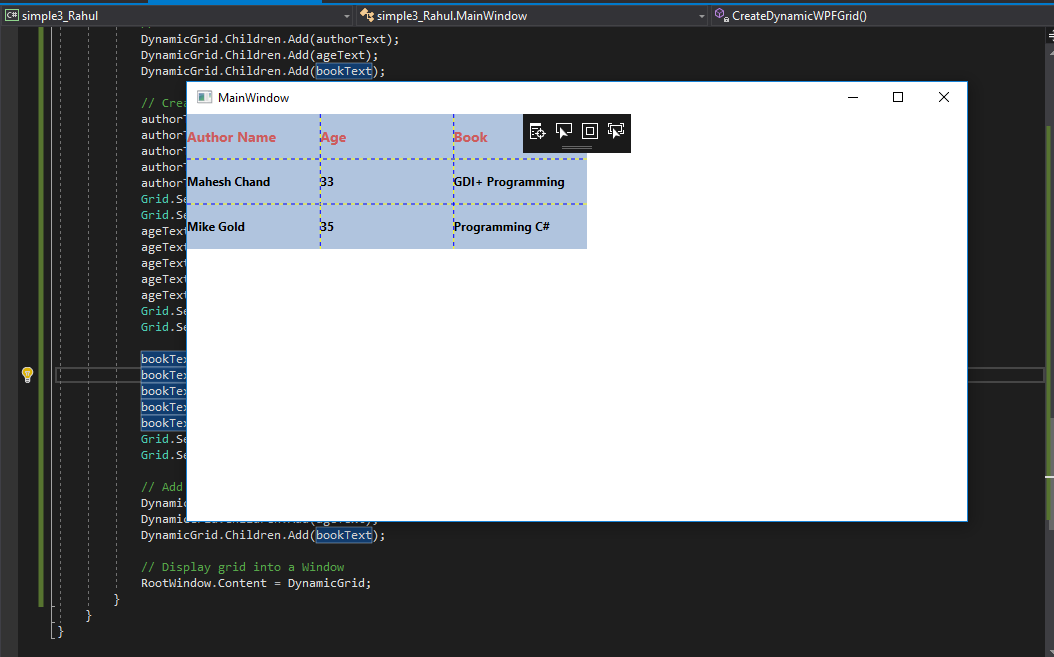
The following code snippet creates a Grid element and sets its background, width, height, vertical and horizontal alignment properties.

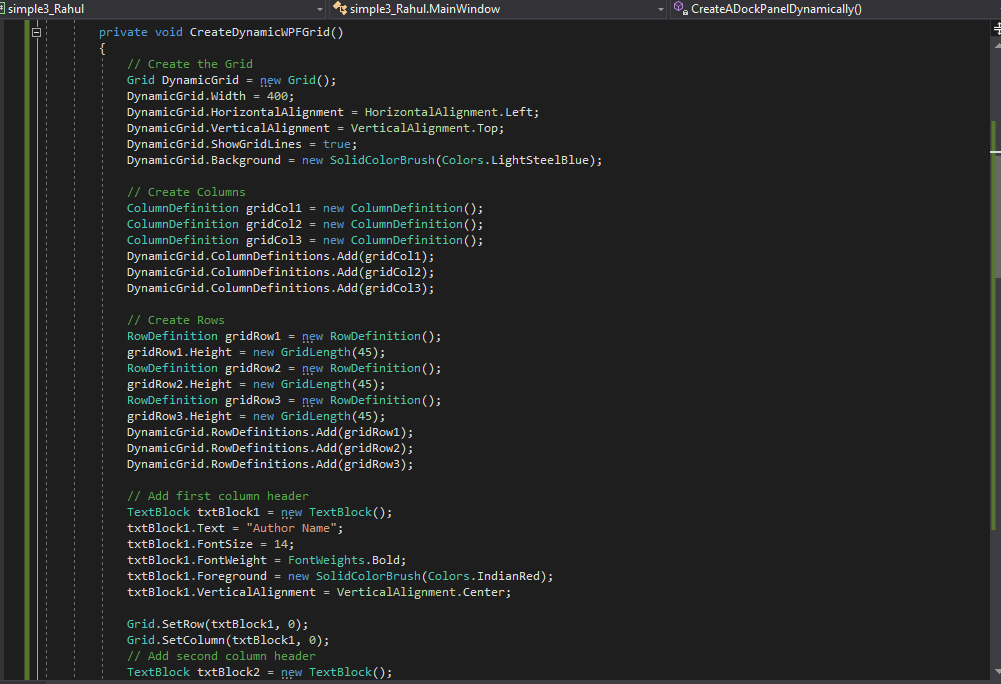
**Grid Properties**  
A Grid has the three major properties, Row Definitions, Column Definitions and Show Grid Lines. The Row Definitions property is a collection of Row Definitions. The Column Definitions property represents a collection of Column Definition. The Show Grid Lines property determines whether grid lines of a Grid panel are visible or not.  
  
**Create Grid**  
  
The Grid element in XAML represents a WPF Grid control. The following code snippet creates a Grid control, sets it width and foreground color and ensures the grid lines are visible.

**Create a Grid Dynamically**  
The Grid class in WPF represents a Grid control. The following code snippet in Listing 2 creates a Grid control, sets its width, horizontal alignment, vertical alignment and shows grid lines and sets a background color.

**Code snippet and images: **

****

****

****

**Stack Panel**

A StackPanel arranges child elements in a single line that can be oriented horizontally or vertically.

The StackPanel element in XAML represents a StackPanel.

The following code snippet creates a StackPanel at design-time using XAML.

The Orientation property represents the direction of the children that can be vertical or horizontal.

The default property of Orientation is vertical.

The following code snippet sets the Orientation property to Horizontal.

**Code Snippet and Images:**

private void CreateDynamicStackPanel()

{

StackPanel dynamicStackPanel = new StackPanel();

dynamicStackPanel.Width = 300;

dynamicStackPanel.Height = 200;

// dynamicStackPanel.Background = new SolidColorBrush(Colors.LightBlue);

dynamicStackPanel.Orientation = Orientation.Horizontal;

Ellipse redCircle = new Ellipse();

redCircle.Width = 100;

redCircle.Height = 100;

redCircle.Fill = new SolidColorBrush(Colors.Red);

dynamicStackPanel.Children.Add(redCircle);

Ellipse orangeCircle = new Ellipse();

orangeCircle.Width = 80;

orangeCircle.Height = 80;

orangeCircle.Fill = new SolidColorBrush(Colors.Orange);

dynamicStackPanel.Children.Add(orangeCircle);

Ellipse yellowCircle = new Ellipse();

yellowCircle.Width = 60;

yellowCircle.Height = 60;

yellowCircle.Fill = new SolidColorBrush(Colors.Yellow);

dynamicStackPanel.Children.Add(yellowCircle);

Ellipse greenCircle = new Ellipse();

greenCircle.Width = 40;

greenCircle.Height = 40;

greenCircle.Fill = new SolidColorBrush(Colors.Green);

dynamicStackPanel.Children.Add(greenCircle);

Ellipse blueCircle = new Ellipse();

blueCircle.Width = 20;

blueCircle.Height = 20;

blueCircle.Fill = new SolidColorBrush(Colors.Blue);

dynamicStackPanel.Children.Add(blueCircle);

RootWindow.Content = dynamicStackPanel;

}

<Window x:Class="simple3\_Rahul.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:simple3\_Rahul"

mc:Ignorable="d"

Title="MainWindow" Height="450" Width="800" Name="RootWindow" WindowStyle="ThreeDBorderWindow">

<StackPanel >

<Ellipse Width="100" Height="100" Fill="Red" />

<Ellipse Width="80" Height="80" Fill="Orange" />

<Ellipse Width="60" Height="60" Fill="Yellow" />

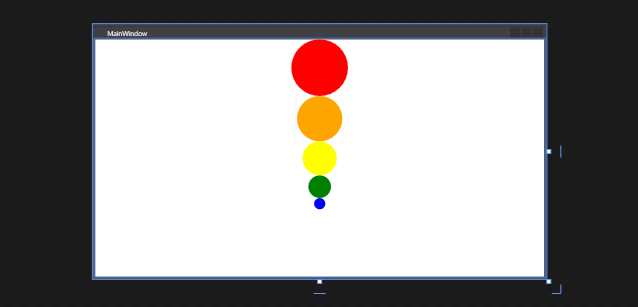
<Ellipse Width="40" Height="40" Fill="Green" />

<Ellipse Width="20" Height="20" Fill="Blue" />

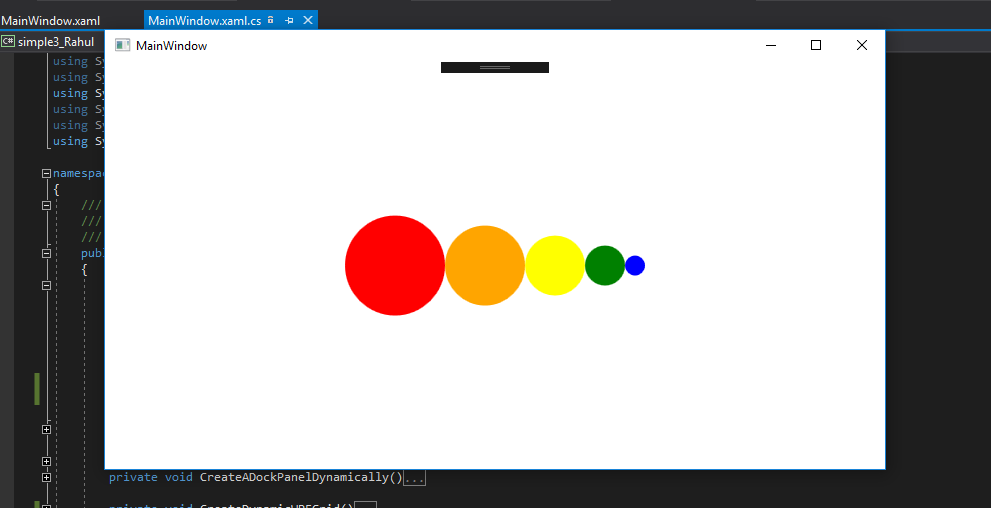
</StackPanel>

</Window>

**XAML Coding Image**

****

**Code behind coding image**

****

**Wrap Panel:**

WrapPanel is similar to StackPanel but it has an additional feature.

If elements that are stacked horizontally or vertically don't fit in the row or column they are in, the remaining elements will wrap around in the same sequence.  
  
The WrapPanel element in XAML represents a WrapPanel.

The following code snippet creates a WrapPanel and sets its height, width and background properties at design-time using XAML.

The Orientation property is used to wrap child items horizontally or vertically.

The ItemHeight and ItemWidth properties of WrapPanel are used to set a fixed uniform height and width of all items that are contained within a WrapPanel.

The following code snippet sets Orientation, ItemHeight and ItemWidth properties of a WrapPanel in XAML.

The following code will have every item within the WrapPanel with a height and width of 100 each.

**Code Snippet**

<Window x:Class="simple3\_Rahul.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:simple3\_Rahul"

mc:Ignorable="d"

Title="MainWindow" Height="450" Width="800" Name="RootWindow" WindowStyle="ThreeDBorderWindow">

<WrapPanel >

<Ellipse Width="100" Height="100" Fill="Red" />

<Ellipse Width="90" Height="90" Fill="Orange" />

<Ellipse Width="80" Height="80" Fill="Yellow" />

<Ellipse Width="70" Height="70" Fill="LightGreen" />

<Ellipse Width="60" Height="60" Fill="Green" />

<Ellipse Width="50" Height="50" Fill="LightBlue" />

<Ellipse Width="40" Height="40" Fill="Blue" />

<Ellipse Width="30" Height="30" Fill="Black" />

</WrapPanel>

</Window>

private void CreateDynamicWrapPanel()

{

// Create a WrapPanel and set its properties

WrapPanel dynamicWrapPanel = new WrapPanel();

dynamicWrapPanel.Background = new SolidColorBrush(Colors.LightBlue);

dynamicWrapPanel.Orientation = Orientation.Horizontal;

// Create Ellipses and add to StackPanel

Ellipse redCircle = new Ellipse();

redCircle.Width = 100;

redCircle.Height = 100;

redCircle.Fill = new SolidColorBrush(Colors.Red);

dynamicWrapPanel.Children.Add(redCircle);

Ellipse orangeCircle = new Ellipse();

orangeCircle.Width = 80;

orangeCircle.Height = 80;

orangeCircle.Fill = new SolidColorBrush(Colors.Orange);

dynamicWrapPanel.Children.Add(orangeCircle);

Ellipse yellowCircle = new Ellipse();

yellowCircle.Width = 60;

yellowCircle.Height = 60;

yellowCircle.Fill = new SolidColorBrush(Colors.Yellow);

dynamicWrapPanel.Children.Add(yellowCircle);

Ellipse greenCircle = new Ellipse();

greenCircle.Width = 40;

greenCircle.Height = 40;

greenCircle.Fill = new SolidColorBrush(Colors.Green);

dynamicWrapPanel.Children.Add(greenCircle);

Ellipse blueCircle = new Ellipse();

blueCircle.Width = 20;

blueCircle.Height = 20;

blueCircle.Fill = new SolidColorBrush(Colors.Blue);

dynamicWrapPanel.Children.Add(blueCircle);

// Display WrapPanel into a Window

RootWindow.Content = dynamicWrapPanel;

}

**Parallel Programming and async and await :**