

[](https://www.tutorialr.com/tutorials/)

Windows App SDK

Uniform Layout





# Uniform Layout

**Uniform Layout** shows how to create a **Uniform** **Panel** using **Windows App SDK**

## Step 1

Follow **Setup and Start** on how to get **Setup** and **Install** what you need for **Visual Studio 2022** and **Windows App SDK**.

|  |  |
| --- | --- |
| In **Windows 11** choose **Start** and then find or search for **Visual Studio 2022** and then select it. | Text  Description automatically generated |
| Once **Visual Studio 2022** has started select **Create a new project**. | **Graphical user interface, text  Description automatically generated** |
| Then choose the **Blank App, Packages (WinUI in Desktop)** and then select **Next**. | **Graphical user interface, text  Description automatically generated** |
| After that in **Configure your new project** type in the **Project name** as *UniformLayout*, then select a Location and then select **Create** to start a new **Solution**. | **Graphical user interface, text, application, email  Description automatically generated** |

## Step 2

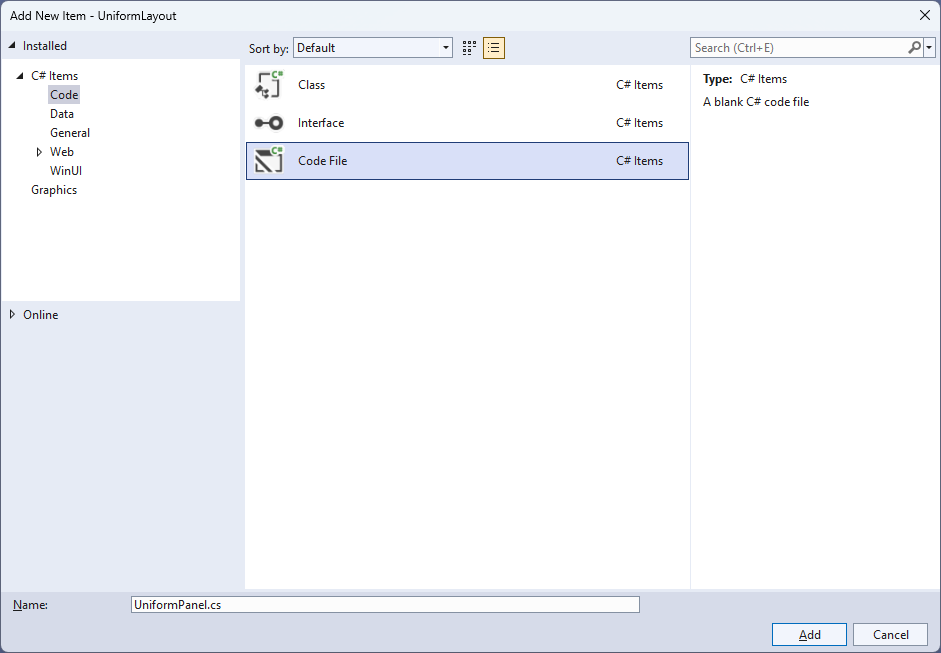
Then in **Visual Studio** within **Solution** **Explorer** for the **Solution**, right click on the **Project** shown below the **Solution** and then select **Add** then **New Item…**

Table

Description automatically generated with low confidence

## Step 3

Then in **Add New Item** from the **C# Items** list, select **Code** and then select **Code File** from the list next to this, then type in the name of *UniformPanel.cs* and then **Click** on **Add**.



## Step 4

|  |  |
| --- | --- |
| Then from **Solution** **Explorer** for the **Solution** double-click on **UniformPanel.cs** to see the **Code** for the **User Control**. |  |

## Step 5

You will now be in the **View** for the **Code** of *UniformPanel.cs*, within this type in the following **Code**:

using Microsoft.UI.Xaml;

using Microsoft.UI.Xaml.Controls;

using System;

using Windows.Foundation;

namespace UniformLayout;

public class UniformPanel : Panel

{

// Members, Dependency Properties & Properties

// Update Computed Values Method

// Measure Override Method

// Arrange Override Method

}

There are **using** statements for the **User Control**, a **namespace** for **UniformLayout** along with a **class** of **UniformPanel** that will represent the **User Control** and **Inherits** the **class** of **Panel**.

## Step 6

Then in the **namespace** of **UniformLayout** in the **class** of **UniformPanel** after the **Comment** of **// Members, Dependency Properties & Properties** type the following **Members**, **Dependency Properties** and **Properties**:

private int \_columns;

private int \_rows;

public static readonly DependencyProperty ColumnsProperty =

DependencyProperty.Register(nameof(Columns), typeof(int),

typeof(UniformPanel), new PropertyMetadata(0));

public static readonly DependencyProperty FirstColumnProperty =

DependencyProperty.Register(nameof(FirstColumn), typeof(int),

typeof(UniformPanel), new PropertyMetadata(0));

public static readonly DependencyProperty RowsProperty =

DependencyProperty.Register(nameof(Rows), typeof(int),

typeof(UniformPanel), new PropertyMetadata(0));

public int Columns

{

get { return (int)GetValue(ColumnsProperty); }

set { SetValue(ColumnsProperty, value); }

}

public int FirstColumn

{

get { return (int)GetValue(FirstColumnProperty); }

set { SetValue(FirstColumnProperty, value); }

}

public int Rows

{

get { return (int)GetValue(RowsProperty); }

set { SetValue(RowsProperty, value); }

}

The **Members** will be used to store values for the **User Control** and the **Dependency Properties** or **Properties** for the **User Control** can be customised for the **Uniform Panel**.

## Step 7

While still in the **namespace** of **UniformLayout** in the **class** of **UniformPanel** after the **Comment** of **// Update Computed Values Method** type the following **Method**:

private void UpdateComputedValues()

{

\_columns = Columns;

\_rows = Rows;

if (FirstColumn >= \_columns) FirstColumn = 0;

if ((\_rows == 0) || (\_columns == 0))

{

var row = 0;

var column = 0;

var count = Children.Count;

while (column < count)

{

var element = Children[column];

if (element.Visibility != Visibility.Collapsed)

{

row++;

}

column++;

}

if (row == 0) row = 1;

if (\_rows == 0)

{

if (\_columns > 0)

{

\_rows = (row + FirstColumn + (\_columns - 1)) / \_columns;

}

else

{

\_rows = (int)Math.Sqrt(row);

if ((\_rows \* \_rows) < row)

{

\_rows++;

}

\_columns = \_rows;

}

}

else if (\_columns == 0)

{

\_columns = (row + (\_rows - 1)) / \_rows;

}

}

}

The **Method** of **UpdateComputedValues** calculates the **Rows** and **Columns** and adjusts the layout accordingly based on the **Visibility** of the elements to produce the correct number of **Rows** and **Columns** needed by the **User Control**.

## Step 8

While still in the **namespace** of **UniformLayout** in the **class** of **UniformPanel** after the **Comment** of **// Measure Override Method** type the following **Method**:

protected override Size MeasureOverride(Size availableSize)

{

UpdateComputedValues();

var available = new Size(

availableSize.Width / \_columns,

availableSize.Height / \_rows);

double width = 0.0;

double height = 0.0;

int value = 0;

int count = Children.Count;

while (value < count)

{

var element = Children[value];

element.Measure(available);

var desiredSize = element.DesiredSize;

if (width < desiredSize.Width)

width = desiredSize.Width;

if (height < desiredSize.Height)

height = desiredSize.Height;

value++;

}

return new Size(width \* \_columns, height \* \_rows);

}

The **Method** of **MeasureOverride** will **Measure** the **Size** required to layout the **Children** of the **Panel** using the **Method** of **UpdateComputedValues** to respond to changes that require the layout to be updated and arranged accordingly for the **User Control**.

## Step 9

While still in the **namespace** of **UniformLayout** in the **class** of **UniformPanel** after the **Comment** of **// Arrange Override Method** type the following **Method**:

protected override Size ArrangeOverride(Size finalSize)

{

var rect = new Rect(0.0, 0.0,

finalSize.Width / \_columns, finalSize.Height / \_rows);

double width = rect.Width;

double value = finalSize.Width - 1.0;

rect.X += rect.Width \* FirstColumn;

foreach (var element in Children)

{

element.Arrange(rect);

if (element.Visibility != Visibility.Collapsed)

{

rect.X += width;

if (rect.X >= value)

{

rect.Y += rect.Height;

rect.X = 0.0;

}

}

}

return finalSize;

}

The **Method** of **ArrangeOverride** will position the **Children** of the **Panel** for the **User Control**.

## Step 10

|  |  |
| --- | --- |
| Within **Solution** **Explorer** for the **Solution** double-click on **MainWindow.xaml** to see the **XAML** for the **Main Window**. |  |

## Step 11

In the **XAML** for **MainWindow.xaml** there be some **XAML** for a **StackPanel**, this should be **Removed** by removing the following:

<StackPanel Orientation="Horizontal"

HorizontalAlignment="Center" VerticalAlignment="Center">

<Button x:Name="myButton" Click="myButton\_Click">Click Me</Button>

</StackPanel>

## Step 12

While still in the **XAML** for **MainWindow.xaml** above **</Window>**, type in the following **XAML**:

<local:UniformPanel Columns="4"

HorizontalAlignment="Center" VerticalAlignment="Center">

<Rectangle Width="100" Height="100" Fill="Red" Margin="10"/>

<Rectangle Width="100" Height="100" Fill="Orange"/>

<Rectangle Width="100" Height="100" Fill="Yellow"/>

<Rectangle Width="100" Height="100" Fill="Green"/>

<Rectangle Width="100" Height="100" Fill="Cyan"/>

<Rectangle Width="100" Height="100" Fill="Blue"/>

<Rectangle Width="100" Height="100" Fill="Magenta"/>

<Rectangle Width="100" Height="100" Fill="Purple"/>

</local:UniformPanel>

This **XAML** contains the **User Control** of **UniformPanel** with **Columns** set to **4** and the **Children** containing **Controls** for a **Rectangle** in various colours.

## Step 13

|  |  |
| --- | --- |
| Then, within **Solution** **Explorer** for the **Solution** select the arrow next to **MainWindow.xaml** then double-click on **MainWindow.xaml.cs** to see the **Code** for the **Main Window**. |  |

## Step 14

In the **Code** for **MainWindow.xaml.cs** there be a **Method** of **myButton\_Click(...)** this should be **Removed** by removing the following:

private void myButton\_Click(object sender, RoutedEventArgs e)

{

myButton.Content = "Clicked";

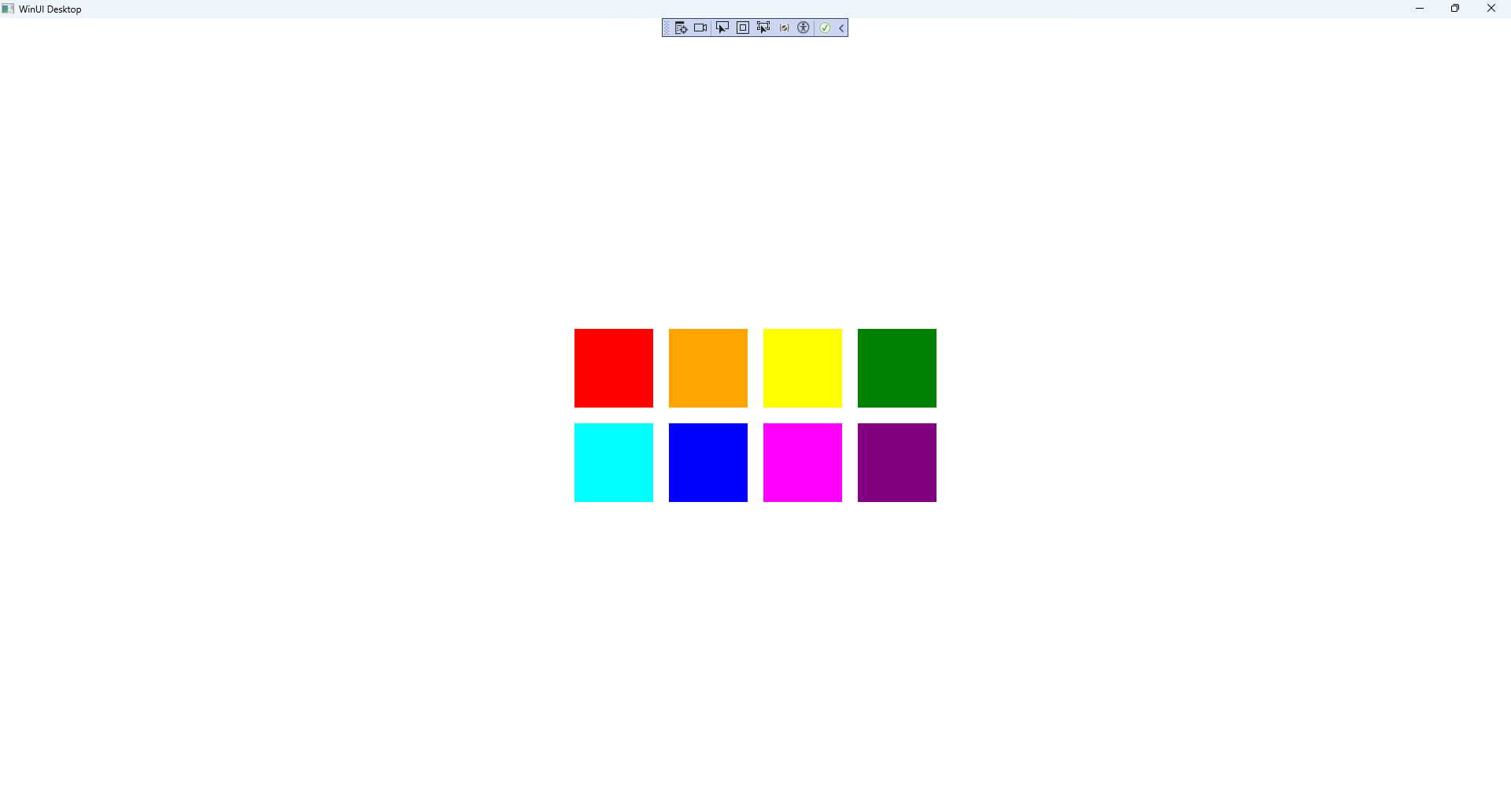
}

## Step 15

|  |  |
| --- | --- |
| That completes the **Windows App SDK** application. In **Visual Studio 2022** from the **Toolbar** select **UniformLayout (Package)** to **Start** the application. |  |

## Step 16

Once running you will see the **Uniform Panel** displayed.

****

## Step 17

|  |  |
| --- | --- |
| To **Exit** the **Windows App SDK** application, select the **Close** button from the top right of the application as that concludes this **Tutorial** for **Windows App SDK** from [tutorialr.com](https://tutorialr.com)! |  |